Virtuoso© SpectreRF Simulation Option User Guide

Product Version 5.1.41
June 2004

Contents

D
The RF Library 1077

F
Using QPSS Analysis Effectively

G
Introduction to the PLL library

H
Using Port in SpectreRF Simulations 1301

I
Analyzing Time-Varying Noise 1333

Preface

This manual assumes that you are familiar with RF circuit design and that you have some familiarity with SPICE simulation. It contains information about theVirtuoso® SpectreRF Simulation Option as it is used within the analog design environment. The SpectreRF Simulation Option is also known as SpectreRF in the software and in this manual.

SpectreRF analyses support the efficient calculation of the operating point, transfer function, noise, and distortion of common RF and communication circuits, such as mixers, oscillators, sample and holds, and switched capacitor filters.

Related Documents

The following documents can give you more information about SpectreRF and related products.

- To learn more about the Analog Circuit Design Environment, consult the *Virtuoso® Analog Circuit Design Environment User Guide*.
- To learn more about SpectreRF Simulator Option theoretical concepts, see *Virtuoso® SpectreRF Simulation Option Theory*.

1

SpectreRF Analyses

The SpectreRF analyses add capabilities to the Spectre circuit simulator, such as direct, efficient computation of steady-state solutions and simulation of circuits that translate frequency. You use the SpectreRF simulator analyses in combination with the Fourier analysis capability of Spectre simulation and with the Verilog®-A behavioral modeling language.

Periodic Analyses

The SpectreRF analyses add periodic large and small-signal analyses to Spectre simulation.

- Periodic Steady-State Analysis, PSS (Large-Signal)
- Periodic AC Analysis, PAC (Small-Signal)
- Periodic S-Parameter Analysis, PSP (Small-Signal)
- Periodic Transfer Function Analysis, PXF (Small-Signal)
- Periodic Noise Analysis, Pnoise (Small-Signal)

Periodic Steady-State (PSS) analysis is a large-signal analysis that directly computes the periodic steady-state response of a circuit. With PSS, simulation times are independent of the time constants of the circuit, so PSS can quickly compute the steady-state response of circuits with long time constants, such as high-Q filters and oscillators. You can also sweep frequency or other variables using PSS.

After completing a PSS analysis, the SpectreRF simulator can model frequency conversion effects by performing one or more of the periodic small-signal analyses, Periodic AC analysis (PAC), Periodic S-Parameter analysis (PSP), Periodic Transfer Function analysis (PXF) and Periodic Noise analysis (Pnoise). The periodic small-signal analyses are similar to the Spectre AC, SP, XF, and Noise analyses, but you can apply the periodic small-signal analyses to periodically driven circuits that exhibit frequency conversion. Examples of important frequency conversion effects include conversion gain in mixers, noise in oscillators, and filtering using switched-capacitors.

Therefore, with periodic small-signal analyses you apply a small signal at a frequency that may be noncommensurate (not harmonically related) to the small signal fundamental. This small signal is assumed to be small enough so that it is not distorted by the circuit.

Quasi-Periodic Analyses

The SpectreRF analyses add quasi-periodic large and small-signal analyses to Spectre simulation.

- Quasi-Periodic Steady-State Analysis, QPSS (Large-Signal)
- Quasi-Periodic AC Analysis, QPAC (Small-Signal)
- Quasi-Periodic S-Parameter Analysis, QPSP (Small-Signal)
- Quasi-Periodic Transfer Function Analysis, QPXF (Small-Signal)
- Quasi-Periodic Noise Analysis, QPnoise (Small-Signal)

Quasi-Periodic Steady-State (QPSS) analysis, a large-signal analysis, is used for circuits with multiple large tones. With QPSS, you can model periodic distortion and include harmonic effects. (Periodic small-signal analyses assume the small signal you specify generates no harmonics). QPSS computes both a large signal, the periodic steady-state response of the circuit, and also the distortion effects of a specified number of moderate signals, including the distortion effects of the number of harmonics that you choose.

With QPSS, you can apply one or more additional signals at frequencies not harmonically related to the large signal, and these signals can be large enough to create distortion. In the past, this analysis was called Pdisto analysis.

Quasi-Periodic Noise (QPnoise) analysis is similar to a transient noise analysis, except that it includes frequency conversion and intermodulation effects. QPnoise analysis is useful for predicting the noise behavior of mixers, switched-capacitor filters and other periodically or quasi-periodically driven circuits. QPnoise analysis linearizes the circuit about the quasi-periodic operating point computed in the prerequisite QPSS analysis. It is the quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion and intermodulation.

The Quasi-Periodic AC (QPAC), Quasi-Periodic S-Parameter (QPSP) and Quasi-Periodic Transfer Function (QPXF) analyses all work in a similar way for their respective type of analyses.

Envelope Following Analysis

Envelope Following analysis allows RF circuit designers to efficiently and accurately predict the envelope transient response of the RF circuits used in communication systems.

Large vs. Small Signal Analysis

SpectreRF provides a variety of time-varying small signal analysis for both periodic and quasi-periodic circuits. These small-signal analyses accurately model the frequency translation effects of time-varying circuits. Rather than using traditional small-signal analyses for circuits that exhibit frequency translation, such as amplifiers and filters, you can simulate these circuits using time-varying small-signal analyses.

Circuits designed to translate from one frequency to another include mixers, detectors, samplers, frequency multipliers, phase-locked loops, and parametric oscillators. Such circuits are commonly found in wireless communication systems.

Other circuits that translate energy between frequencies as a side effect include oscillators, switched-capacitor and switched-current filters, chopper-stabilized and parametric amplifiers, and sample-and-hold circuits. These circuits are found in both analog and RF circuits.

The quasi-periodic small-signal analyses accurately model the small signal characteristics of circuits with a quasi-periodic operating point, such as mixers with multiple LO frequencies or large RF inputs. The periodic small-signal analyses are more useful for circuits with a single fundamental frequency.

Applying a time-varying small-signal analysis is a two-step process.

1. First, the simulator ignores the small input or noise signals while performing a PSS or QPSS analysis to compute the steady-state response to the remaining large-signals, such as the LO or the clock. The initial PSS or QPSS analysis, linearizes the circuit about the time-varying large-signal operating point.

2. For each subsequent small-signal analysis, the simulator uses the time-varying operating point computed by the PSS or QPSS analysis to predict the circuit response to a small sinusoid at an arbitrary frequency. You can perform any number of small-signal analyses after calculating the time-varying large-signal operating point.

   The input signals for the small-signal analyses must be sufficiently small that the circuit does not respond to them in a significantly nonlinear fashion. You should use input signals that are at least 10 dB smaller than the 1 dB compression point. This restriction does not apply to the signals you apply in the large-signal analysis.

This two-step process is widely applicable because most circuits that translate frequency react in a strongly nonlinear manner to one stimulus, usually either the LO or the clock, while they react in a weakly nonlinear manner to other stimuli such as the inputs. A mixer is a typical example. Its noise and conversion characteristics improve if it is discontinuously switched between two states by the LO, yet it must respond linearly to the input signal over a wide dynamic range.

- To analyze a mixer with a small RF input and a single LO, you should use a PSS large-signal analysis followed by one or more of the PAC, PNoise, PSP or PXF small-signal analyses.
- If the mixer has a small RF input and a large blocker as well as the LO, then a QPSS analysis would be the more appropriate large-signal analysis. Follow the QPSS analysis with one or more of the QPAC, QPNoise, QPSP or QPXF small-signal analyses for the RF input.

Some circuits, such as frequency dividers, generate subharmonics. PSS can simulate the large-signal behavior of such circuits if you specify the period *T* to be that of the subharmonic. For other circuits, such as delta-sigma modulators, the periodically driven circuits respond chaotically, and you must use transient analysis rather than the PSS or QPSS analyses.

With the time-varying small-signal analyses such as QPAC or PXF, unlike traditional small-signal analyses such as AC or XF, there are many transfer functions between any single input and output due to harmonics. Usually, however, only one or two harmonics provide useful information. For example, when you analyze the down-conversion mixers found in receivers, you want to know about the transfer function that maps the input signal at the RF to the output signal at the IF, which is usually the LO minus the RF.

Unlike harmonic balance methods, the combination of PSS or QPSS and a time-varying small-signal analyses is efficient for circuits that respond in a strongly nonlinear manner to the LO or the clock. Consequently, you can use the SpectreRF simulations with strongly nonlinear circuits such as switched-capacitor filters, switching mixers, chopper-stabilized amplifiers, PLL-based frequency multipliers, sample-and-holds, and samplers.

Descriptions of SpectreRF Analyses

SpectreRF simulation provides unique analyses that are useful on RF circuits. These analyses directly compute the steady-state response and the small-signal behavior of circuits that exhibit frequency translation.

The individual SpectreRF analyses described in this section are

- PSS, periodic steady-state analysis (large-signal analysis)
  - PAC, periodic AC analysis (small-signal analysis)
  - PSP, periodic S-parameter analysis (small-signal analysis)
  - PXF, periodic transfer function analysis (small-signal analysis)
  - Pnoise, periodic noise analysis (small-signal analysis)
- QPSS, quasi-periodic steady-state analysis (large-signal analysis)
  - QPAC, quasi-periodic AC analysis (small-signal analysis)
  - QPSP, quasi-periodic S-parameter analysis (small-signal analysis)
  - QPXF, quasi-periodic transfer function analysis (small-signal analysis)
  - QPnoise, quasi-periodic noise analysis (small-signal analysis)
- Envlp, envelope following analysis

If a parameter or feature described in the *Spectre Reference* manual is not mentioned here, it is not accessible through the Analog Circuit Design Environment.

Periodic Steady-State Analysis

The PSS analysis computes the periodic steady-state response of a circuit at a specified fundamental frequency, with a simulation time independent of the time-constants of the circuit. The PSS analysis also determines the circuit's periodic operating point which is required starting point for the periodic time-varying small-signal analyses: PAC, PSP, PXF, and Pnoise. The PSS analysis works with both autonomous and driven circuits. See "PSS Analysis with Autonomous and Non-Autonomous Circuits" on page 35for more information.

The PSS Algorithm

SpectreRF simulation uses a technique called the shooting method to implement PSS analysis. This method is an iterative, time-domain method that finds an initial condition that directly results in a steady-state. It starts with a guess of the initial condition.

The shooting method requires few iterations if the final state of the circuit after one period is a near-linear function of the initial state. This is usually true even for circuits that have strongly nonlinear reactions to large stimuli (such as the clock or the local oscillator). Typically, shooting methods need about five iterations on most circuits, and they easily simulate the nonlinear circuit behavior within the shooting interval. This is the strength of shooting methods over other steady-state methods such as harmonic balance.

A new approach to Fourier analysis makes PSS more accurate with strongly nonlinear circuits than previous methods. Cadence's Fourier integral method approaches the accuracy of harmonic balance simulators for near-linear circuits, and far exceeds it for strongly nonlinear circuits.

In the case of a driven circuit, when you set `errpreset` to either `moderate` or `conservative`, SpectreRF automatically performs a high-order refinement after the shooting method. SpectreRF uses the Multi-Interval Chebyshev polynomial spectral algorithm (MIC) to refine the simulation results. When you set the `highorder` flag to `no`, MIC is turned off. However, when you set the `highorder` flag explicitly to `yes`, SpectreRF will try harder to converge. In a case where MIC fails to converge, SpectreRF falls back to the original PSS solution. For more information, see "The High-Order and Finite Difference Refinement Parameters" on page 36.

You can also use the finite difference (FD) refinement method after the shooting method to refine the simulation results. For more information, see "The High-Order and Finite Difference Refinement Parameters" on page 36

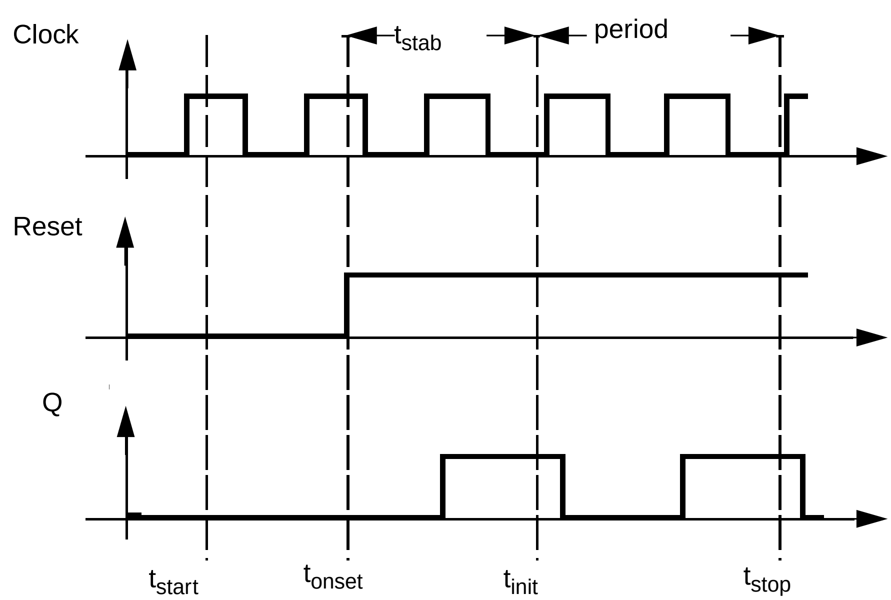
A PSS analysis consists of two phases

- The initial transient phase, a standard transient analysis to initialize the circuit.

- The shooting phase, to compute the periodic steady-state solution for the circuit using the shooting method.

As shown in Figure 1-1 on page 33, PSS starts by performing a standard transient analysis for the interval from t_{start} to $t_{\text{onset}} + t_{\text{stab}} + period$. The first interval begins at `tstart`, which is normally 0, and continues through the onset of periodicity t_{onset} for the independent sources. The onset of periodicity, which is automatically generated, is the earliest time for which all sources are periodic. The second interval is an optional user specified stabilization interval whose length is `tstab`. The final interval whose length is `period` for driven circuits, and estimated as `4x` `period` for autonomous circuits, has a special use for the autonomous PSS analysis—the autonomous PSS analysis monitors the waveforms in the circuit and develops a better estimate of the oscillation period. As is true for transient analysis, the DC solution is the initial condition for the PSS analysis unless you specify otherwise.

Figure 1-1 Initial Transient Analysis and Timing Relationships for PSS.



t

$\mathbf{t_{start}}$	Start time, `tstart`, for the transient analysis that you specify. This value can be negative or positive and defaults to 0.
$\mathbf{t_{onset}}$	Time when the last stimulus waveform becomes periodic. This value is automatically determined by the simulator for built-in independent sources.

t_{stab} Additional time that you specify to let the circuit settle. The `tstab` parameter is useful if you want a particular solution for a circuit that has multiple periodic solutions. A long t_{stab} might also improve convergence.

t_{init} Time when the PSS analysis begins. $t_{init} = t_{stab} + max(t_{start}, t_{onset})$

period Analysis interval for the PSS analysis determined from the fundamental frequency (`fund`) or the `period` that you specify.

t_{stop} Time when the PSS analysis ends. $t_{stop} = t_{init} + period$

Once the initial transient phase is complete, the shooting phase begins. During the shooting phase, the circuit is repeatedly simulated over one period while adjusting the initial condition (and the period for autonomous circuits) to find the periodic steady-state solution.

PSS analysis estimates the initial condition for subsequent transient analyses with an interval `period`. For an accurate estimate for this initial condition, the final state of the circuit must closely match its initial state. PSS then performs a transient analysis, prints the maximum mismatch, and generates an improved estimate of the necessary initial condition if the convergence criteria are not satisfied. This procedure repeats until the simulation converges.

Typically, the simulation requires three to five such iterations to reach the steady-state circuit response. After completion, if you request it, PSS computes the frequency-domain response.

For driven circuits, you can use `writepss` and `readpss` to save or reuse the steady state solution from a previously converged PSS simulation.

Within the Analog Circuit Design Environment, you define `tstart` in the *simulation Interval Parameters* section of the PSS Options form. Define `tstab` in the *Additional Time for Stabilization* section of the PSS Choosing Analysis form.

The PSS analysis determines the `period` value from the fundamental frequency (`fund`) you specify in the Fundamental Tones (PSS and QPSS) section of the PSS Choosing Analysis form.

Use the `skipdc` Initial-Condition Parameter to specify rampup before the transient (`tstab`) analysis. Use `skipdc` only for very special cases where there are several DC solutions in the system.

Set `skipdc=no` to calculate the initial solution using the usual DC analysis. (This is the default.) Set `skipdc=yes` to use either the initial solution given in the `readic` parameter file or the values specified on the `ic` statements.

When you set `skipdc=sigrampup`, independent source values start at 0 and ramp up to their initial values during the first phase of the simulation. After the rampup phase, waveform production is enabled in the time-varying independent source.The rampup simulation is from t_{start} to `time=0` seconds. The main simulation is from `time=0` seconds to t_{stab}. If you do not specify the t_{start} parameter, the default start time is set to `-0.1*` t_{stab}.

The initial transient analysis can direct the circuit to a particular steady-state solution and avoid unwanted solutions. The initial transient simulation also helps convergence by eliminating large, but fast decaying, modes in the circuit. In Figure 1-1 on page 33, the initial transient analysis is executed from the starting time to t_{stop}.

You can save the initial transient results by setting `saveinit` to `yes` on the PSS Choosing Analysis form. The steady-state results are always computed for the period from t_{init} to t_{stop}. By default, t_{start} (`tstart`) and t_{stab} (`tstab`) are set to zero, while t_{init}, t_{onset}, and t_{stop} are always automatically computed.

In some circuits, the linearity of the relationship between the initial and final states depends on when the shooting interval begins. Theoretically, the starting time of the shooting interval does not matter, as long as it begins after the stimuli become periodic. Practically, it is better to start the shooting interval when signals are quiescent or changing slowly and to avoid starting times when the circuit displays strongly nonlinear behavior. Choosing a poor starting time slows the analysis.

PSS Analysis with Autonomous and Non-Autonomous Circuits

The PSS analysis works with both autonomous (non-driven) and driven (non-autonomous) circuits.

- Autonomous circuits are time-invariant circuits with time-varying responses. Thus, autonomous circuits generate non-constant waveforms even though they are not driven by a time-varying stimulus.The most common autonomous circuit is an oscillator.
- Driven circuits require some time-varying stimulus to generate a time-varying response. Common driven circuits include amplifiers, filters, mixers, and so on.

For autonomous circuits, since they do not have drive signals and you do not know the actual period of oscillation in advance, you estimate the oscillation period and the PSS analysis computes the precise period along with the periodic solution waveforms. PSS analysis of an autonomous circuit, requires you to specify a pair of nodes, `p` and `n`. In fact this is how PSS analysis determines whether it is being applied to an autonomous or a driven circuit. If the pair of nodes is supplied, the PSS analysis assumes the circuit is autonomous; if not, the circuit is assumed to be driven. See Chapter 5, “Simulating Oscillators,” for an example.

For driven circuits, you specify either the period of the analysis, the `period` parameter, or its corresponding fundamental frequency, the `fund` parameter. The `period` parameter value must be an integer multiple of the period of the drive signals.

The errpreset parameter works differently for autonomous (non-driven) and driven (non-autonomous) circuits. For detailed information, see "The errpreset Parameter in PSS Analysis" on page 38.

Simulation Accuracy Parameters

The accuracy of your simulation results depends on your accuracy parameter settings, not on the number of harmonics you request. It is recommended that you adjust the accuracy parameters using the `errpreset` parameter settings as described in "The errpreset Parameter in PSS Analysis" on page 38.

Several parameters determine the accuracy of the PSS analysis.The `steadyratio` parameter specifies the maximum allowed mismatch between node voltages or current branches from the beginning of the steady-state period to its end. The `steadyratio` value is multiplied by the `lteratio` and `reltol` parameter values to determine the convergence criterion. The `reltol` and `abstol` parameters control the accuracy of the discretized equation solution. These parameters determine how well charge is conserved and how accurately steady-state or equilibrium points are computed. You can set the integration error, or the errors in the computation of the circuit dynamics (such as time constants), relative to the `reltol` and `abstol` parameters by setting the `lteratio` parameter.

You can follow the progress of the steady-state iterations because the relative convergence norm is printed in the simulation log file along with the actual mismatch value at the end of each iteration. The iterations continue until the convergence norm value is below one.

Plotting the Current Spectrum

In order to plot the current or power spectrum for a PSS analysis, or for any of the periodic small signal analyses, you must set up the analysis to put the terminal currents into the rawfiles associated with the steady-state results. To do this, choose *Outputs - Save All* to display the Save Options form where you set `currents=selected` and `useprobes=yes.` You also need to use an `iprobe` component from the `analogLib` library.

The High-Order and Finite Difference Refinement Parameters

The `highorder` parameter specifies the use of the high-order Chebyshev refinement method (MIC) which is used after the shooting method to refine the shooting method's steady-state solution. The `highorder` flag applies only to the driven case and when `errpreset` is set to either `moderate` or `conservative`.

- When `highorder=no`, the MIC method is turned off.

- When `highorder=yes`, the MIC method tries harder to converge.
- When `errpreset` is set to either `moderate` or `conservative` and `highorder` is not set by the user, MIC is used but it does not aggressively try to converge unless `highorder` is explicitly set to `yes`.

The `finitediff` parameter specifies the use of the finite difference (FD) refinement method which is used after the shooting method to refine the simulation results. This flag is only meaningful when the `highorder` flag is set to `no`.

- When `finitediff=no`, the FD method is turned off.
- When `finitediff=yes`, PSS applies the FD refinement method.
- When `finitediff=refine`, PSS applies the FD refinement method and tries to refine the time steps.

When the simulation uses the 2nd-order method, uniform 2nd order gear is used. When `readpss` and `writepss` are used to re-use PSS results, the `finitediff` parameter automatically changes from `no` to `yes`.

Usually FD will eliminate the above mismatch in node voltages or current branches. It can also refine the grid of time steps. In some cases, the numerical error of the linear solver still introduces a mismatch. In this case, you can adjust the *steadyratio* parameter to a smaller value to activate a tighter tolerance for the iterative linear solver.

The *maxacfreq* parameter automatically adjusts *maxstep* to reduce errors due to aliasing in subsequent periodic small-signal analyses. By default, *maxacfreq* is four times the frequency of the largest harmonic you request, but it is never less than 40 times the fundamental.

The `relref` parameter determines how the relative error is treated. Table 1-1 lists the `relref` parameter options.

Table 1-1 The relref Parameter Options

Option	Definition
relref=pointlocal	Compares the relative errors in quantities at each node to that node alone.
relref=alllocal	Compares the relative errors at each node to the largest values found for that node alone for all past time.
relref=sigglobal	Compares relative errors in each of the circuit signals to the maximum for all signals at any previous point in time.

Option	Definition
relref=allglobal	Same as `relref=sigglobal` except that it also compares the residues (KCL error) for each node to the maximum of that node’s past history.

The errpreset Parameter in PSS Analysis

The `errpreset` parameter quickly adjusts several simulator parameters to fit your needs. In most cases, `errpreset` should be the only parameter you need to adjust.

Using the errpreset Parameter With Driven Circuits

If your driven (non-autonomous) circuit includes only one periodic tone and you are only interested in obtaining the periodic operating point, set `errpreset` to *`liberal`*. The liberal setting produces reasonably accurate results with the fastest simulation speed.

If your driven circuit contains more than one periodic tone and you are interested in intermodulation results, set `errpreset` to *`moderate`*. The moderate setting produces very accurate results.

If you want an extremely low noise floor in your simulation results and accuracy is your main interest, set `errpreset` to *`conservative`*.

For both *`moderate`* and *`conservative`* settings, the Multi-Interval Chebyshev (MIC) algorithm is activated automatically unless you explicitly set `highorder=no`. If MIC has difficulty converging, the simulator reverts back to the original method. If you set `highorder=yes`, MIC will continue to attempt to converge.

Table 1-2 shows the effect of the `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the values of the other parameters used with driven circuits.

Table 1-2 Parameter Default Values and Estimated Numerical Noise Floor for `errpreset` Settings in Driven Circuits

errpreset	reltol	relref	method	Iteratio	steadyratio	noisefloor
liberal	$1e^{-3}$	sigglobal	traponly	3.5	0.001	-70 dB
moderate	$1e^{-3}$	alllocal	gear2only and MIC	3.5	0.001	-120 dB

errpreset	reltol	relref	method	Iteratio	steadyratio	noisefloor
conservative	$1e^{-4}$	alllocal	gear2only and MIC	10	0.01	-200 dB

These `errpreset` settings include a default `reltol` value which is an enforced upper limit for `reltol`. The only way to decrease the `reltol` value is in the `options` statement. The only way to *relax* the `reltol` value is to change the `errpreset` setting.

The estimated numerical `noisefloor` value is for a weakly nonlinear circuit with a successful MIC simulation. For a linear circuit, the `noisefloor` is even lower. For a very nonlinear circuit, you might need to tighten the `psaratio` parameter or increase the `maxacfreq` parameter to achieve this `noisefloor` value.

Using the errpreset Parameter With Autonomous Circuits

For an autonomous circuit, if you want a fast simulation with reasonable accuracy, set `errpreset` to *`liberal`*. For greater accuracy, set `errpreset` to *`moderate`*. For greatest accuracy, set `errpreset` to *`conservative`*.

Table 1-3 shows the effect of the `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the values of the other parameters used with autonomous circuits.

Table 1-3 Parameter Default Values for `errpreset` Settings in Autonomous Circuits

errpreset	reltol	relref	method	Iteratio	steadyratio	maxstep
liberal	$1e^{-3}$	sigglobal	traponly	3.5	0.001	period/50
moderate	$1e^{-4}$	alllocal	gear2only	3.5	0.1	period/200
conservative	$1e^{-5}$	alllocal	gear2only	10	0.1	period/400

These `errpreset` settings include a default `reltol` value which is an enforced upper limit for `reltol`. The only way to decrease the `reltol` value is in the `options` statement. The only way to *relax* the `reltol` value is to change the `errpreset` setting. Spectre sets the `maxstep` parameter value so that it is no larger than the value given in Table 1-3.

Except for the `reltol` and `maxstep` values, the `errpreset` setting does not change any parameter values you have explicitly set. The actual values used for the PSS analysis are given in the log file.

Parameters for PSS Analysis

Simulation Interval Parameters

period (s)	Steady-state analysis period (or its estimate for autonomous circuits).
fund (Hz)	Alternative to period specification. Steady state analysis fundamental frequency (or its estimate for autonomous circuits).
tstab=0.0 (s)	Extra stabilization time after the onset of periodicity for independent sources.
tstart=0.0 (s)	Initial transient analysis start time.

Time-Step Parameters

maxstep (s)	Maximum time step. Default derived from `errpreset`.
maxacfeq	Maximum frequency requested in a subsequent periodic small-signal analysis. Default derived from `errpreset` and `harms`.
step=0.001*period (s)	Minimum time step that would be used solely to maintain the aesthetics of the results.

Initial-Condition Parameters

ic=all	Used to set initial conditions. Possible values are `dc`, `node`, `dev`, or `all`.
skipdc=no	Determines how the initial solution is calculated. Possible values are `no`, `yes` and `sigrampup`.
readic	File that contains initial condition.

Convergence Parameters

readns	File that contains the estimate of the initial transient solution.
cmin=0 F	Minimum capacitance from each node to ground.

Output Parameters

harms=9	Number of harmonics to output when `outputtype=freq` or `outputtype=all`.
harmsvec=[...]	Array of desired harmonics. Alternate form of `harms` that allows you to select specific harmonics.

outputtype=time	Output type. Possible values are `all`, `time` or `freq`. (Use `time` for PSS and `freq` for QPSS.)
save	Signals to output. Possible values are `all`, `lvl`, `allpub`, `lvlpub`, `selected`, or `none`.
nestlvl	Levels of subcircuits to output.
oppoint=no	Should operating point information be computed for initial timestep, and if so, where should it be sent. Possible values are `no`, `screen`, `logfile`, or `rawfile`.
skipstart=starttime s	The time to start skipping output data.
skipstop=stoptime s	The time to stop skipping output data.
skipcount	Save only one of every `skipcount` points.
strobeperiod (s)	The output strobe interval (in seconds of transient time).
strobedelay=0 (s)	The delay (phase shift) between the `skipstart` time and the first strobe point.
compression=no	Do data compression on output. Possible values are `no` or `yes`.
saveinit=no	If set, the waveforms for the initial transient before steady state are saved. Possible values are `no` or `yes`.

State-File Parameters

write	File in which to write the initial transient solution (before steady-state).
writefinal	File in which to write the final transient solution in steady-state.
swapfile	Temporary file that holds steady-state information. Tells Spectre to use a regular file rather than virtual memory to hold the periodic operating point. Use this option if Spectre does not have enough memory to complete the analysis.
writepss	File in which to write the converged steady-state solution. Also sets `finitediff` to yes to improve PSS results.
readpss	File from which to read a previously converged steady-state solution. PSS loads the solution and checks only the residue of the circuit equations. The solution is re-used if the residue is satisfying. Otherwise, the solution is re-converged using the finite difference method.

Integration Method Parameter

method	Integration method. Default derived from `errpreset`. Possible values are `euler`, `trap`, `traponly`, `gear2`, or `gear2only`.

Accuracy Parameters

errpreset=moderate	Selects a reasonable collection of parameter settings. Possible values are `liberal`, `moderate` or `conservative`.
relref	Reference used for the relative convergence criteria. Default derived from `errpreset`. Possible values are `pointlocal`, `alllocal`, `sigglobal`, or `allglobal`.
lteratio	Ratio used to compute LTE tolerances from the Newton tolerance. Default derived from `errpreset`.
steadyratio	Ratio used to compute steady state tolerances from the LTE tolerance. Default derived from `errpreset`.
maxperiods	`maxperiods`=20 for driven PSS, 50 for autonomous PSS Maximum number of simulated periods to reach steady-state.
itres=1e^{-4}	Relative tolerance for linear solver.
finitediff	Options for finite difference method refinement after shooting method for driven circuits. Possible values are no, `yes` or `refine`.
highorder	After low-order convergence, perform a high-order refinement using the Multi-Interval Chebyshev (MIC) polynomial spectral algorithm. Possible values are no or `yes`.
psaratio=1	Ratio used to compute high-order polynomial spectral accuracy from Newton tolerance.
maxorder=16	The maximum order of the Chebyshev polynomials used in waveform approximation. Possible values range from 2 to 16.
fullpssvec	Use the full vector containing solutions at all PSS time steps in the linear solver. Default derived from the size of the equation and the property of the PSS time steps. Possible values are no or `yes`.

Annotation Parameters

stats=no	Analysis statistics. Possible values are no or `yes`.
annotate=sweep	Degree of annotation. Possible values are no, `title`, `sweep`, `status`, or `steps`.
title	Analysis title.

Newton Parameters

maxiters=5	Maximum number of iterations per time step.
restart=yes	If any condition has changed, restart the DC solution from scratch. If not, use the previous solution as the initial guess. Possible values are no or yes.

Circuit Age Parameter

circuitage (Years)	Stress Time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.

Periodic AC Analysis

The Periodic AC (PAC) small-signal analysis computes transfer functions for circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, lower noise amplifier, sample-and-holds, and similar circuits. A PAC analysis cannot be used alone. It must follow a large signal PSS analysis. However, any number of periodic small-signal analyses can follow a signal PSS analysis.

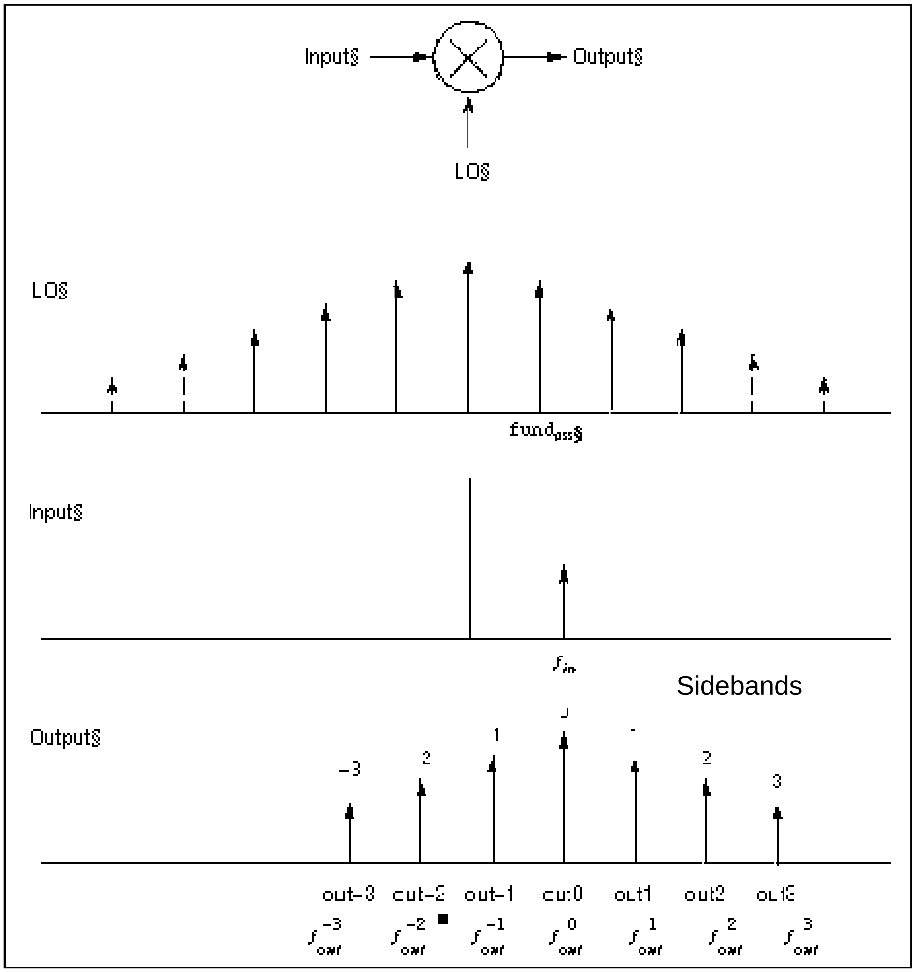
When you apply a small sinusoid to a linear time-invariant circuit, the steady-state response is a sinusoid at the same frequency. However, when you apply a small sinusoid to a linear circuit that is periodically time-varying, the circuit responds with sinusoids at many frequencies, as is shown in Figure 1-2 on page 45.

Because PAC is a small-signal analysis, the magnitude and phase of each tone computed by PAC is linearly related to the magnitude and phase of the input signal. PAC computes a series of transfer functions, one for each frequency. These transfer functions are unique because the input and output frequencies are offset by the harmonics of the LO. The SpectreRF simulation labels the transfer functions with the offsets from the input signal in multiples of the LO fundamental frequency. These same labels identify the corresponding sidebands of the output signals. The labels are used as follows:

- For circuits in which the input and output are at the same frequency, such as switched-capacitor filters, the transfer function or sideband is labeled 0.
- For down-conversion mixers, the transfer function or sideband is labeled -1 because the output frequency is offset from the input frequency by -1 times the LO frequency.
- For up-conversion mixers, the transfer function or sideband is labeled +1.
- For samplers, sidebands far from zero might be used.

In Figure 1-2 on page 45, all transfer functions from -3 to +3 are computed. As shown in the figure, the input signal is replicated and translated by each harmonic of the LO. In down-conversion mixers, the -1 sideband usually represents the IF output.

Figure 1-2 The Small-Signal Response of a Mixer as Computed by PAC Analysis.



PAC performance is not reduced if the input and LO frequencies are close or equal.

PAC Synopsis

You select the periodic small-signal output frequencies you want by specifying either the maximum sideband or an array of sidebands.

For a set of *n* integer numbers representing the sidebands k_1, k_2, ... k_n, the output frequency at each sideband is computed as $f^k_{out} = f_{in} + k \times fund_{pss}$

where

- f_{in} is the (possibly swept) input frequency.

- *$fund_{pss}$* is the fundamental frequency used in the corresponding PSS analysis.

If you specify the maximum sideband value as k_{max}, all $2 \times k_{max} + 1$ sidebands from $-k_{max}$ to $+k_{max}$ are generated.

Intermodulation Distortion Computation

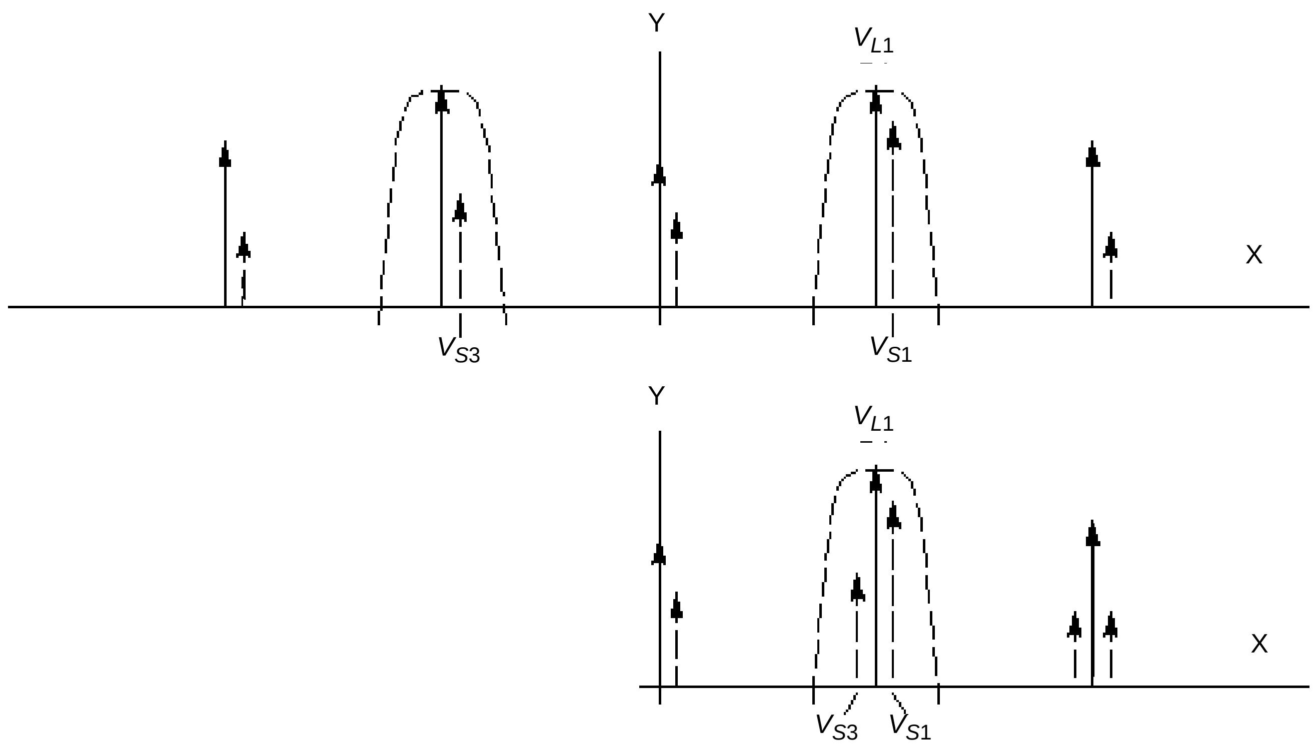
The PAC analysis can measure the intermodulation distortion of amplifiers and mixers. You can also measure intermodulation distortion with a QPSS analysis by applying two large, same-amplitude, closely spaced tones to the input and then measuring the third-order intermodulation products. The PSS/PAC approach is slightly different. You apply only one large tone in the PSS analysis. The PSS analysis is therefore faster than the QPSS analysis. Once the PSS analysis computes the circuit response to one large tone, then the PAC analysis applies the second tone close to the first. If you consider the small input signal to be one sideband of the large input signal, then the response at the other sideband is the third-order intermodulation distortion, as shown in Figure 1-3 on page 47.

In Figure 1-3, V_{L1} is the fundamental of the response due to the large input tone. V_{S1} is the fundamental of the response due to the small input tone and is the upper sideband of V_{L1}. V_{S3} is the lower sideband of V_{L1} (in this case, it is the -2 sideband of the response due to the small tone). V_{S3} represents the intermodulation distortion.

In the lower part of Figure 1-3, all of the signals are mapped into positive frequencies, which is the most common way of viewing such results.

Figure 1-3 Intermodulation Distortion Measured with PAC Analysis

.



Intermodulation distortion is efficiently measured by applying one large tone (L1), performing a PSS analysis, and then applying the second small tone (S1) with a PAC analysis. In this case, the first tone drives the circuit hard enough to cause distortion and the second tone is used to measure only the intermodulation distortion. Once V_{L1}, V_{S1}, and V_{S3} are measured in dB at the output, the output third-order intercept point is computed using the following equation.

$$IP_3 = V_{L1} - \frac{V_{S3} + V_{S1}}{2}$$

In the equation, V_{L1}, V_{S1}, and V_{S3} must be given in some form of decibels. Currently, dBV is used in the Analog Circuit Design Environment. In this example, V_{L1}, V_{S1}, and V_{S3} are given in dBV. Consequently, the intercept point is also computed in dBV. If V_{L1}, V_{S1}, and V_{S3} are given in dBm, the resulting intercept point is computed in dBm.

The intermodulation distortion of a mixer is measured in a similar manner, except that the PSS analysis must include both the LO and one large tone. For an example of measuring the intermodulation distortion of a mixer, see Chapter 4, “Simulating Mixers with SpectreRF simulation.”

If you must measure intermodulation distortion using two large tones to compare against bench measurements, you can run a PSS analysis with two large tones. However, a small commensurate frequency of the two tones can slow down simulation. At other times, use the quicker PSS/PAC approach.

Frequency Sweep

You can specify sweep limits by providing either the end points or the center value and the span of the sweep.

Steps can be linear or logarithmic and you can specify either the number of steps or the size of each step. You can specify a step size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step size parameter, the sweep is linear when the ratio of stop to start values is less than 10:1, and logarithmic when this ratio is equal to or greater than 10:1.

Alternatively, you may specify particular values for the sweep parameter using the values parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

Parameters for PAC Analysis

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.

sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.
relharmnum=1	Harmonic to which relative frequency sweep should be referenced.

Output Parameters

sidebands=[...]	Array of relevant sidebands for the analysis.
maxsideband	An alternative to the ‘sidebands’ array specification, which automatically generates the array: [-maxsideband ... 0 ... +maxsideband]
sidebands=[...]	Array of relevant sidebands for the analysis.
freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the output frequency. Default is ‘in’ for logarithmic frequency sweeps and ‘absout’ otherwise. Possible values are absout, out, or in.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.
outputperiod=0.0 (no output)	Time-domain output period. The time-domain small-signal response is computed for the period specified, rounded to the nearest integer multiple of the PSS period.

Convergence Parameters

tolerance=1e^{-9}	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration.
solver=turbo	Solver type. Possible values are std or turbo.
oscsolver=turbo	Oscillator solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Periodic S-Parameter Analysis

The Periodic S-Parameter (PSP) analysis is used to compute scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers and other similar circuits.

PSP is a small-signal analysis similar to the conventional SP analysis, except, the circuit is first linearized about a periodically time-varying operating point as opposed to a simple DC operating point. Linearizing about a periodically time-varying operating point allows the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

PSP analysis also calculates noise parameters in frequency-converting circuits. PSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. The noise features of PSP analysis, as for Pnoise analysis but unlike SP analysis, include noise folding effects due to the periodically time-varying nature of the circuit.

Computing the n-port S-parameters and noise parameters of a periodically varying circuit is a two step process.

- First, the small stimulus is ignored and the periodic steady-state response of the circuit to possibly large periodic stimulus is computed using PSS analysis.

  As a normal part of the PSS analysis, the periodically time-varying representation of the circuit is computed and saved for later use.

- Then, using the PSP analysis, small-signal excitations are applied to compute the n-port S-parameters and noise parameters.

  A PSP analysis cannot be used alone, it must follow a PSS analysis. However, any number of periodic small-signal analyses such as PAC, PSP, PXF, and Pnoise, can follow a single PSS analysis.

Like other SpectreRF small-signal analyses, the PSP analysis can sweep only frequency.

PSP Synopsis

For a PSP analysis, you need to specify the port and port harmonic relations. Select the ports of interest by setting the *port* parameter. Set the periodic small-signal output frequencies of interest by setting the *portharmsvec* or the *harmsvec* parameters.

For a given set of *n* integer numbers representing the harmonics K_1, K_2, ... K_n, the scattering parameters at each port are computed at the frequencies

$f(scattered) = f(rel) + K_i \times fund(pss)$

where *f(rel)* represents the relative frequency of a signal incident on a port, *f(scattered)* represents the frequency to which the relevant scattering parameter represents the conversion, and *fund(pss)* represents the fundamental frequency used in the corresponding PSS analysis.

Thus, when analyzing a down-converting mixer, with signal in the upper sideband, and sweeping the RF input frequency, the most relevant harmonic for RF input is $K_i = 1$ and for IF output $K_i = 0$. Hence we can associate $K_2=0$ with the IF port and $K_1=1$ with the RF port. *S21* will represent the transmission of signal from the RF to IF, and *S11* the reflection of signal back to the RF port. If the signal was in the lower sideband, then a choice of $K_1=-1$ would be more appropriate.

You can use either the *portharmsvec* or the *harmsvec* parameters to specify the harmonics of interest. If you give *portharmsvec*, the harmonics must be in one-to-one correspondence with the ports, with each harmonic associated with a single port. If you specify harmonics with the optional *harmsvec* parameter, then all possible frequency-translating scattering parameters associated with the specified harmonics are computed.

For PSP analysis, the frequencies of the input and of the response are usually different (this is an important way in which PSP differs from SP). Because the PSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the *freqaxis* and *sweeptype* parameters behave somewhat differently in PSP than they do in PAC and PXF.

The *sweeptype* parameter controls the way the frequencies are swept in PSP analysis. Specifying *relative* sweep, sweeps relative to the port harmonics (not the PSS fundamental). Specifying *absolute* sweep, sweeps the absolute input source frequency. For example, with a PSS fundamental of 100MHz, the *portharmsvec* set to [9 1] to examine a down-converting mixer, *sweeptype=relative*, and a sweep *range* of *f(rel)=0->50MHz*, then S21 would represent the strength of signal transmitted from the input port in the range 900->950MHz to the output port at frequencies 100->150MHz.

When `sweeptype=absolute`, the frequency range is for the first port only. You should have a frequency range for each port but this is not required since spectreRF knows the port harmonics of interest at each port, the absolute frequency range for each port can be calculated from the absolute frequency range of the first port. For example, when `sweeptype=absolute`, f (abs) = 900 -> 950 MHz, fund(pss) = 100 MHz, and `portharmsvec` = [9 1], then f (abs) for the first port is 900 -> 950 MHz, and f (abs) for the second port is 100 – 150 MHz.

The *freqaxis* parameter is used to specify whether the results should be output versus the scattered frequency at the input *port(in)*, the scattered frequency at the output *port(out)*, or the absolute value of the frequency swept at the input *port(absin)*.

Note: Unlike in PAC, PXF, and Pnoise analyses, increasing the number of requested ports and harmonics *increases* the simulation time linearly.

To insure accurate results in PSP analysis, you should set the *maxacfreq* parameter for the corresponding PSS analysis to guarantee that |*max{f(scattered)}*| is less than the *maxacfreq* parameter value, otherwise the computed solution might be contaminated by aliasing effects.

PSP analysis also computes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is skipped only when the *donoise* parameter is set to no, requires additional simulation time.

Noise Calculations Performed by PSP

Name	Description	Output Label
N_o	Total output noise at frequency f	
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise	**In**
G	Gain of the circuit (See **Note:**)	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
F_{dsb}	Double sideband noise factor	F_{dsb}
NF_{dsb}	Double sideband noise figure	NF_{dsb}
F_{ieee}	IEEE single sideband noise factor	F_{ieee}
NF_{ieee}	IEEE single sideband noise figure	NF_{ieee}

Note: The gain computed by PSP is the voltage gain from the actual circuit input to the circuit output, not the gain from the internal port voltage source to the output.

PSP analysis performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}^2}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

To insure accurate noise calculations, you need to set the *maxsideband* or *sidebands* parameters to include the relevant noise folding effects. The *maxsideband* parameter is only relevant to the noise computation features of PSP.

Parameters for PSP Analysis

Unlike other analyses in Spectre, the PSP analysis can only sweep frequency.

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.
sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.

Port Parameters

ports=[...]	List of active ports. Ports are numbered in the order given. For purposes of noise figure computation, the input is considered port 1 and the output is considered port 2.
portharmsvec=[...]	List of harmonics active on the specified list of ports. The `portharmsvec` vector must have a one-to-one correspondence with the `ports` vector.
harmsvec=[...]	Secondary list of additional, active harmonic's combinations. Harmonics included in the `harmsvec` vector are in addition to those associated with specific ports by `portharmsvec`.

Output Parameter

freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the input frequency. Default is `in`. Possible values are `absin`, `in` or `out`.

Noise Parameter

donoise=yes	Perform noise analysis. If *oprobe* is specified as a valid port, `donoise` is set to *`yes`*, and a detailed noise output is generated. Possible values are `no` or `yes`.

Probe Parameter

maxsideband=7	Maximum sideband included when computing noise either up-converted or down-converted to the output by the periodic drive signal.

Convergence Parameters

tolerance=1e^{-9}	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration, 1 or 2.
solver=turbo	Solver type. Possible values are std or turbo.
oscsolver=turbo	Oscillator solver type. Possible values are std or turbo.

Annotation Parameters

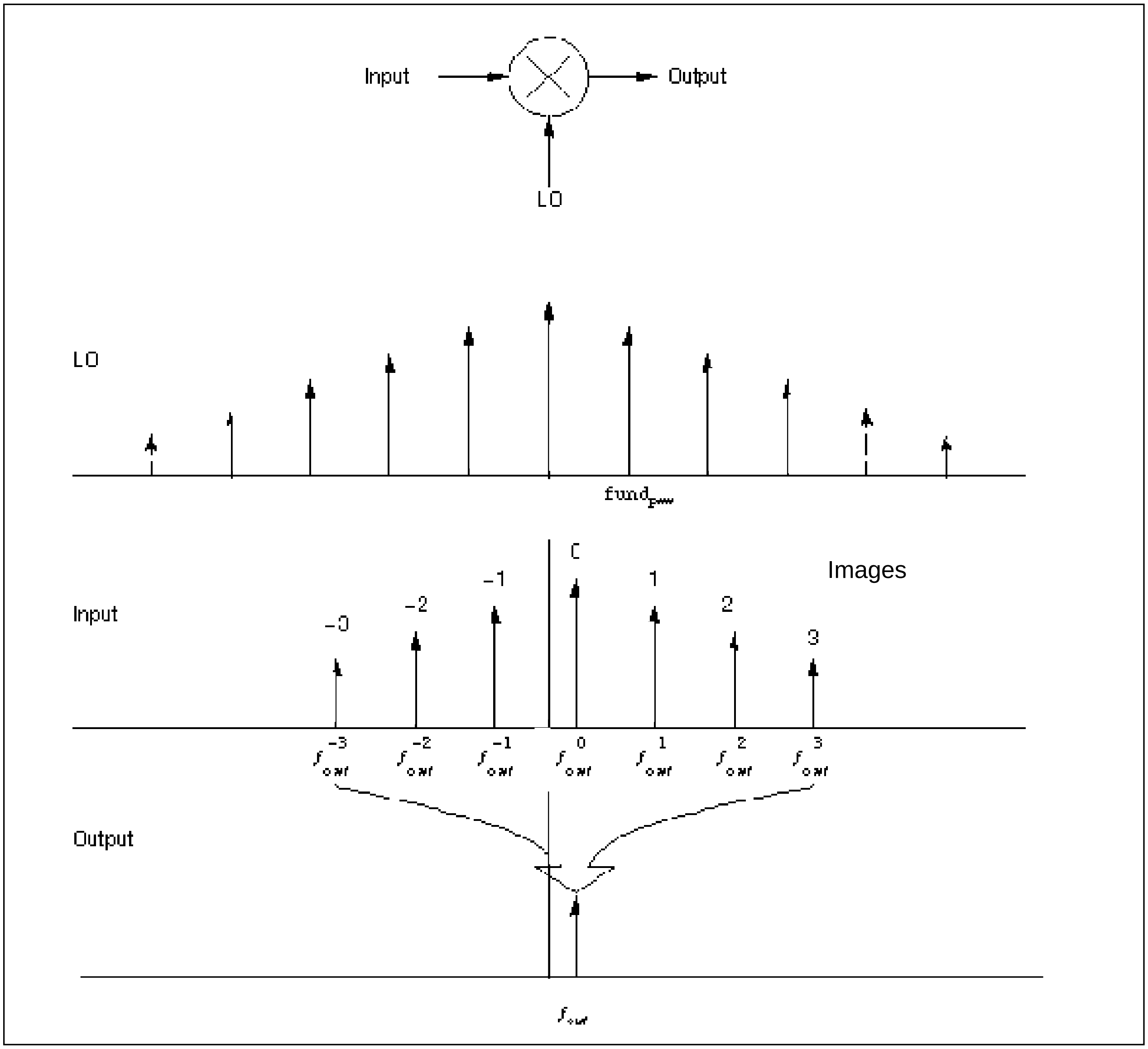
annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Periodic Transfer Function Analysis

Whereas a conventional AC analysis computes the response from a single stimulus to every node in the circuit, a conventional transfer function (XF) analysis computes the transfer function from every source in the circuit to a single output. The difference between PAC and PXF analysis is similar. The PXF analysis computes the transfer functions from any source at any frequency to a single output at a single frequency. Like PAC analysis, PXF analysis models frequency conversion effects. This is illustrated in Figure 1-4. PXF computes such useful quantities as conversion efficiency, image and sideband rejection, and LO feed through and power supply rejection.

The output is sensitive to signals at many frequencies at the input of the mixer. The input signals are replicated and translated by each harmonic of the LO. The signals shown in Figure 1-4 are those that end up at the output frequency.

Figure 1-4 Mixer Output Signals Shown by PXF Analysis



A PXF analysis can measure conversion gains, especially those from the input source to the output. A PXF analysis computes the conversion gain of the specified sideband as well as various unwanted images, including the baseband feed through. It also computes the coupling from other inputs such as the LO and the power supplies. These computations model frequency translation, and you can determine the sensitivity of the output to either up-converted or down-converted noise from the power supplies or the LO.

A PXF analysis, like the PAC analysis, must follow a PSS analysis.

PXF Synopsis

The output variable you measure can be voltage or current, and the variable frequency is not limited by the period of the large-periodic solution. When you sweep a selected output frequency, you can select the periodic small-signal input frequencies by specifying either the maximum sideband or an array of sidebands.

For a set of *n* integer numbers representing the sidebands k_1, k_2, …k_n, the input signal frequency at each sideband is computed as $f^{k}_{in} = f_{out} + k \times fund_{pss}$

where

- f_{out} represents the (possibly swept) output signal frequency
- $fund_{pss}$ represents the fundamental frequency used in the corresponding PSS analysis

When you analyze a down-converting mixer and sweep the IF output frequency, *k*= +1 for the RF input represents the first upper sideband, and *k* = -1 for the RF input represents the first lower sideband. If you specify k_{max} as the maximum sideband value, you select all $2 \times k_{max} + 1$ sidebands from $-k_{max}$ to $+k_{max}$. The number of sidebands you request does not affect accuracy, so request only sidebands that you want to see.

You can specify the output with a pair of nodes or a probe component. Any component with two or more terminals can be a voltage probe. Any component that naturally computes current as an internal variable, such as a voltage source, can be a current probe. In the Analog Circuit Design Environment, you set probes on the Choosing Analysis form when you specify input or output sources, and also when you specify positive or negative output nodes.

The *stimuli* parameter on the PXF Options form specifies the inputs for the transfer functions. You select from the following choices:

- *stimuli = sources* specifies that the sources present in the circuit are used. The `xfmag` component parameter provided by the sources can adjust the computed gain to compensate for gains or losses in a test fixture.
- *stimuli = nodes_and_terminals* specifies that all possible transfer functions are computed. Use this option when you cannot anticipate which transfer functions you might need to examine.

Transfer functions for nodes are computed assuming that a unit magnitude flow (current) source is connected from the node to ground. Transfer functions for terminals are computed assuming that a unit magnitude value (voltage) source is connected in series with the terminal. By default, the PXF analysis computes the transfer functions from a small set of terminals.

Parameters for PXF Analysis

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.
sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.
relharmnum=1	Harmonic to which relative frequency sweep should be referenced.

Probe Parameter

probe	Compute every transfer function to this probe component.

Output Parameters

stimuli=sources	Stimuli used for PXF analysis. Possible values are sources or nodes_and_terminals.
maxsideband	An alternative to the ‘sidebands’ array specification, which automatically generates the array: [-maxsideband... 0... +maxsideband]
sidebands=[...]	Array of relevant sidebands for the analysis.
freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the input frequency. Default is ‘out’ for logarithmic frequency sweeps and ‘absin’ otherwise. Possible values are absin, in or out.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.

nestlvl	Levels of subcircuits to output.

Convergence Parameters

tolerance=1e^{-9}	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration.
solver=turbo	Solver type. Possible values are std or turbo.
oscsolver=turbo	Oscillator solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Periodic Noise Analysis

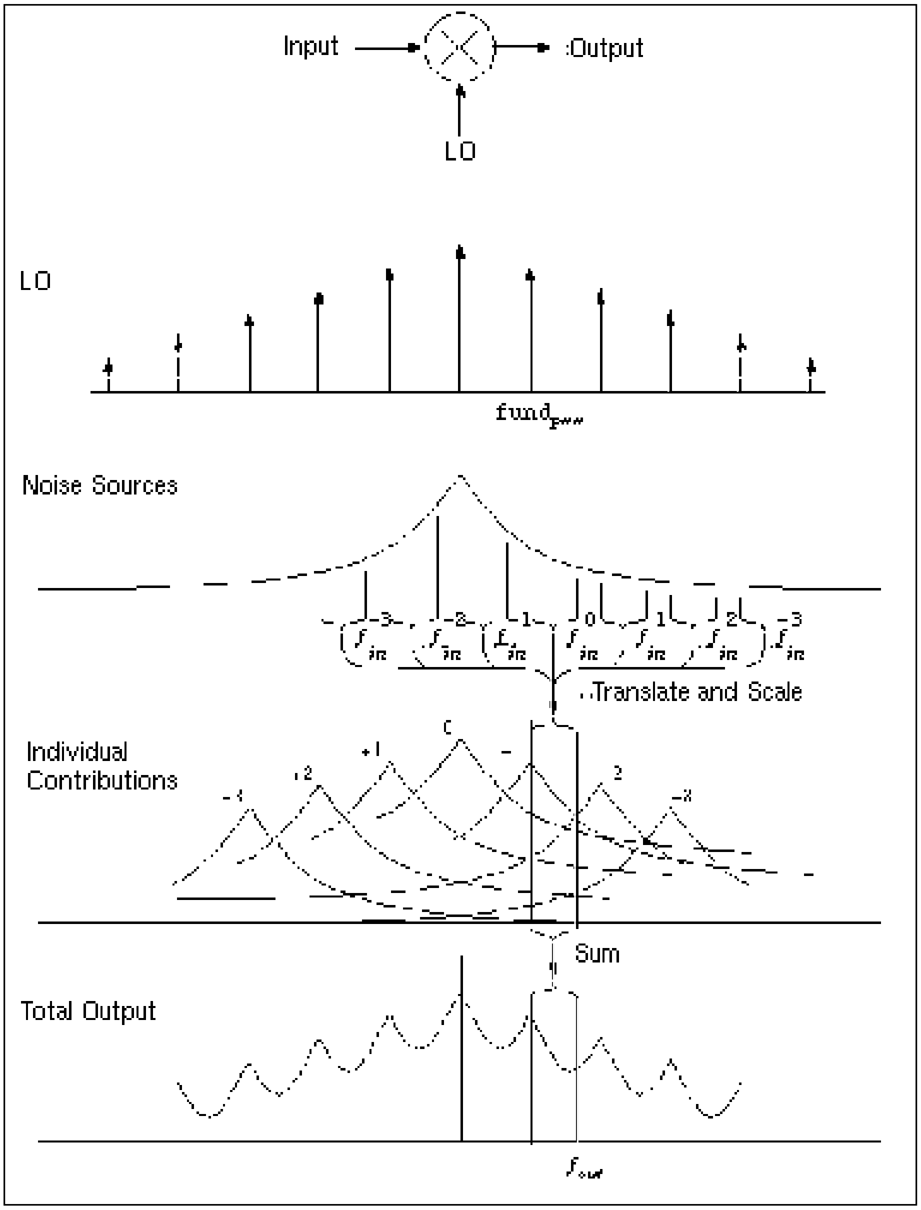
The Periodic Noise analysis (Pnoise) is similar to the conventional noise analysis except that it models frequency conversion effects. It can compute the phase noise of oscillators as well as the noise behavior of mixers, switched-capacitor filters, chopper-stabilized amplifiers, and other similar circuits.

First, the circuit is linearized using the PSS operating point. The periodically time-varying qualities of the linearized circuit create the frequency conversion. Then the Pnoise analysis simulates the effect of a periodically time-varying bias point on component-generated noise.

Initially, PSS computes the response to a large periodic signal such as a clock or a LO. These results are labeled LO and shown in Figure 1-5. The subsequent Pnoise analysis computes the resulting noise performance.

In periodic systems, there are two effects that act to translate noise in frequency. First, for noise sources that are bias dependent, such as shot noise sources, the time-varying operating point modulates the noise sources. Second, the transfer function from the noise source to the output is also periodically time-varying and modulates the noise source contribution to the output.

Figure 1-5 How noise is moved around by a mixer



The time-average of the noise at the output is computed as a spectral density versus frequency. You identify the output by specifying a probe component or a pair of nodes. To specify the output with a probe, the preferred approach, use the *oprobe* parameter. If the output is voltage (or potential), choose a *resistor* or a *port* component for the output probe. If the output is current (or flow), choose a *vsource* or *iprobe* component for the output probe.

The reference sideband (*refsideband*) specifies which conversion gain is used to compute the input-referred noise, the noise factor, and the noise figure. The reference sideband specifies the input frequency relative to the output frequency with

$$|f(input)| = |f(output) + refsideband \times fund(pss)|$$

Use *refsideband=0* when the input and output of the circuit are at the same frequency, such as with amplifiers and filters. When *refsideband* does not equal 0, the single sideband noise figure is computed.

The pnoise analysis computes the total noise at the output, which includes contributions from the input source, the circuit itself and the output load. The amount of the output noise that is attributable to each noise source in the circuit is also computed and output individually. If the input source is identified (using `iprobe`) and is a `vsource` or `isource`, the input-referred noise is computed, which includes the noise from the input source itself. Finally, if the input source is identified (using `iprobe`) and is also noisy, as is the case with ports, the noise factor and noise figure are computed.

Noise Calculations Performed by Pnoise

Name	Description	Output Label
N_o	Total output noise	**Out**
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise	**In**
G	Gain of the circuit	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
F_{dsb}	Double sideband noise factor	F_{dsb}
NF_{dsb}	Double sideband noise figure	NF_{dsb}
F_{ieee}	IEEE single sideband noise factor	F_{ieee}
NF_{ieee}	IEEE single sideband noise figure	NF_{ieee}

SpectreRF performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}^2}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

Pnoise Synopsis

Noise can mix with each harmonic of the periodic drive signal from the PSS analysis and appear at the output frequency. However, Pnoise analysis models only noise that mixes with a set of harmonics that you normally specify with the *maxsideband* parameter, but which you might specify with the *sidebands* parameter in special circumstances. If K_i represents sideband *i*, then

$$f(\mathrm{NoiseSource}) = f(\mathrm{out}) + K_i\, fund(\mathrm{pss})$$

The *maxsideband* parameter specifies the maximum $|K_i|$ included in the Pnoise calculation. Therefore, Pnoise ignores noise at frequencies less than $f(\mathrm{out}) -$ *maxsideband fund*(pss) and greater than $f(\mathrm{out}) +$ *maxsideband fund*(pss). If you specify sidebands with the *sidebands* parameter, then Pnoise includes only the specified sidebands in the calculation. When you specify *sidebands* parameter values, be careful not to omit any sidebands that might contribute significant output noise.

In the Analog Circuit Design Environment, you specify the *maxsideband* parameter with the *Maximum sideband* specification in the Choosing Analyses form. You specify the *sidebands* parameter in the *Array of Sidebands* field.

Increasing the number of harmonics computed by a Pnoise analysis improves simulation accuracy because it increases the number of harmonics used to compute the effects of noise folding. Noise folding is noise mixing with the harmonics into the frequency range of interest.

Parameters for Pnoise Analysis

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.

sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.
relharmnum=1	Harmonic to which relative frequency sweep should be referenced.

Probe Parameters

oprobe	Compute total noise at the output defined by this component.
iprobe	Refer the output noise to this component.
refsideband	Conversion gain associated with this sideband is used when computing input-referred noise or noise figure.

Output Parameters

noisetype=sources	Specifies whether the pnoise analysis should output cross-power densities or noise source information. Possible values are sources, correlations, or timedomain.
maxsideband=7	Maximum sideband included when computing noise either up-converted or down-converted to the output by the periodic drive signal.
sidebands=[...]	Array of relevant sidebands for the analysis.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.
maxcycles=0	Maximum cycle correlation frequency included when computing noise either up-converted or down-converted to the output by the periodic drive signal.
cycles=[...]	Array of relevant cycle frequencies. Valid only if noisetype=correlations.
numberofpoints=5	Number of time points of interest in the period where time-domain PSD is calculated. The simulator divides the period evenly into `numberofpoints` segments and calculates time-domain PSD on the starting time point of each segment. When `numberofpoints` is less than zero, the `numberofpoints` parameter is ignored.

noiseskipcount=-1	Calculates time-domain noise for one of every noiseskipcount time points. When noiseskipcount is greater than or equal to zero, the simulator uses the noiseskipcount parameter and ignores numberofpoints. When noiseskipcount is less than zero, the simulator ignores the noiseskipcount parameter.
noisetimepoints=[...]	Additional time points used when computing time-domain noise.
saveallsidebands=no	Save noise contributors by sideband. Possible values are no or yes.

Convergence Parameters

tolerance=1e^{-9}	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration.
solver=turbo	Solver type. Possible values are std or turbo.
oscsolver=turbo	Oscillator solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Noise Figure

When you use `pnoise` analysis to compute the noise factor or noise figure of a circuit, and the load generates noise, specify the output with the `oprobe` parameter rather than using a pair of nodes. Using the `oprobe` parameter explicitly specifies the load as the output probe. This is preferable because it excludes the noise of the load from the calculation of the noise figure.

As an alternative, you can specify the output with a pair of nodes and make the load component a noiseless resistor. Results with this approach are similar to those computed if you specify a resistor or a port as the output probe (load). The only difference is that the noiseless resistor is considered noiseless for other noise calculations, such as total output noise and input-referred noise, and the resistor is noiseless at all frequencies. When you specify a conventional resistor or port for the load, its noise is subtracted from only the noise factor and noise figure calculations, and only at the output frequency. Consequently, noise from load frequencies other than the output frequency can appear at the output frequency if the circuit has a nonlinear output impedance.

Pnoise computes the single-sideband noise figure. To match the IEEE definition of noise figure, you must use a *port* as the input probe and a *resistor* or a *port* as the output probe. In addition, the input port noise temperature must be 290 K (`noisetemp` = 16.85) and have no excess noise. (You must not specify `noisevec` and `noisefile` on the input port.) The 290K temperature is the average noise temperature of an antenna used for terrestrial communication. However for your application, you can specify the input port noise temperature to be any appropriate value. For example, the noise temperature for antennas pointed at satellites is usually much lower.

Flicker Noise

To avoid inaccurate results with `pnoise` analysis on a circuit that mixes *flicker noise* or *1/f* noise up to the carrier or its harmonics, place a cluster of frequencies near each harmonic to resolve the noise peaks accurately, but do not put frequency points precisely on the harmonics. In addition, choose `pnoise` start and stop frequencies to avoid placing points precisely on the harmonics of the periodic drive signal. Then use the `values` parameter to specify a vector of additional frequency points near the harmonics. In the Analog Circuit Design Environment, the `values` parameter is set in the *Add Specific Points* field in the Choosing Analyses form.

The effect of specifying appropriate additional frequency points is shown in the following three diagrams. Figure 1-6 shows the true output noise of a mixer with flicker noise.

Figure 1-6 Actual Mixer Noise Output Including Flicker Noise

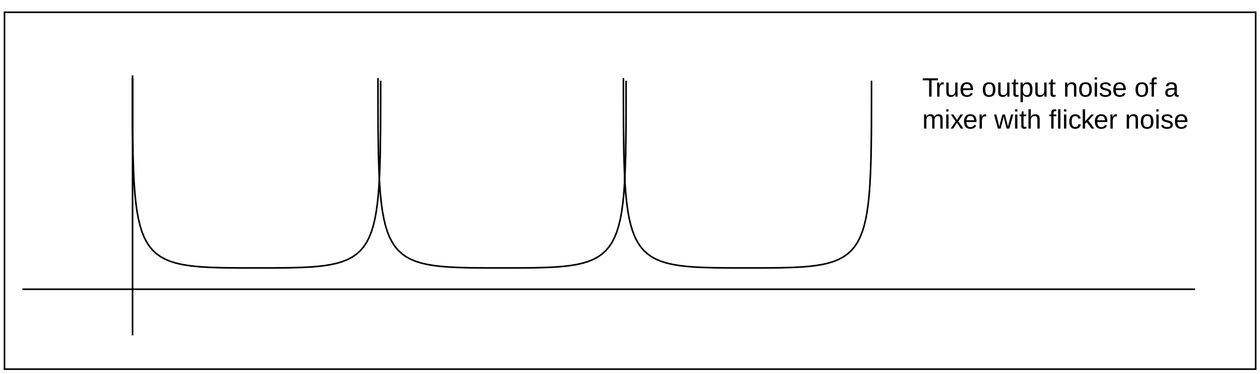


Figure 1-7 shows the output noise computed with a typical choice of points. In this case, total output noise is typically exaggerated by many orders of magnitude.

Figure 1-7 Noise Output Computed With Typical Points

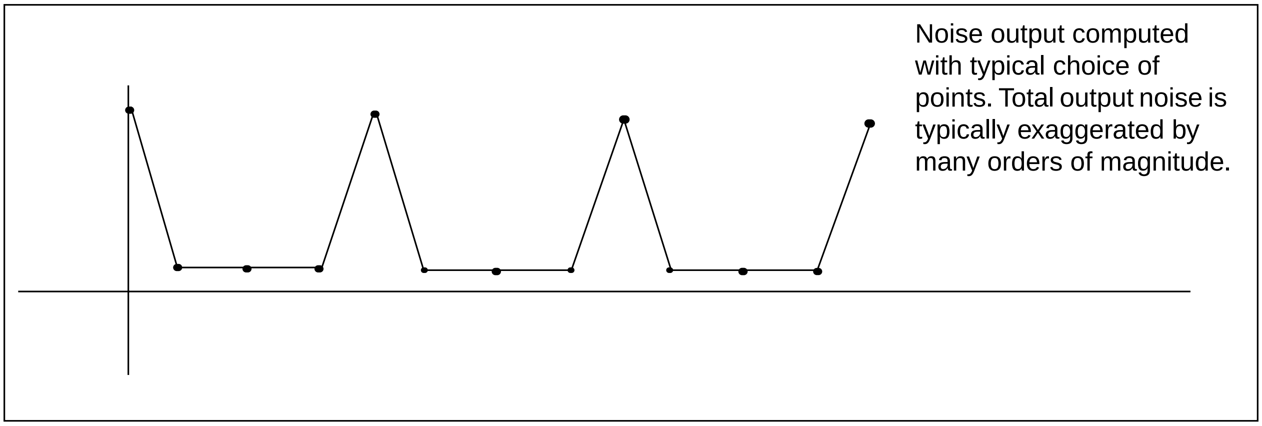
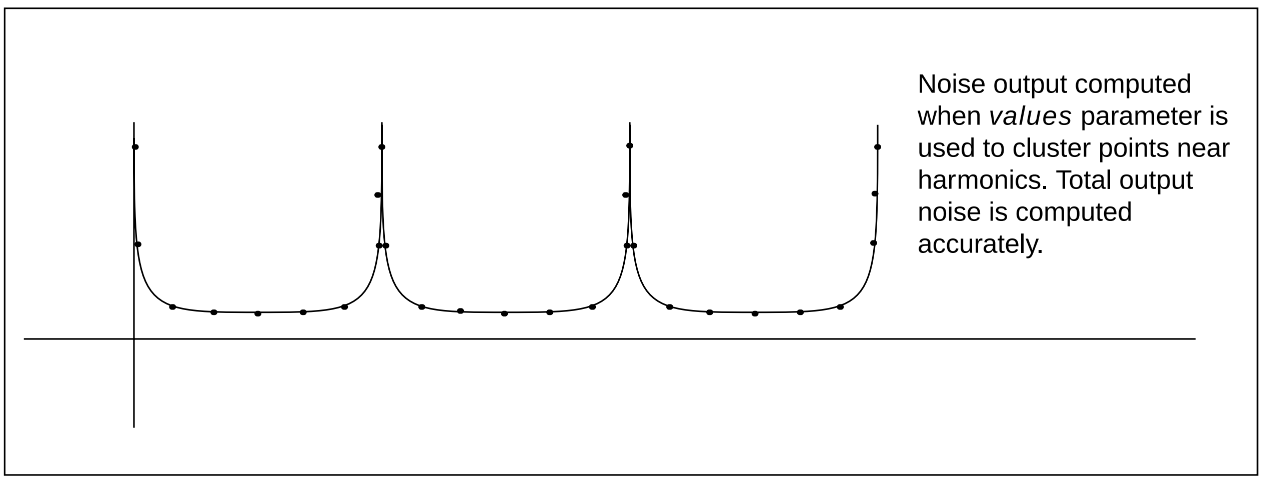


Figure 1-8 shows noise output computed when the values parameter is used to cluster points near harmonics. By comparing Figures 1-7 and 1-8, you can see that total output noise is computed accurately when points are carefully chosen.

Figure 1-8 Noise Output Computed With Clustered Points



Flicker Noise Spectrum

Flicker noise depends on the current, `I`, of the channel. For all devices, flicker, or `1/f`, noise depends on the following equation

$$\text{Flicker noise current source} = sign(\mathtt{I})|\mathtt{I}|^{A_f}\left(\frac{K_f}{f}\right)$$

This means that you will see only odd harmonics. The power looks like a rectified sine wave because

$$\text{Power} \sim \mathtt{I}^2$$

Quasi-Periodic Steady-State Analysis

The quasi-periodic steady-state (QPSS) analysis computes the quasi-periodic steady-state response of a circuit that operates on multiple time scales. A quasi-periodic signal has multiple fundamental frequencies. Closely spaced or incommensurate fundamentals cannot be efficiently resolved by PSS. QPSS allows you to compute responses to several moderately large input signals in addition to a strongly nonlinear tone which represents the LO or clock signal. The circuit is assumed to respond in a strongly nonlinear fashion to the large tone and in a weakly nonlinear fashion to the moderate tones. You might use a QPSS analysis, for example, to model intermodulation distortion with two moderate input signals. QPSS treats one of the input signals (usually the one that causes the most nonlinearity or the largest response) as the large signal, and the others as moderate signals.

When you perform a QPSS analysis

1. An initial transient analysis runs with all moderate input signals suppressed.
2. A number of stabilizing iterations run (always at least 2) with all signals activated.
3. The shooting Newton method runs.

The QPSS analysis employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. For details about the MFT algorithm, see *Steady-State Methods for Simulating Analog and Microwave Circuits*, by K. S. Kundert, J. K. White, and A. Sangiovanni-Vincentelli, Kluwer, Boston, 1990.

As is true for PSS analysis, QPSS analysis uses the shooting Newton method as its backbone. However, unlike PSS analysis (where each Newton iteration performs a single transient integration), for each Newton iteration the QPSS analysis performs a number of transient integrations of one large signal period. Each integration differs by a phase-shift in each moderate input signal.

The large tone is resolved in the time domain and the moderate tones are resolved in the frequency domain (hence the name mixed frequency time algorithm). When you set up a QPSS analysis, you determine the number of integrations performed by the number of harmonics of moderate fundamental you select.

You select the moderate signals to model with the `maxharms` parameter as follows

$$\text{maxharms} = [k_1, k_2, \ldots, k_n]$$

The total number of integrations for the simulation is calculated as

$$(2k_1 + 1) \times (2k_2 + 1) \times \ldots \times (2k_n + 1)$$

One consequence is that the efficiency of the algorithm depends significantly on the number of harmonics required to model the responses of moderate fundamentals. Another consequence is that the number of harmonics of the large fundamental does not significantly affect the efficiency of the shooting algorithm. The boundary conditions of a shooting interval are such that the time domain integrations are consistent with a frequency domain transformation with a shift of one large signal period.

Comparing QPSS Analysis with Using PSS and PAC Analyses

A QPSS analysis is similar to a PSS analysis followed by a PAC analysis in that if you treat one of the input signals as a small signal, a signal whose harmonics do not contribute significantly to the output, you can use the PSS/PAC analyses to model intermodulation distortion effectively.

The QPSS analysis has the following advantages that make it the analysis of choice in many situations. For example, with a QPSS analysis you can measure

- Harmonic distortion and frequency translation effects created by multiple moderate-signal inputs, including all third order products. These measurements are impossible to obtain using PSS/PAC analysis because they do not model the effects of small-signal harmonics.
- The effects of multiple moderate signals. With PSS/PAC analysis, you are restricted to modeling the effects of a single small signal on the fundamental you compute with the PSS analysis. A QPSS analysis lets you model all moderate-signal inputs, including the sums of sinusoids that are not periodic.

To see the difference between the information available with QPSS and PSS/PAC analyses, compare how the two approaches determine the output signals produced by input signals at 900 MHz and 905 MHz. The discussion below assumes you are interested in data only for fundamentals and their first harmonics.

If you perform a PSS analysis for the 900 MHz signal followed by a PAC analysis that applies 905 MHz as a small-signal with `maxsideband = k`$_1$, you get

$$i \times 900M + 905M$$

Where

$$i = \pm 1, \pm 2, \ldots, \pm k_1$$

If you perform a QPSS analysis with input signals at 900 MHz (for the large tone) and 905 MHz (for the moderate tone), you get information about the following additional frequency translation signals created by moderate tones.

The output signals are centered at the following frequencies:

$$i \times 900M + j \times 905M$$

where

$$i = \pm 1, \pm 2, \ldots, \pm k_1$$

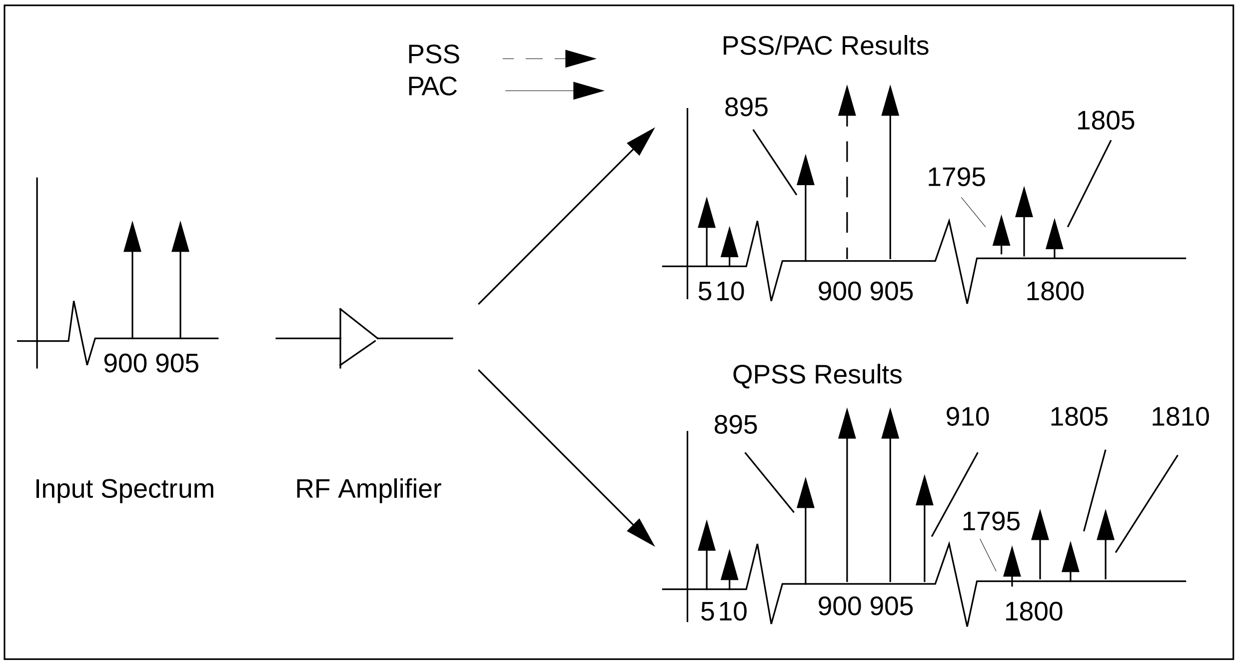
and

$$j = \pm 1, \pm 2, \ldots, \pm k_2$$

Again, the number of harmonics of the moderate tone (k_2) effects the simulation time.

Figure 1-9 shows that more information is available from a QPSS analysis than from a PSS analysis followed by a PAC analysis.

Figure 1-9 Comparison of Information From QPSS and PSS/PAC Analyses



QPSS Synopsis

The QPSS analysis inherits a majority of its parameters from the PSS analysis with a few new parameters added and a few parameters extended.

The most important parameters for QPSS analysis are the `funds` and `maxharms` parameters. In QPSS analysis, the `funds` and `maxharms` parameters replace and extend the PSS parameters, `fund` (or `period`) and `harms`, respectively.

- The `funds` parameter accepts a list of fundamental names that are present in the sources. (These fundamental names are specified by the source parameter `fundname`.) The simulator figures out the frequencies associated with the fundamental names.

  An important feature of the `funds` parameter is that each input signal can be composed of more than one source. However, these sources must all have the same fundamental name. For each fundamental name, its fundamental frequency is the greatest common factor of all frequencies associated with the name.

  If you do not list all the fundamental names on the `funds` parameter, the current simulation is terminated. However, if you do not specify `maxharms`, a warning message displays, and the number of harmonics defaults to 1 for each fundamental.

The first fundamental is considered as the large signal. You can use a few heuristics to pick the large fundamental.

- ❑ Pick the fundamental which is not sinusoidal.
- ❑ Pick the fundamental which causes the most nonlinearity.
- ❑ Pick the fundamental which causes the largest response.

- The `maxharms` parameter accepts a list of numbers of harmonics that are required to sufficiently model responses due to different fundamentals.

The role of some PSS parameters is extended for QPSS analysis.

- The `maxperiods` parameter that controls the maximum number of shooting iterations for PSS analysis also controls the maximum number of shooting iterations for QPSS analysis.
- The `tstab` parameter controls both the length of the initial transient integration, with only the clock tone activated, and the number of stabilizing iterations, with the moderate tones activated. The stable iterations are run before Newton iterations begin.

Distortion Fundamental Parameters

funds=[...]	Array of fundamental frequency names for fundamentals to use in analysis.
maxharms=[...]	Array of number of harmonics of each fundamental to consider for each fundamental.

Simulation interval parameters

tstab=0.0 s	Extra stabilization time after the onset of periodicity for independent sources.
tstart=0.0 s	Initial transient analysis start time.

Time-step parameters

maxstep (s)	Maximum time step. Default derived from `errpreset`.
step=0.001*period s	Minimum time step that would be used solely to maintain the aesthetics of the results.

Initial-condition parameters

ic=all	Used to set initial condition. Possible values are dc, node, dev, or all.

skipdc=no	If yes, there will be no dc analysis for transient. Possible values are no or yes.
readic	File that contains initial condition.

Convergence parameters

readns	File that contains estimate of initial transient solution.
cmin=0 F	Minimum capacitance from each node to ground.

Output parameters

save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.
oppoint=no	Should operating point information be computed for initial timestep, and if so, where should it be sent. Possible values are no, screen, logfile, or rawfile.
skipstart=starttime s	The time to start skipping output data.
skipstop=stoptime s	The time to stop skipping output data.
skipcount	Save only one of every skipcount points.
strobeperiod (s)	The output strobe interval (in seconds of transient time).
strobedelay=0 s	The delay (phase shift) between the skipstart time and the first strobe point.
compression=no	Do data compression on output. Possible values are no or yes.

State-file parameters

write	File to which initial transient solution (before steady-state) is to be written.
writefinal	File to which final transient solution in steady- state is to be written.
swapfile	Temporary file that holds steady-state information. Tells Spectre to use a regular file rather than virtual memory to hold the periodic operating point. Use this option if Spectre complains about not having enough memory to complete this analysis.

Integration method parameter

method	Integration method. Default derived from 'errpreset'. Possible values are euler, trap, traponly, gear2, or gear2only.

Accuracy parameters

errpreset=moderate	Selects a reasonable collection of parameter settings. Possible values are liberal, moderate or conservative.
relref	Reference used for the relative convergence criteria. Default derived from 'errpreset'. Possible values are pointlocal, alllocal, sigglobal, or allglobal.
lteratio	Ratio used to compute LTE tolerances from Newton tolerance. Default derived from 'errpreset'.
steadyratio	Ratio used to compute steady state tolerances from LTE tolerance. Default derived from 'errpreset'.
maxperiods	maxperiods=20 Maximum number of simulated periods to reach steady- state.
tolerance=1e-4	Relative tolerance for linear solver.
finitediff	Options for finite difference method refinement after quasi-periodic shooting method. 'finitediff' is changed from no to samegrid automatically when 'readqpss' and 'writeqpss' are used to re-use QPSS results. Possible values are no, yes or refine.

Annotation parameters

stats=no	Analysis statistics. Possible values are no or yes.
annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
title	Analysis title.

Newton parameters

maxiters=5	Maximum number of iterations per time step.
restart=yes	Do not use previous DC solution as initial guess. Possible values are no or yes

Circuit age

circuitage (Years)	Stress Time. Age of the circuit used to simulate hot- electron degradation of MOSFET and BSIM circuits.
writeqpss	File to which final quasi-periodic steady-state solution is to be written. Small signal analyses such as qpac, qpxf and qpnoise can read in the steady- state solution from this file directly instead of running the qpss analysis again.
readqpss	File from which final quasi-periodic steady-state solution is to be read. Small signal analyses such as qpac, qpxf and qpnoise can read in the steady-state solution from this file directly instead of running the qpss analysis again.

The errpreset Parameter in QPSS Analysis

The `errpreset` parameter quickly adjusts several simulator parameters to fit your needs. In most cases, `errpreset` should be the only parameter you need to adjust.

- For a fast simulation with reasonable accuracy, set `errpreset` to *`liberal`*.
- For greater accuracy, set `errpreset` to *`moderate`*.
- If accuracy is your primary concern, set `errpreset` to *`conservative`*.

If you do not specify the `steadyratio` parameter, it is always `1.0`, and it is not affected by your `errpreset` setting. Table 1-4 shows the effect of `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the default values of other parameters.

Table 1-4 Parameter Default Values for `errpreset` Settings

errpreset	reltol	relref	method	Iteratio	maxstep
liberal	1e-3	sigglobal	gear2only	3.5	clock period/80
moderate	1e-4	sigglobal	gear2only	3.5	clock period/100
conservative	1e-5	sigglobal	gear2only	10.0	clock period/200

These `errpreset` settings include a default `reltol` value which is an enforced upper limit for `reltol`. The only way to decrease the `reltol` value is in the `options` statement. The only way to *relax* the `reltol` value is to change the `errpreset` setting. Spectre sets the `maxstep` parameter value so that it is no larger than the value given in Table 1-4.

Except for the `reltol` and `maxstep` values, the `errpreset` setting does not change any parameter values you have explicitly set. The actual values used for the QPSS analysis are given in the log file.

Quasi-Periodic Noise Analysis

The Quasi-Periodic Noise (QPnoise) analysis is a quasi-periodic small-signal analysis similar to the conventional noise analysis, except that with QPnoise the circuit is first linearized about a quasi-periodically time-varying operating point as opposed to a simple DC operating point. Linearizing about a quasi-periodically time-varying operating point includes frequency conversion and intermodulation effects. Simply linearizing about a DC operating point cannot include frequency translation because linear time-invariant circuits do not exhibit frequency translation. QPnoise also includes the effect of a quasi-periodically time-varying bias point on the noise generated by the various components in the circuit. Hence QPnoise is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits. You cannot use a QPnoise analysis alone. It must follow a QPSS analysis. However, any number of quasi-periodic small signal analyses (QPAC, QPSP, QPXF, and QPnoise) can follow a single QPSS analysis.

Computing the small-signal response of a quasi-periodically varying circuit is a two step process.

1. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Then, the small stimuli representing both individual noise sources in the circuit as well as the input noise are applied to the periodically-varying linear representation to compute the small signal response. This is done using the QPnoise analysis.

QPnoise Output

The time-average of the noise at the output of the circuit is computed in the form of a spectral density versus frequency. The output of the circuit is specified with either a pair of nodes or a probe component. To specify the output of a circuit with a probe, specify it using the `oprobe` parameter. If the output is voltage (or potential), choose a *resistor* or a *port* as the output probe. If the output is current (or flow), choose a `vsource` or `iprobe` as the output probe.

If you want the input-referred noise, specify the input source using the `iprobe` parameter. Currently, only a `vsource`, an `isource`, or a `port` can be used as an input probe. If the input source is noisy, as is a `port`, the noise analysis will compute the noise factor (F) and noise figure (NF). To match the IEEE definition of noise figure, the input probe must be a `port` with no excess noise and you must set its `noisetemp` parameter to 16.85 C (290 K). In addition, the output load must be a `resistor` or `port` and must be identified as the `oprobe`.

Use the `refsideband` parameter to specify which conversion gain to use when computing input-referred noise, noise factor, and noise figure. The reference sideband satisfies:

$$|f(input)| = |f(out) + \text{refsideband frequency shift}|$$

Sidebands are vectors in QPnoise analysis. Assume you have one large tone and one moderate tone in a QPSS analysis. A sideband might be `[K1 K2]`. It corresponds to the following `refsideband` frequency shift.

$$K_1 \times \text{fund(qpss-large-tone)} + K_2 \times \text{fund(qpss-moderate-tone)}$$

Use `refsideband=[0 0...]` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters). When the `refsideband` parameter value differs from the 0 vector, QPnoise computes the single side-band noise figure.

The noise analysis always computes

- Total noise at the output. This includes contributions from the input source and the output load.
- The amount of output noise that is attributable to each noise source in the circuit. These are output individually.

If you identify the input source with `iprobe` and it is a `vsource` or an `isource`, the input-referred noise is computed. This includes the noise from the input source itself.

If you identify the input source with `iprobe` and it is noisy, as is the case with ports, the noise factor and noise figure are computed.

Noise Calculations Performed by QPnoise

Name	Description	Output Label
N_o	Total output noise	**Out**
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise (See **Note:**)	**In**

Noise Calculations Performed by QPnoise

G	Gain of the circuit (See **Note:**)	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
$\mathbf{F_{dsb}}$	Double sideband noise factor	$\mathbf{F_{dsb}}$
$\mathbf{NF_{dsb}}$	Double sideband noise figure	$\mathbf{NF_{dsb}}$
$\mathbf{F_{ieee}}$	IEEE single sideband noise factor	$\mathbf{F_{ieee}}$
$\mathbf{NF_{ieee}}$	IEEE single sideband noise figure	$\mathbf{NF_{ieee}}$

Note: For the QPnoise analysis, the computation of *gain* and *IRN* assumes that the circuit under test is impedance-matched to the input source. This can introduce inaccuracy into the *gain* and *IRN* computation.

SpectreRF performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

QPnoise Synopsis

At the UNIX command line use the optional terminals (p and n) to specify the output of the circuit. If you do not give the terminals, then you must specify the output with a `probe` component.

In practice, noise can mix with each of the harmonics and intermodulation products of the periodic drive signals applied in the QPSS analysis and end up at the output frequency. However, the QPnoise analysis includes only the noise that mixes with a finite set of harmonics. The noise contributions that are folded in are controlled by several parameters: `clockmaxharm`, `sidevec` and the `maxharms` list in QPSS.

The `clockmaxharm` parameter affects only the first large tone, which is typically the LO or a clock frequency. It limits the maximum harmonic order of the large tone that will be considered. The order of the harmonics to be considered for the moderate tones is limited by the corresponding entries in the `maxharms` list in QPSS. In the case of the two tones in QPSS, the harmonics of the first tone are limited by `clockmaxharm`. For the second tone, the maximum harmonic order is `maxharms`[2].

If you select sidebands using the `sidevec` parameter, then these selected sidebands are the only sidebands included in the calculation. Take care when specifying the QPnoise `sidevec` and `clockmaxharm` parameters and the QPSS `maxharms` parameter. Your noise results will be in error if you fail to include a sideband that contributes significant noise to the output.

The number of requested sidebands will substantially change the simulation time.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the sidevec parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by a vector K^1 as $[K^1_1 \; K^1_2]$. The corresponding frequency translation is

K^1_1 x f(qpss-large-tone) + K^1_2 x f(qpss-moderate-tone)

When there are L tones total in the QPSS analysis (1 large tone and L-1 moderate tones), there is also a given set of n integer vectors representing the sidebands

$$K^1 = \{ K^1_1, \; ..., \; K^1_j, \; ..., \; K^1_L\}$$
$$K^2 = \{ K^2_1, \; ..., \; K^2_j, \; ..., \; K^2_L\}$$
...
$$K^n = \{ K^n_1, \; ..., \; K^n_j, \; ..., \; K^n_L\}$$

The QPnoise analysis computes the output frequency corresponding to each sideband as follows.

$$f^i(out) = f(in) + \sum_{i=1}^{L} \left(K^i_j \cdot f_j\right)$$

where

- *f(in)* represents the (possibly swept) input frequency
- f_j represents the fundamental frequency used in the corresponding QPSS analysis.

Enter the sidevec parameter as a sequence of integer numbers separated by spaces. For example, you would enter the set of vectors {1 1 0} {1 -1 0} {1 1 1} as follows

```
sidevec=[ 1 1 0 1 -1 0 1 1 1]
```

The other way to specify the sidebands is the clockmaxharm parameter and the maxharms parameter in the preceding QPSS analysis. Only the large tone, the first fundamental in QPSS, is affected by the QPnoise clockmaxharm parameter value. It limits the maximum harmonic order of the large tone that will be considered. All the remaining tones, the moderate tones, are limited by the QPSS maxharms parameter value.

Given the following parameters

- In the QPSS analysis input, maxharms=[k^0_{max} k^2_{max}... k^n_{max}]
- In the QPnoise analysis input, clockmaxharm=K_{max}

The QPnoise analysis output generates the following number of sidebands

$$(2*K_{max} + 1)*(2*k^{2}_{max}+1)*(2*k^{3}_{max}+1)*...*(2*k^{n}_{max}+1)$$

Swept QPnoise Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic as well as the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10:1, and logarithmic when this ratio is equal to or greater than 10:1.

Alternatively, you can specify the particular values for the sweep parameter using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before simulation. All frequencies are in Hertz.

QPnoise Parameters

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.
sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.
relharmnum=[...]	Harmonic to which relative frequency sweep should be referenced.

Probe Parameters

oprobe	Compute total noise at the output defined by this component.
iprobe	Refer the output noise to this component.
refsideband=[...]	Conversion gain associated with this sideband is used when computing input-referred noise or noise figure.
refsidebandoption=individual	Whether to view the sideband as a specification of a frequency or a specification of an individual sideband. Possible values are freq or individual.

Output Parameters

clockmaxharm=7	Maximum clock harmonics range included when computing noise either up-converted or down-converted to the output by the periodic drive signal.
sidevec=[...]	Array of relevant sidebands for the analysis.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.
saveallsidebands=no	Save noise contributors by sideband. Possible values are no or yes.

Convergence Parameters

tolerance=1e-9	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration, 1 or 2.
solver=turbo	Solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Quasi-Periodic AC Analysis

The quasi-periodic AC (QPAC) analysis computes transfer functions for circuits that exhibit multitone frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and similar circuits.

QPAC is a quasi-periodic small-signal analysis like the conventional AC analysis, except that with QPAC the circuit is first linearized about a quasi-periodically time-varying operating point as opposed to a simple DC operating point. Linearizing about a quasi-periodically time-varying operating point produces transfer-functions that include frequency translation. Simply linearizing about a DC operating point cannot include frequency translation because linear time-invariant circuits do not exhibit frequency translation.

Computing the small-signal response of a quasi-periodically varying circuit is a two step process.

1. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Second, the small stimuli are applied to the periodically-varying linear representation to compute the small signal response. This is done using the QPAC analysis.

QPAC Output Frequency and Sideband Vectors

You select the set of quasi-periodic small-signal output frequencies you are interested in by setting either the `clockmaxharm` or the `sidevec` output parameters.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by the vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

$$K^1_1 \times f(\text{qpss-large-tone}) + K^1_2 \times f(\text{qpss-moderate-tone})$$

When there are `L` tones total in the QPSS analysis (`1` large tone and `L-1` moderate tones), there is also a given set of `n` integer vectors representing the sidebands

$$K^1 = \{ K^1_1, \ldots, K^1_j, \ldots, K^1_L\}$$
$$K^2 = \{ K^2_1, \ldots, K^2_j, \ldots, K^2_L\}$$
…
$$K^n = \{ K^n_1, \ldots, K^n_j, \ldots, K^n_L\}$$

The QPAC analysis computes the output frequency corresponding to each sideband as follows.

$$f^{i}(out) = f(in) + \sum_{i=1}^{L} \left(K_{j}^{i} \cdot f_{j}\right)$$

Where

- $f(in)$ represents the (possibly swept) input frequency
- f_j represents the fundamental frequency used in the corresponding QPSS analysis.

In modeling a down-converting mixer with *low-side* LO and with swept RF input frequency, the most relevant sideband for the IF output is `{-1,0}`. For an up-converting mixer with swept IF input frequency, the most relevant sideband for the RF output is `{1,0}.` In a typical IP3 measurement, IM1 will correspond to the {-1, 0} sideband and IM3 to the {-1, 2} sideband.

Enter the `sidevec` parameter as a sequence of integer numbers separated by spaces. For example, you would enter the set of vectors `{1 1 0} {1 -1 0} {1 1 1}` as follows

```
sidevec=[ 1 1 0 1 -1 0 1 1 1]
```

The other way to specify the sidebands is the `clockmaxharm` parameter and the `maxharms` parameter in the preceding QPSS analysis. Only the large tone, the first fundamental in QPSS, is affected by the QPAC `clockmaxharm` parameter value. It limits the maximum harmonic order of the large tone that will be considered. All the remaining tones, the moderate tones, are limited by the QPSS `maxharms` parameter value.

Given the following parameters

- In the QPSS analysis, maxharms=[k^0_{max} k^2_{max}... k^n_{max}]
- In the QPAC analysis, clockmaxharm=K_{max}

The QPAC analysis output generates the following number of sidebands

$(2*K_{max} + 1)*(2*k^2_{max}+1)*(2*k^3_{max}+1)*...*(2*k^n_{max}+1)$

Note: The number of sidebands you request substantially increases the simulation time.

For a QPAC analysis, the stimulus and response frequencies are usually different (this is an important way in which QPAC analysis differs from AC analysis).

Use the QPAC `freqaxis` parameter to specify how to output the QPAC simulation results. For

- `freqaxis=`*in* the results are output versus the input frequency
- `freqaxis=`*out* the results are output versus the output frequency
- `freqaxis=`*absout* the results are output versus the absolute value of the output frequency

Swept QPAC Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic as well as the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10, and logarithmic when this ratio is 10 or greater.

Alternatively, you can specify the particular values that the sweep parameter should take using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before simulation. All frequencies are in Hertz.

QPAC Parameters

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.

sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.
relharmvec=[...]	Harmonic to which relative frequency sweep should be referenced.

Output Parameters

sidevec=[...]	Array of relevant sidebands for the analysis.
clockmaxharm=0	An alternative to the ‘sidevec’ array specification, which automatically generates the array: [- clockmaxharm ... 0 ... +clockmaxharms] [- maxharms(QPSS)[2]...0...maxharms(QPSS)[2]][...].
freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the output frequency. Default is ‘in’ for logarithmic frequency sweeps and ‘absout’ otherwise. Possible values are absout, out or in.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.

Convergence Parameters

tolerance=1e-9	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration, 1 or 2.
solver=turbo	Solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Quasi-Periodic S-Parameter Analysis

The quasi-periodic SP (QPSP) analysis computes scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and the like.

QPSP is a quasi-periodic small-signal analysis similar to the SP analysis except that with QPSP the circuit is first linearized about a quasi-periodically time-varying operating point as opposed to either a simple periodically time-varying operating point or a DC operating point. Linearizing about a quasi-periodically time-varying operating point allows the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

The QPSP analysis also calculates noise parameters in frequency-converting circuits. QPSP computes noise figure (single-sideband, double-sideband, and IEEE single-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. As is also true for QPnoise analysis, but unlike SP analysis, the noise features of the QPSP analysis include noise folding effects due to the quasi-periodically time-varying nature of the circuit.

Computing the n-port S-parameters and noise parameters of a quasi-periodically varying circuit is a two step process.

1. First, the small stimuli are ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Second, using the QPSP analysis, the small-signal excitations are applied to compute the n-port S-parameters and noise parameters.

QPSP Output Frequencies and Sideband Vectors

To specify the QPSP analysis, you must specify the physical ports and the port harmonics combinations that form the *virtual ports* of interest. In QPSP as in PSP, port sidebands are used to assign the frequency translation between ports.

- Set the `port` parameter for the physical ports of interest
- Set either the `portharmsvec` or the `harmsvec` parameter to select the quasi-periodic small-signal output frequencies, or harmonics, of interest
  - When you use the `portharmsvec` parameter, the harmonic vectors must be in one-to-one correspondence with the ports, with one harmonic combination associated with each physical port listed in the `port` list. That is in the presence of two tones

in QPSS, each physical port needs a combination of two harmonics assigned to it, so there are two integer numbers in `portharmsvec` for each entry in port.

- ❑ When you specify harmonic combinations with the optional `harmsvec` parameter, the QPSP analysis computes all possible frequency-translating scattering parameters associated with the specified harmonics.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by a vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

K^1_1 x f(qpss-large-tone) + K^1_2 x f(qpss-moderate-tone)

When there are L tones total in the QPSS analysis (1 large tone and L-1 moderate tones), there is also a given set of n integer vectors representing the sidebands

$$K^1 = \{ K^1_1, ..., K^1_j, ..., K^1_L\}$$
$$K^2 = \{ K^2_1, ..., K^2_j, ..., K^2_L\}$$
...
$$K^n = \{ K^n_1, ..., K^n_j, ..., K^n_L\}$$

QPSP computes the S-parameters from each virtual port to all the others. The frequency translation from one virtual port to the other is calculated using the following.

$$f^i(scattered) = f^m(incident) + \sum_{i=1}^{L}\left(\left(K^i_j - K^m_j\right)\cdot f_j\right)$$

Where

- *f(scattered)* is the frequency to which the relevant scattering parameter represents the conversion
- *f(incident)* represents the relative frequency of a signal incident on a port
- *f* represents the `fundamental frequency` used in the corresponding QPSS analysis.

Input and Output Frequencies in QPSP

For the QPSP analysis, the frequency of the input and the frequency of the response are usually different (this is an important way in which QPSP analysis differs from SP analysis).

When you analyze a down-converting mixer with a signal in the upper sideband and you sweep the RF input frequency

- The most relevant harmonic for RF input is K^i={1,0}
- The most relevant harmonic for IF output is K^i={0,0}

Hence, you can associate

- K^1={1,0} with the RF port
- K^2={0,0} with the IF port.

The frequency translation will be the following

$$\Delta f(RFtoIF) = f(IF) - f(RF) = (0 \cdot f_1 + 0 \cdot f_2) - ((1 \cdot f_1 + 0 \cdot f_2)) = -f(LO)$$

S_{21} represents the transmission of the signal from the RF port with *f(incident)=f(RF)* to the port IF with *f(scattering)=f(IF)=f(RF)+*$\Delta f(RFtoIF)$*=f(incident)-f(LO)*.

S_{11} represents the reflection of the signal back to the RF port.

S_{12} represents the transmission from port IF with *f(incident)=f(IF)* to port RF with *f(scatter)=f(RF)=f(IF)-*$\Delta f(RFtoIF)$*=f(incident)+f(LO)*

If the signal is in the lower sideband, then a choice of K_1={-1,0} is more appropriate.

Because the QPSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the `freqaxis` and `sweeptype` parameters behave somewhat differently in QPSP analysis than they do in both QPAC and QPXF analyses.

The `sweeptype` parameter controls the way the frequencies are swept.

- A *`relative`* `sweeptype` indicates a sweep relative to the first virtual port harmonics vector
- An *`absolute`* `sweeptype` indicates a sweep of the absolute input source frequency

For the relative sweep with the frequency f_{rel} and the first virtual port with harmonic vector [K^1_1 K^1_2] the incident and scatter frequencies for S_{21} are

$$f_{incident} = f_{rel} + K_1^1 \cdot f_1 + K_2^1 \cdot f_2$$

$$f_{scatter} = f_{rel} + K_1^1 \cdot f_1 + K_2^1 \cdot f_2 + \Delta f(RFtoIF) = f_{rel} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2$$

while for S_{12} they are

$$f_{incident} = f_{rel} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2$$

$$f_{scatter} = f_{rel} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2 - \Delta f(RFtoIF) = f_{rel} + K_1^1 \cdot f_1 + K_2^1 \cdot f_2$$

For the *absolute sweeptype* the sweep frequency will be used as an incident frequency on the first virtual port in S_{21}

$$f_{incident} = f_{abs}$$

$$f_{scatter} = f_{abs} + \Delta f(RFtoIF) = f_{abs} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2 - \left(K_1^1 \cdot f_1 + K_2^1 \cdot f_2\right)$$

or scatter frequency for S_{12}

$$f_{incident} = f_{abs} + \Delta f(RFtoIF) = f_{abs} + K_1^2 \cdot f_1 + K_2^2 \cdot f_2 - \left(K_1^1 \cdot f_1 + K_2^1 \cdot f_2\right)$$
$$f_{scatter} = f_{abs}$$

For example, with the following parameter values including `relative sweeptype`

- QPSS fundamentals of 1000 MHz and 1010 MHz
- `portharmsvec = [0 1 -1 1]` (to examine a down converting mixer)
- `sweeptype=relative`
- sweep range of *f(rel)*`=0 -> 5` MHz

The frequency translation

$$\Delta f = (-1 \cdot f_1 + 1 \cdot f_2) - (0 \cdot f_1 + 1 \cdot f_2) = -f(LO)$$

The resulting S_{21} represents the strength of the signal transmitted from the input at the first virtual port in the range 1010->1015 MHz to the output at the second virtual port at frequencies of 10->15 MHz. Accordingly, S_{12} represents the signal transmitted from the second virtual port with $f_{incident}$=10->15 MHz to the first virtual port at $f_{scatter}$ = 1010->1015 MHz.

Using the following changed parameter values including `absolute sweeptype` calculates the same quantities.

- `sweeptype=absolute`
- sweep range of *f(abs)*=`1010->1015` MHz

Both configuration calculate the same quantities

f(abs) = `1010->1015` MHz

$f(rel) = f(abs) - (K^1_1 * f_1 + K^1_2 * f_2)$= 0 -> 5 MHz

because

- K^1_1=`0`
- K^1_2=`1`
- f_1=`1000` MHz
- f_2=`1010` MHz.

Use the `freqaxis` parameter to specify whether the results should be output versus the frequency at the first virtual port, the frequency at the second virtual port (out), or the absolute value of the frequency swept at the first virtual port (`absin`).

Note: Requesting additional ports and harmonics increases the simulation time substantially.

Noise Analysis with QPSP

The QPSP analysis performs noise analysis which includes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is performed by default and skipped only when you set `donoise=no`, requires additional simulation time beyond that required for S-parameter calculation.

Noise Calculations Performed by QPSP

Name	Description	Output Label
N_o	Total output noise at frequency f	
N_s	Noise at the output due to the input probe (the source)	
N_{si}	Noise at the output due to the image harmonic at the source	
N_{so}	Noise at the output due to harmonics other than input at the source	
N_l	Noise at the output due to the output probe (the load)	
IRN	Input referred noise	**In**
G	Gain of the circuit (See **Note:**)	**Gain**
F	Single sideband noise factor	**F**
NF	Single sideband noise figure	**NF**
F_{dsb}	Double sideband noise factor	F_{dsb}
NF_{dsb}	Double sideband noise figure	NF_{dsb}
F_{ieee}	IEEE single sideband noise factor	F_{ieee}
NF_{ieee}	IEEE single sideband noise figure	NF_{ieee}

Note: For the QPSP analysis, the gain computed is the voltage gain from the actual circuit input to the circuit output, not the gain from the internal port voltage source to the output. For the noise characterization the first virtual port is dedicated as the input, the second virtual port serves as the output.

SpectreRF performs the following noise calculations.

Input referred noise

$$IRN = \sqrt{\frac{N_o^2}{G^2}}$$

Single sideband noise factor

$$F = \frac{(N_o^2 - N_l^2)}{N_s^2}$$

Single sideband noise figure

$$NF = 10 \times \log 10(F)$$

Double sideband noise factor

$$F_{dsb} = \frac{N_o^2 - N_l^2}{N_s^2 + N_{si}^2}$$

Double sideband noise figure

$$NF_{dsb} = 10 \times \log 10(F_{dsb})$$

IEEE single sideband noise factor

$$F_{ieee} = \frac{N_o^2 - N_l^2 - N_{so}}{N_s^2}$$

IEEE single sideband noise figure

$$NF_{ieee} = 10 \times \log 10(F_{ieee})$$

Noise Folding Effects

To insure accurate noise calculations, set the `clockmaxharm` parameter to include the relevant noise folding effects.The `clockmaxharm` parameter is only relevant to the noise computation features of QPSP.

Swept QPSP Analysis

Specify sweep limits by providing either the end points (start and stop) or by providing the center value and the span of the sweep (center and span).

Specify sweep steps as linear or logarithmic. Either specify the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10, and logarithmic when this ratio is 10 or greater.

Alternatively, you may specify the particular values that the sweep parameter should take using the `values` parameter. If you give both a specific set of values and a set of values specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

QPSP Parameters

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.
sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.

Port Parameters

ports=[...]	List of active ports. Ports are numbered in the order given. For purposes of noise figure computation, the input is considered port 1 and the output is considered port 2.

portharmsvec=[...]	List of harmonics active on the specified list of ports. The `portharmsvec` vector must have a one-to-one correspondence with the `ports` vector.
harmsvec=[...]	Secondary list of additional, active harmonic's combinations. Harmonics included in the `harmsvec` vector are in addition to those associated with specific ports by `portharmsvec`.

Output Parameter

freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the input frequency. Default is `in`. Possible values are `absin`, `in` or `out`.

Noise Parameter

donoise=yes	Perform noise analysis. If *`oprobe`* is specified as a valid port, `donoise` is set to *`yes`*, and a detailed noise output is generated. Possible values are `no` or `yes`.

Probe Parameter

clockmaxharm=7	Maximum clock harmonics range included when computing noise either up-converted or down-converted to the output by the periodic drive signal. The `clockmaxharm` parameter is only relevant to the noise computation features of QPSP (when `donoise=yes`).

Convergence Parameters

tolerance=1e-9	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration, 1 or 2.
solver=turbo	Solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Quasi-Periodic Transfer Function Analysis

The quasi-periodic transfer-function (QPXF) analysis produces transfer-functions that include multitone frequency translation. The QPXF analysis computes the transfer functions from any source at any frequency to a single output at a single frequency.

The QPXF analysis directly computes

- Conversion Efficiency --The transfer function from the input to the output at a specified frequency.
- Image Rejection and Sideband Rejection -- The input to output at an undesired frequency.
- LO Feed-Through and Power Supply Rejection -- Undesired input to output at all frequencies.

Computing the small-signal response of a quasi-periodically varying circuit is a two step process.

1. First, the small stimulus is ignored and the quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed with a QPSS analysis. As a normal part of the QPSS analysis, the quasi-periodically time-varying representation of the circuit is computed and saved for later use.
2. Second, the small stimulus is applied to the quasi-periodically time-varying linear representation and the small signal response is computed. This is done using the QPXF analysis.

QPXF Output Frequencies and Sideband Vectors

Either voltage or current can be the QPXF output variable of interest. The variable's frequency is not constrained by the period of the large signal quasi-periodic solution. When you sweep a selected output frequency, you select the set of quasi-periodic small-signal input frequencies you are interested in by setting either the `clockmaxharm` or the `sidevec` output parameter.

In quasi-periodic analyses sidebands are vectors, or, in other words, harmonic combinations. One way to specify them is using the `sidevec` parameter. When a QPSS analysis has one large tone and one moderate tone, the sideband is represented by a vector K^1 as $[K^1_1\ K^1_2]$. The corresponding frequency translation is

$$K^1_1 \times f(\text{qpss-large-tone}) + K^1_2 \times f(\text{qpss-moderate-tone})$$

When there are `L` tones total in the QPSS analysis (`1` large tone and `L-1` moderate tones), there is also a given set of `n` integer vectors representing the sidebands

$K^1 = \{ K^1_1, ..., K^1_j, ..., K^1_L \}$
$K^2 = \{ K^2_1, ..., K^2_j, ..., K^2_L \}$
…
$K^n = \{ K^n_1, ..., K^n_j, ..., K^n_L \}$

The QPXF analysis computes the output frequency corresponding to each sideband as follows.

$$f^i(in) = f(out) + \sum_{i=1}^{L} \left(K^i_j \cdot f_j \right)$$

where

- *f(out)* represents the (possibly swept) output signal frequency
- f_j represents the fundamental frequency used in the corresponding QPSS analysis.

When you analyze a down-converting mixer, while sweeping the IF output frequency

- K_i=`{1,0}` for the RF input represents the first upper-sideband
- K_i=`{-1,0}` for the RF input represents the first lower-sideband.

Enter the `sidevec` parameter as a sequence of integer numbers separated by spaces. For example, you would enter the set of vectors `{1 1 0} {1 -1 0} {1 1 1}` as follows

```
sidevec=[ 1 1 0 1 -1 0 1 1 1]
```

The other way to specify the sidebands is the `clockmaxharm` parameter and the `maxharms` parameter in preceding QPSS analysis. Only the large tone, the first fundamental in QPSS, is affected by the QPAC `clockmaxharm` parameter value. It limits the maximum harmonic order of the large tone that will be considered. All the remaining tones, the moderate tones, are limited by the QPSS `maxharms` parameter value.

Given the following parameters

- In the QPSS analysis, `maxharms=[`k^0_{max} k^2_{max} ... k^n_{max}]
- In the QPXF analysis, `clockmaxharm=`K_{max}

The QPXF analysis output generates the following number of sidebands

$(2*K_{max}+1)*(2*k^2_{max}+1)*(2*k^3_{max}+1)*...*(2*k^n_{max}+1)$

Note: The number of sidebands you request substantially increases the simulation time.

For a QPXF analysis, the stimulus and response frequencies are usually different (this is an important way in which QPXF differs from XF analysis).

Use the QPXF `freqaxis` parameter to specify how to output the QPXF simulation results. For

- `freqaxis=`*in* the results are output versus the input frequency
- `freqaxis=`*out* the results are output versus the output frequency
- `freqaxis=`*absin* the results are output versus the absolute value of the input frequency

You can specify the output for the QPXF analysis with either a probe component or with a pair of nodes. Any component with two or more terminals can be a voltage probe. When there are more than two terminals, the terminals are grouped in pairs and you use the `portv` parameter to select the appropriate pair of terminals. Alternatively, you can simply give a pair of nodes to specify a voltage as the output.

Any component that naturally computes current as an internal variable can be a current probe. If the probe component computes more than one current, use the `porti` parameter to select the appropriate current. Do not specify both `portv` and `porti` parameters for a simulation. If you specify neither a `portv` or `porti` parameter, the probe component provides a reasonable default.

Transfer Function Inputs

The QPXF `stimuli` parameter specifies the transfer function inputs. You have two choices.

- The `stimuli=sources` parameter value uses the sources present in the circuit as inputs. You can adjust the computed gain to compensate for gains or losses in a test fixture with the `xfmag` parameters provided by the sources. You can use the `save` and `nestlvl` parameters to limit the number of sources in hierarchical netlists.
- The `stimuli=nodes_and_terminals` parameter value computes all possible transfer functions. This is useful when you do not know in advance which transfer functions might be interesting.

Transfer functions for nodes are computed assuming that a unit magnitude flow (current) source is connected from the node to ground. Transfer functions for terminals are computed assuming that a unit magnitude value (voltage) source is connected in series with the terminal.

By default, the transfer functions are computed from a small set of terminals.

- If you want transfer functions from specific terminals, specify the terminals in the `save` statement. Use the `:probe` modifier (ex. `Rout:1:probe`) or specify `useprobes=yes` on the `options` statement.
- If you want transfer functions from all terminals, specify `currents=all` and `useprobes=yes` on the `options` statement.

Swept QPXF Analysis

Specify sweep limits by providing either the end points (`start` and `stop`) or by providing the center value and the span of the sweep (`center` and `span`).

Specify sweep steps as linear or logarithmic as well as the number of steps or the size of each step. You can give a step-size parameter (`step`, `lin`, `log`, `dec`) to determine whether the sweep is linear or logarithmic. If you do not give a step-size parameter, the sweep is linear when the ratio of `stop` to `start` values is less than 10, and logarithmic when this ratio is 10 or greater.

Alternatively, you may specify the particular values that the sweep parameter should take using the `values` parameter. If you give both a specific set of values and a set specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hertz.

QPXF Parameters

Sweep Interval Parameters

start=0	Start sweep limit.
stop	Stop sweep limit.
center	Center of sweep.
span=0	Sweep limit span.
step	Step size, linear sweep.
lin=50	Number of steps, linear sweep.
dec	Points per decade.
log=50	Number of steps, log sweep.
values=[...]	Array of sweep values.

sweeptype	Specifies if the sweep frequency range is absolute frequency of input or if it is relative to the port harmonics. Possible values are absolute or relative.
relharmnum=[...]	Harmonic to which relative frequency sweep should be referenced.

Probe Parameters

probe	Compute every transfer function to this probe component.

Output Parameters

stimuli=sources	Stimuli used for xf analysis. Possible values are sources or nodes_and_terminals.
sidevec=[...]	Array of relevant sidebands for the analysis.
clockmaxharm=0	An alternative to the ‘sidevec’ array specification, which automatically generates the array: [- clockmaxharm ... 0 ... +clockmaxharms] [- maxharms(QPSS)[2]...0...maxharms(QPSS)[2]][...].
freqaxis	Specifies whether the results should be output versus the input frequency, the output frequency, or the absolute value of the input frequency. Default is ‘out’ for logarithmic frequency sweeps and ‘absin’ otherwise. Possible values are absin, in or out.
save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.

Convergence Parameters

tolerance=1e-9	Relative tolerance for linear solver.
gear_order=2	Gear order used for small-signal integration, 1 or 2.
solver=turbo	Solver type. Possible values are std or turbo.

Annotation Parameters

annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
stats=no	Analysis statistics. Possible values are no or yes.
title	Analysis title.

Envelope Following Analysis

RF circuit designers who simulate the modulation schemes (AM and PM) of communications systems want efficient and accurate prediction of the envelope transient response of RF circuits. Typically, these modulation schemes are difficult to simulate because they are clocked at a frequency whose period is orders of magnitude smaller than the time interval of interest. Using classical transient simulation for these circuits is time-consuming, and new RF analyses, such as PSS and QPSS might not work directly because the modulation signal is sometimes stochastic and might be neither periodic nor quasi-periodic.

Envelope Following Analysis (Envlp) permits efficient simulation of these modulation schemes and reduces the simulation time without compromising accuracy. This analysis depends on the property that circuit behavior in a fixed given high-frequency clock cycle is similar, but not identical, to the behavior of the preceding and following clock cycles. In particular, the *envelope* of the high-frequency clock can be followed by accurately computing the circuit behavior over occasional cycles that accurately model the fast transient behavior. The slow-varying modulation is accurately followed by a piecewise polynomial. As a result, the spectrum of the circuit response can be obtained by combining the piecewise polynomial and the integration of the occasional clock cycles.

The Envlp analysis computes the envelope response of a circuit. You specify the analysis `clockname` parameter. The simulator automatically determines the clock period by examining all the sources with the specified name. The envelope response is computed over the interval from the `start` to `stop` parameter values. If the interval you specify is not a multiple of the clock period, it is rounded off to the nearest multiple before the `stop` time. The initial condition is assumed to be the DC steady-state solution unless you specify otherwise.

Envelope Following Analysis is most efficient for circuits whose modulation bandwidth is orders of magnitude lower than the clock frequency. This is typically the case, for example, in circuits whose clock is the only fast varying signal, and whose other input signals have a spectrum with a frequency range orders of magnitude lower than the clock frequency.

In general, the Envlp analysis is not intended for circuits working with multiple fundamentals. However, like PSS analysis, you can use it for specific classes of circuits that operate with multiple fundamentals.

You can, for example, use Envelope Following Analysis when the multiple fundamentals are commensurate. In this situation, you use the greatest common denominator of all fundamental frequencies as the clock frequency. This is similar to computing the beat frequency for PSS analysis. The analysis efficiency is also analogous to PSS analysis. At each integration of the clock period, many fast cycles might be required. Greater numbers of fast cycles decrease the efficiency of the analysis.

There is a second situation when Envelope Following Analysis is useful for circuits with multiple fundamentals. This is when the down conversion of two closely placed frequencies can also generate a slow-varying modulation envelope whose frequency is orders of magnitude lower than the input frequencies. In this case, you can use Envelope Following Analysis to trace out the modulation envelope by choosing either of the fast-varying signals as the clock. However, you usually choose the signal that causes the most nonlinearity.

Envelope Following Analysis generates two types of output files:

- A voltage versus time (*td*) file
- An amplitude/phase versus time (*fd*) file for each specified harmonic of the clock fundamental.

The *td* file contains time-domain real waveforms. They are similar to the waveforms generated by transient analysis. The difference between the analyses is that the integration of a clock cycle occurs much less frequently with Envelope Following Analysis. Consequently, you usually see big gaps between integrated clock cycles. You can express the signals using time varying Fourier coefficients.

$$v(t) = \sum_{k=1}^{M} a_k(t) e^{jk\omega_0 t}$$

The *fd* file contains time varying Fourier coefficients $a_k(t)$(complex) up to harmonic M, where M is specified using the `harms` parameter. The most interesting harmonic is probably the first one. The real and imaginary parts of this harmonic are called the in-phase and quadrature components of the baseband signal. Assume, for example, that you simulate a transmitter that is driven by a complex baseband signal. For a phase-modulation scheme, you plot the phase of the time-varying Fourier coefficients of harmonic 1 versus time to see the transmitted phase. To see the trajectory of the transmitted baseband signal, plot the real versus the imaginary parts of the time-varying Fourier coefficients of harmonic 1. The trajectory lets you visually check the error vector magnitude (EVM). To compute the transmitted power spectral density about the carrier, compute the power spectral density of the time-varying Fourier coefficients of harmonic 1. From the power spectral density, you can estimate Adjacent Channel Power Ratio (ACPR).

The spectrum of each harmonic response is calculated from the Analog Circuit Design Environment. This is useful for applications such as ACPR calculation.

You can also use the *fd* result for applications such as constellation diagrams, by plotting amplitude versus phase.

Envlp Synopsis

Envelope Fundamental Parameters

clockname	Name of the clock fundamental.
modulationbw (Hz)	Modulation bandwidth.

Simulation Interval Parameters

stop (s)	Stop time.
start=0 s	Start time.
tstab=0 s	Initial stabilization time.
outputstart=start s	Output is saved only after this time is reached.

Time-Step Parameters

maxstep (s)	Maximum time step for inner transient integration. Default derived from ‘errpreset’.
envmaxstep (s)	Maximum outer envelope step size. Default derived from `errpreset`.

Initial-Condition Parameters

ic=all	What should be used to set initial condition. Possible values are dc, node, dev, or all.
skipdc=no	If yes, there will be no dc analysis for initial transient. Possible values are no or yes.
readic	File that contains initial transient condition.

Convergence Parameters

readns	File that contains the estimate of the initial DC solution.
cmin=0 F	Minimum capacitance from each node to ground.

State-File Parameters

write	File to which initial transient solution is to be written.
writefinal	File to which final transient solution is to be written.

swapfile	Temporary file that holds the matrix information used by Newton’s method. Tells Spectre to use a regular file rather than virtual memory to hold the matrix information. Use this option if Spectre complains about not having enough memory to complete this analysis.

Integration Method Parameter

method	Integration method. Default derived from ‘errpreset’. Possible values are euler, trap, traponly, gear2, or gear2only.

Accuracy Parameters

errpreset=moderate	Selects a reasonable collection of parameter settings. Possible values are conservative, moderate or liberal.
relref	Reference used for the relative convergence criteria. Default derived from ‘errpreset’. Possible values are pointlocal, alllocal, sigglobal, or allglobal.
lteratio	Ratio used to compute LTE tolerances from Newton tolerance. Default derived from errpreset.
steadyratio	Ratio used to compute steady state tolerances from LTE tolerance. Default derived from ‘errpreset’.
envlteratio	Ratio used to compute envelope LTE tolerances. Default derived from ‘errpreset’.

Annotation Parameters

stats=no	Analysis statistics. Possible values are no or yes.
annotate=sweep	Degree of annotation. Possible values are no, title, sweep, status, or steps.
title	Analysis title.

Output Parameters

harms=1	Number of clock harmonics to output.
harmsvec=[...]	Array of desired clock harmonics. Alternate form of `harms` that allows selection of specific harmonics.
outputtype=both	Output type. Possible values are both, envelope or spectrum.

save	Signals to output. Possible values are all, lvl, allpub, lvlpub, selected, or none.
nestlvl	Levels of subcircuits to output.
compression=no	Do data compression on output. Possible values are no or yes.
strobeperiod (s)	The output strobe interval (in seconds of envelope following time). The actual strobe interval is rounded to an integer multiple of the clock period.

Newton Parameters

maxiters=5	Maximum number of Newton iterations per transient integration time step.
envmaxiters=3	Maximum number of Newton iterations per envelope step.
restart=yes	Do not use the previous DC solution as an initial guess. Possible values are no or yes.

Circuit Age

circuitage (Years)	Stress Time. Age of the circuit used to simulate hot-electron degradation of MOSFET and BSIM circuits.

The simulator determines the clock frequency by examining all the sources whose name matches the clock name specified by the `clockname` parameter. If multiple frequencies are found, the least common factor of these frequencies is used as the clock frequency.

The maximum envelope step size is affected by many parameters. It can be directly limited by the `envmaxstep` parameter. It is also limited by `modulationbw`. The user gives an estimate of the modulation bandwidth. The simulator will put at least eight points within the modulation period.

The `harms` and `harmsvec` parameters do not affect the simulation time in any significant way. The `harms` and `harmsvec` parameters affect only post processing steps that occur after each envelope integration step has converged. Compared to integration steps and shooting iterations the cost of the post processing step is insignificant. This is an advantage of shooting methods over harmonic balance methods.

The time varying spectrum is calculated for all the specified harmonics for all sampled integration cycles as the Envlp analysis marches on. For each harmonic, an *fd* file is generated. Typically, `harms` is set to `1` or `2`.

The errpreset Parameter in Envlp Analysis

Most parameters of the Envlp analysis are inherited from either transient or PSS analysis and their meanings are consistent. The effect of the `errpreset` parameter on some other Envlp analysis parameters is shown in Table 1-5.

Table 1-5 Envlp Parameter Default Values for `errpreset` Settings

errpreset	envmaxstep	steadyratio	envlteratio
liberal	Interval/10	1.0	10.0
moderate	Interval/50	0.1	1.0
conservative	Interval/100	0.01	0.1

The effect of the `errpreset` parameter on parameters such as reltol, relref, method, maxstep, and lteratio are the same as defined in Table 1-4.

2

SpectreRF Simulation Form Reference

The SpectreRF simulation forms include the Choosing Analysis form, the Options forms, and the Results forms. The simulation forms change to display only the fields relevant for the currently selected analysis.

The field description topics are presented in the following sections.

- "The Choosing Analyses Form" on page 111
- "Field Descriptions for the Options Forms" on page 174
- "Field Descriptions for the Plot Form" on page 191
- "The ACPR Wizard" on page 211

Within each section, the form field descriptions are arranged alphabetically according to the top-level headings on the forms. The top-level headings are usually found at the leftmost margin of the form.

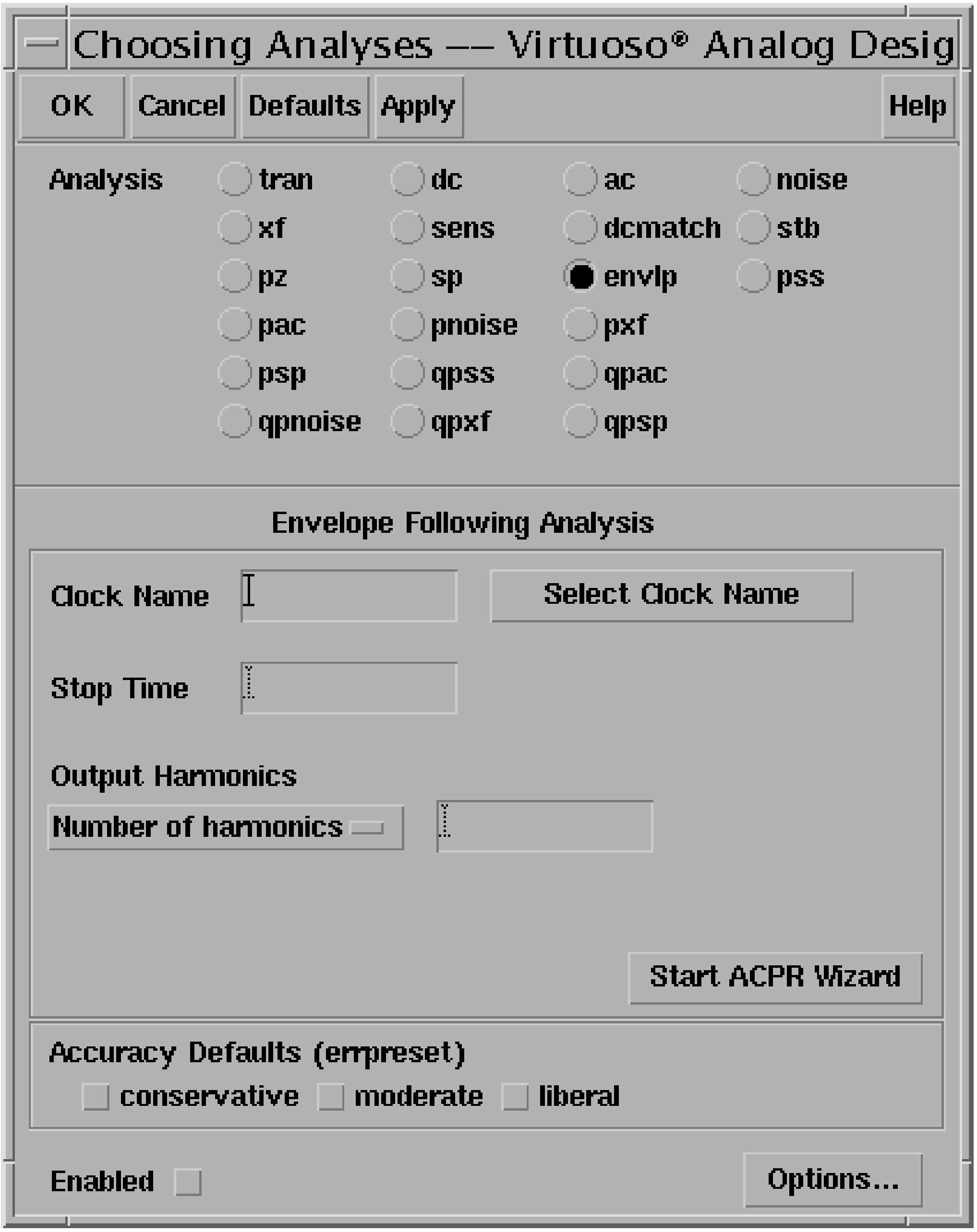
The Choosing Analyses Form

Use the Choosing Analyses form in the Analog Circuit Design Environment (the Simulation window) to select and set up RF simulations.

Opening the Choosing Analysis Form

In the Simulation window, select the *Analyses – Choose* command to open the Choosing Analysis form. Figure 2-1 on page 112 shows the Choosing Analysis form for the Envelope Following analysis. The content of the Choosing Analyses form changes depending on the analysis selected.

Figure 2-1 The Choosing Analyses Form



The Analysis section at the top of the Choosing Analyses form displays the available Spectre analyses, including the spectreRF analyses (`envlp`. `pss`, `pac`, `pnoise`, `pxf`, `psp`, `qpss`, `qpac`, `qpnoise`, `qpxf` and `qpsp`). When you highlight an analysis in this section, the form changes to display the title of the analysis (directly below the analysis section), and below the title, relevant parameters for the selected simulation.

At the bottom of the form, highlight *Enabled* to select and run the analysis with the next simulation. Click *Options* to display the Options form for the selected analysis. Each Options form displays only the parameters relevant for that particular analysis.

The Spectre RF Analyses

The RF analyses are

- The periodic large-signal `pss`, periodic steady state analysis
- The periodic large-signal `qpss`, quasi-periodic steady state analysis
- The periodic small-signal analyses: `pac`, `psp`, `pnoise`, and `pxf`
- The quasi-periodic small-signal analyses: `qpac`, `qpsp`, `qpnoise`, and `qpxf`
- The envelope following `envlp` analysis

When you highlight an analysis type, the Choosing Analysis form changes to allow you to specify information for that simulation.

When your simulation requires that two analyses be run (for example, a `pss` large-signal analysis followed by a `pnoise` small-signal analysis), the Choosing Analysis form maintains the simulation set-up data for the two simulations interactively. For example, when you highlight `pnoise`, the values displayed in the Choosing Analysis form will reflect the information you entered for the `pnoise` analysis. When you highlight `pss`, the values displayed in the Choosing Analysis form will reflect the information you entered for the `pss` analysis.

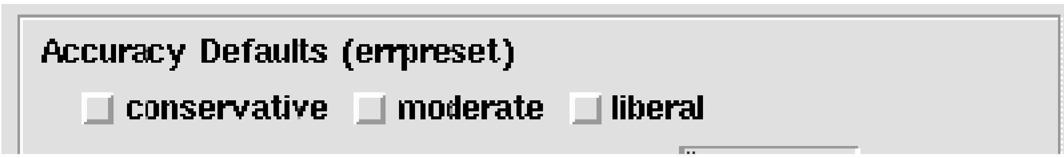
Run the periodic small-signal `pac`, `psp`, `pnoise` and `pxf` analyses after a large-signal `pss` analysis. Run the quasi-periodic small-signal `qpac`, `qpsp`, `qpnoise` and `qpxf` analyses after a large-signal `qpss` analysis.

Field Descriptions for the Choosing Analysis Form

The following sections describe all the fields found on the Choosing Analysis form. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the form. See “The RF Simulation Forms Quick Reference” on page 218 for a brief description of the contents of each form.

Accuracy Defaults (errpreset) (PSS, QPSS, and Envlp)

Quickly adjusts the simulator parameters.



The `errpreset` parameter quickly adjusts the simulator accuracy parameters to fit your needs. In most cases, `errpreset` should be the only parameter you need to adjust.

- For a fast simulation with reasonable accuracy, set `errpreset` to *`liberal`*.
- For greater accuracy, set `errpreset` to *`moderate`*.
- For maximum accuracy, set `errpreset` to *`conservative`*.

The effect of `errpreset` on other parameters varies depending on the type of analysis to which you are applying it. For details see

- “The errpreset Parameter in PSS Analysis” on page 38
    - “Using the errpreset Parameter With Driven Circuits” on page 38
    - “Using the errpreset Parameter With Autonomous Circuits” on page 39
- “The errpreset Parameter in QPSS Analysis” on page 78
- “The errpreset Parameter in Envlp Analysis” on page 110

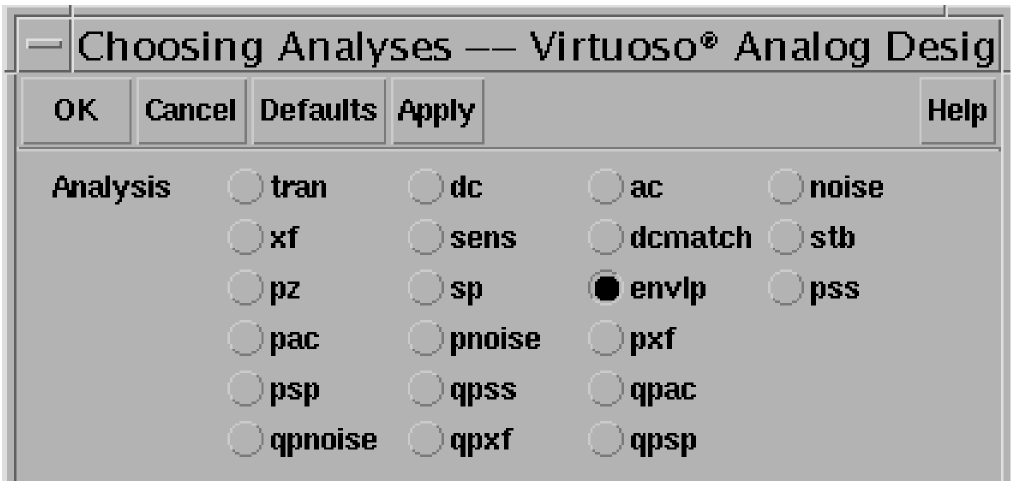
Additional Time for Stabilization (tstab) (PSS and QPSS)

Specifies an amount of additional time to allow for the circuit to settle.

Use `tstab` if the circuit exhibits more than one periodic solution and you want only one. A long `tstab` can also improve convergence.

Analysis

Selects the SpectreRF analysis type.



The RF analyses are

- The periodic large-signal `pss` analysis
- The quais-periodic large-signal `qpss` analysis
- The periodic small-signal analyses: `pac`, `psp`, `pnoise`, and `pxf`
- The quasi-periodic small-signal analyses: `qpac`, `qpsp`, `qpnoise`, and `qpxf`
- The Envelope Following analysis, `envlp`

When you highlight an analysis type, the Choosing Analysis form changes to allow you to specify information for that simulation. Below the Analysis buttons, the analysis title changes to the name of the analysis you select.

When your simulation requires that two analyses be run (for example, a `pss` analysis followed by a `pnoise` small-signal analysis), the Choosing Analysis form maintains the simulation set-up data for both simulations. You can edit the date for both simulations interactively. For example, when you highlight `pnoise`, the values displayed in the `pnoise` Choosing Analysis form will reflect the information you entered for the `pnoise` analysis.

Run the periodic small-signal analyses `pac`, `psp`, `pnoise`, and `pxf`, after a large-signal `pss` analysis. Run the quasi-periodic small-signal `qpnoise` analysis after a `qpss` analysis.

Beat Frequency, Beat Period, and Auto Calculate (PSS)

Determines whether the `pss` analysis uses beat frequency or beat period and supplies an initial value.

- Highlight either *Beat Frequency* or *Beat Period*.

  The *Beat Frequency* (or *Beat Period*) field is initially empty.

- Enter a *Beat Frequency* value in one of two ways
  - Type a frequency value for which all the tone frequencies are integer multiples of the value.
  - Click the *Auto Calculate* button to automatically calculate and enter a value into the field.

    The *Beat Frequency* field value is calculated based on the tones present in the Fundamental Tones list box. It is the greatest common multiple of all the tone frequencies that are not small signals.

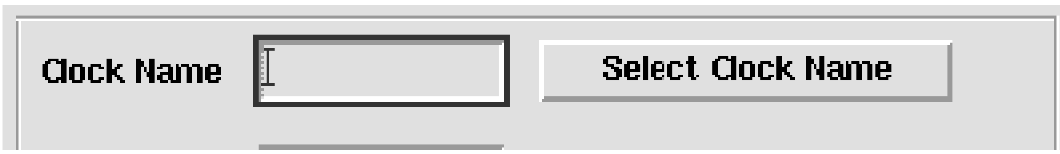
- Enter a *Beat Period* value in one of two ways.
  - Type a period value for which all the tone frequencies are integer multiples of the inverse of the value.
  - Click the *Auto Calculate* button to automatically calculate and enter a value into the field.

    The *Beat Period* field value is calculated based on the tones present in the Fundamental Tones list box. The *Beat Period* field value is set to the inverse of the frequency.

The *Beat Frequency* (or *Period*) value is displayed in a read-only field at the top of the periodic small-signal analysis forms.

Clock Name and Select Clock Name Button (Envlp)

Identifies the clock signal for an Envelope Following Analysis.

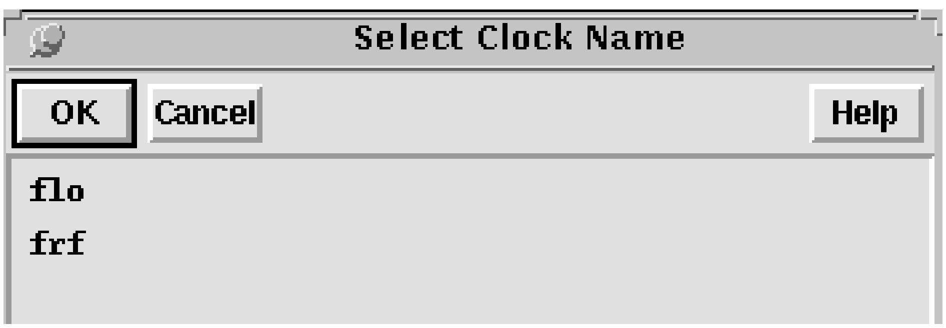


Enter a clock signal name in the *Clock Name* field in one of two ways

- By typing the clock signal name in the *Clock Name* field

  or

- By clicking the *Select Clock Name* button to display a list of clock signals



  - Click to highlight a clock signal from the list.
  - Click *OK* to select the signal and display it in the *Clock Name* field.

See “Stop Time (Envlp)” on page 161 for related information.

Do Noise (PSP and QPSP)

Performs noise measurements during the `psp` or `qpsp` analysis.

- Highlight *yes* for *Do Noise* to compute noise figures, equivalent noise sources, and noise parameters during the analysis.
- To include relevant noise folding effects, specify a maximum sideband value in the *MaximumSideband* field. A sideband array of the form

  `[-max. sideband . . . 0 . . . + max. sideband]`

  is automatically generated.

Enabled

Includes the analysis in the next simulation.

Highlight *Enabled* to perform the analysis in the next simulation. Enabled analyses are listed in the Simulation window.

Click *Options* to display the Options Form for that analysis.

Frequency Sweep Range (Hz), Sweep Type, and Add Specific Points (All Small-Signal Analyses)

Defines the analysis sweep range, sweep type, and any additional sweep points for a small-signal analysis. The `pac`, `pnoise`, `pxf`, and `psp` periodic small-signal analyses follow a `pss` large-signal analysis. The `qpac`, `qpnoise`, `qpxf`, and `qpsp` quasi-periodic small-signal analyses follow a `qpss` analysis.

Frequency Sweep Range (Hz)

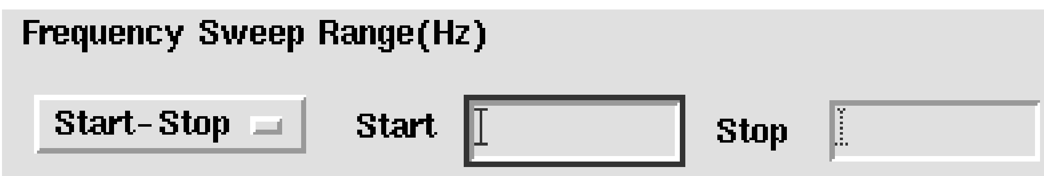
Defines the bounds for the small-signal analysis. Choices are: *Start-Stop*, *Center-Span*, and *Single-Point*.

Note: For a small-signal analyses following a swept `pss` or `qpss` analysis, *Single-Point* and *Freq* is the only *Frequency Sweep Range (Hz)* option for the small-signal analyses.

When you make a selection from the *Frequency Sweep Range (Hz)* cyclical field, the form fields change to let you specify appropriate data.

Start - Stop

Defines the beginning and ending points for the sweep.



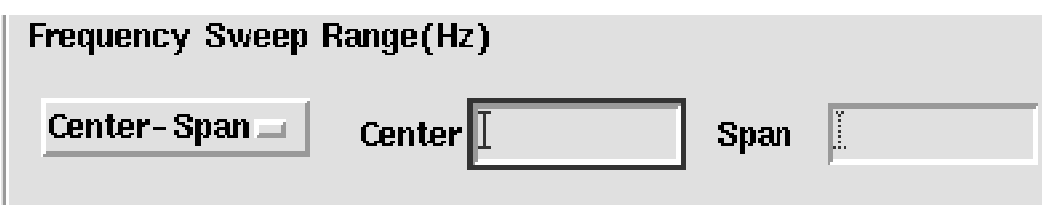
- Highlight *Start-Stop*.

  The form changes to let you type the start and stop points.

- Type the initial point for the sweep in the *Start* field.
- Type the final point in the *Stop* field.

Center - Span

Defines the center point for the sweep and its span.



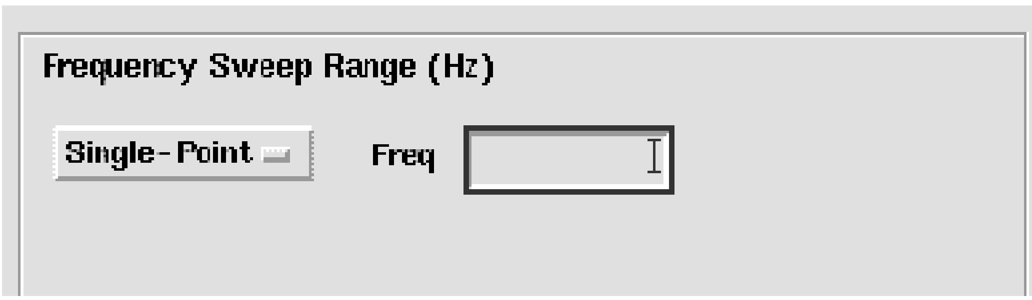
- Highlight *Center-Span*.

  The form changes to let you type the center point and span.

- Type the midpoint for the sweep in the *Center* field.
- Type the span in the *Stop* field.

Single - Point and Freq

Defines the frequency range as a single point and prompts you for the point value.



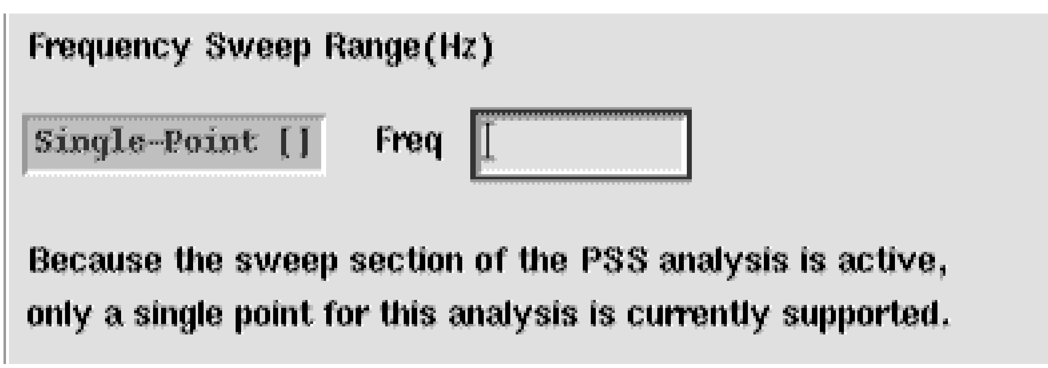
- Highlight *Single-Point*.

  The form changes to let you type the frequency.

- Type the specific frequency for the small-signal analysis.

Single - Point and Freq Following Swept PSS or Swept QPSS Analysis

When the small-signal analyses follows a swept `pss` or `qpss` analysis, the only *Frequency Sweep Range (Hz)* option for the small-signal analyses is *Single-Point* and *Freq*.



In the *Freq* field, type the specific frequency for the small-signal analysis that follows each pass of a `pss` or `qpss` sweep.

Sweep Type

Specifies whether the small-signal sweep is linear, logarithmic, or automatic.

Linear

Specifies a linear sweep.

- Highlight *Linear*.

  The form changes to let you type either the step size or the total number of points (steps).

  - ❑ Highlight *Step Size* and type the size of each step in the field.

    or

  - ❑ Highlight *Number of Steps* and type the number of steps in the field.

Logarithmic

Specifies a logarithmic sweep.

- Highlight *Logarithmic*.

The form changes to let you type either the number of points per decade or the total number of points (steps).

- ❑ Highlight *Points per Decade* and type the number of points per decade in the field.

  or

- ❑ Highlight *Number of Steps* and type the number of steps in the field.

Automatic

Lets the simulator determine whether the *Sweep Type* is *Linear* or *Logarithmic*.

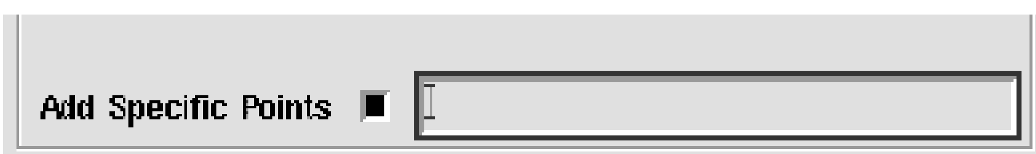
- ■ Highlight *Automatic*.

  The *Sweep Type* is

  - ❑ *Linear* if the ratio of *Start* to *Stop* values is less than 10.
  - ❑ *Logarithmic* if the ratio of *Start* to *Stop* values is10 or higher.

Add Specific Points

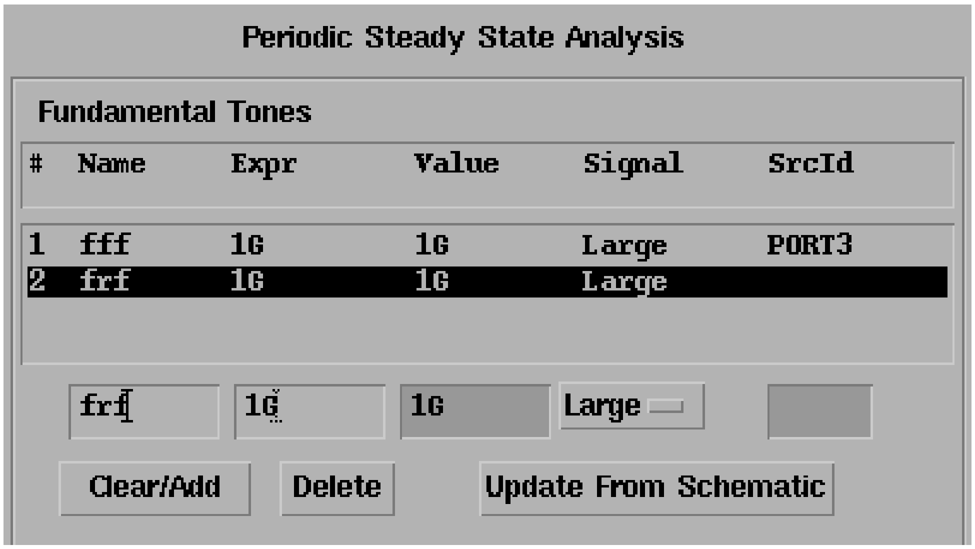
Specifies additional sweep points for the small-signal analysis.



Highlight *Add Specific Points* and type the additional sweep point values into the field. Separate them with spaces.

Fundamental Tones (PSS and QPSS)

The *Fundamental Tones* fields include the following list box, data entry fields, and data entry buttons. In addition the `pss` analysis includes the related *Beat Frequency*, *Beat Period*, and *Auto Calculate* buttons.



Fundamental Tones List Box

The *Fundamental Tones* list box displays information about every top-level tone in the circuit that has both a non-zero frequency or period value, and a non-zero amplitude value (absolute). All tones in the list box are arranged alphabetically by name.

For each tone entry, you can edit

- The *Signal* level designation (for both `PSS` and `QPSS` analyses)
- The *Harms* (Harmonic Range) value (for `QPSS` analysis only).

To edit values for a tone, highlight the tone in the list box then edit in the data entry fields.

For tones that are not at the top level of the schematic, you can manually create a tone entry by typing the pertinent information in the data entry fields.

For non-small-signal tones, each tone name and its correlated frequency value are used in the *Select from range* and *Array of coefficients* fields in the *Output Harmonics* and *Sidebands* sections of the `pss`, `pac`, `pnoise`, `pxf`, and `qpnoise` small-signal analysis forms.

Fundamental Tones List Box Terms and Data Entry Fields

- **Name** – Displays the name assigned to the tone. This tone name must be entered into the pertinent Component Description Format (CDF) fields of each source in the schematic with a tone. The current CDF name field prompts are “First frequency name”, “Second frequency name”, “Frequency name”, and “Frequency name for 1/period”.
- **Expr** – Displays the value or expression representing the frequency of a particular tone. The expression can also be a user variable or it can contain user variables. If the frequency for the tone is specified as a variable, the *Expr* field displays the name of the variable. Otherwise, the field displays the numerical value of the frequency.
- **Value** – Displays the evaluated value of the *Expr* field using the current values of the user variables.
- **Signal** – This cyclic field displays one of three values: *Large*, *Moderate*, or *Small*.
    - For qpss analysis, you must select one Large tone. Specify *Moderate* for all other tones you want to include in the simulation.
    - For pss analysis, specify *Large* for all tones you want to include in the simulation.

    Tones that you specify as *Small* are ignored by both pss and qpss analysis.
- **SrcId** – Displays the instance name of the source in the schematic where the tone is declared.
- **Harms** – Used *only* with qpss analysis.
    - Specifies the range of harmonics, which in turn determines the maximum number of harmonics of the tone to be used during the simulation.
    - The *Harms* value must be 1 or higher for the tone to be included in the simulation. The default value is 1.
    - Cannot deal with AHDL sources unless they are done with inlined Spectre sources.

Setting Up Tones for a QPSS Analysis

When you set up tones for a qpss analysis,

- Designate one signal as the *Large* tone. Designate all other tones as *Moderate*. Any tone you designate as *Small* is excluded from qpss analysis.
- Designate a non-zero harmonic range (*Harms*) value for each tone. Never set the harmonic range value to zero. Your simulation will not run properly and you might get incorrect results.

By choosing *Large* as the *Signal* value for a tone, you specify that tone to be the *Large* or clock signal. Each qpss analysis must have one tone set to be the *Large* signal.

Select your *Large* signal to be the signal that

- Causes the largest response in the circuit
- Is the least sinusoidal signal in the circuit
- Causes the most nonlinearity in the circuit

Choose at least one harmonic for each signal that you want to include in a qpss analysis. A signal with an harmonic range (*Harms*) value of 0 is ignored by the simulation. In general, when selecting a *Harms* value:

- For the *Large* signal, in most cases, set the *Harms* value to be equal to or greater than 5. An harmonic range of 5 gives 11 harmonics (-5, ..., 0, ... +5) for the *Large* tone.

  The *Harms* value you use for the *Large* signal varies with the circuit you are analyzing. For example, for a down converting mixer, a *Harms* value of 1 is sufficient. Entering a large *Harms* value for the large tone does not affect the simulation run time.

- For *Moderate* tones, set the *Harms* value to be approximately 2 or 3. The *Harms* value you use varies with the circuit you are analyzing.

  - Setting the *Harms* value to 2 gives you up to the 3rd order intermodulation terms
  - Setting the *Harms* value to 3 gives you up to the 5th order intermodulation terms

  To obtain higher order intermodulation terms, you can increase the *Harms* value accordingly. However, for *Moderate* tones, increasing the *Harms* value increases the simulation run time.

For example, when you specify *Harms* values of 5 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [5,2]` which gives you 11 harmonics for the *Large* tone and 5 harmonics (-2, -1, 0, 1, 2) for the *Moderate* tone. As a result, you get noise from 11 x 5 = 55 frequency sidebands.

The *Harms* value you select depends on the degree of nonlinearity the signal causes. If the signal is not nonlinear, then a *Harms* value of 1 is sufficient. For a moderate signal, a few harmonics should be sufficient to accurately capture the nonlinearity. However, it is hard to determine a priori what is sufficient. Generally, for a given *Harms* value, if increasing it does not significantly change the spectrum results, then the *Harms* value is high enough. In some situations, you could use a *Harms* value as high as 9. For a high *Harms* value, the algorithm still works, but not as efficiently.

Fundamental Tones Data Entry Buttons

- **Clear/Add** – Clears the data entry fields for the purpose of manually adding a new tone to the list box. This button also resets the list box so that no line is currently selected.
- **Delete** – Deletes a tone selected in the *Fundamental Tones* list box.

  You cannot use the *Delete* button to delete tones that are specified in the schematic. Such tones are deleted by setting the CDF frequency or period field value to zero or blank and CDF amplitude value(s) to zero (absolute) or blank.
- **Update From Schematic** – Updates the values in the *Fundamental Tones* list box from the schematic.

  Note: Before you can update from the schematic, you must have performed a *Check and Save* on the design in the Schematic window.

Using the Fundamental Tones Data Entry Fields and Buttons

Use the data entry fields below the list box to edit the information in the *Fundamental Tones* list box or to add new tones.

To specify a new tone,

1. Make sure there are no selected tones in the list box. If a there is a selected tone, click the *Clear/Add* button.
2. Type a value in any of the data entry fields (typically starting with the *Name* field).

   As you advance to a second data entry field, a new tone line is added to the list box and is selected.
3. If required, select a value from the *Signal* cyclic field.
4. Continue editing each data entry field until all the pertinent information is complete.
5. The last value in the data entry fields is recorded in the list box when the next operation is performed in the analysis form or when you click the *Clear/Add* button. (Other operations that terminate the editing operation include moving the cursor off the Choosing Analysis form, clicking either the *OK* or *Apply* button, or changing a value in a non-related field.)

To edit an existing tone,

1. First select the tone in the list box.

   The tone data appears in the data entry fields where you can edit it.

2. Modify a value in any one of the data entry fields. Values you cannot edit in the data entry fields (such as values specified in the schematic), are grayed out. Values originally specified in the schematic must be edited in the schematic.
3. The last modified value in the data entry fields is recorded in the list box when the next operation is performed in the analysis form or when you click the *Clear/Add* button.

Input Source and Reference Side-Band (Pnoise)

Identifies the noise generator and the reference sidebands to use for the pnoise simulation.

The Reference Side-Band field specifies which conversion gain to use when the SpectreRF simulation computes the input-referred noise, noise factor, and noise figure.

The designated reference sidebands, as well as the sideband zero, are included in the pool of sidebands used in noise calculations. For example, for the *ne600* test schematic,

- If `refsideband=-1` and `sidebands=[-2]`, then `pnoise` computes contributions from sidebands -2, -1, and 0.
- If `refsideband=-2` and `sidebands=[-2]`, then Spectre computes contributions from sidebands -2 and 0.

See "Sidebands (PAC, Pnoise, and PXF)" on page 154 for information on selecting sidebands.

The output total noise is different for the two simulation setups. The input-referred noise, noise factor, and noise figure are also different. `Pnoise` analysis internally includes the refsideband as a contribution to the total noise. This inclusion is not reflected in the netlist.

Input Source

Choices for Input Source are: *voltage*, *current*, *port*, or *none*.

Voltage

- Select *voltage* from the *Input Source* cyclic field.

- Either type the name of the noise voltage generator in the *Input Voltage Source* field or click on *Select* and then click on the generator in the schematic.
- For Reference Side-Band, type an integer value in the field.
  - `0` for amplifiers and filters
  - `-K` for down converters
  - `+K` for up converters

  where `K` is the mixing harmonic.

Current

- Select *current* from the *Input Source* cyclic field.
- Either type the name of a noise current generator in the *Input Current Source* field or click on *Select* and then click on the generator in the schematic.
- For Reference Side-Band, type an integer value in the field.
  - `0` for amplifiers and filters
  - `-K` for down converters
  - `+K` for up converters

  where `K` is the mixing harmonic.

Port

- Select *port* from the *Input Source* cyclic field.

- Either type the name of a port in the *Input Port Source* field or click on *Select* and then click on the port in the schematic.
- For Reference Side-Band, type an integer value in the field.
  - `0` for amplifiers and filters
  - `-K` for down converters
  - `+K` for up converters

    where `K` is the mixing harmonic.

  When you select *port*, the analysis computes the noise voltage across the port and subtracts the contribution of this port in noise figure calculations.

None

➤ Select *none* from the *Input Source* cyclic field.

When you select none, there are certain calculations you cannot perform. For example, input referred noise.

Reference Side-Band

Reference Side-Band specifies which conversion gain to use when the SpectreRF simulation computes the input-referred noise, noise factor, and noise figure.

Choices are *Enter in field* or *Select from list*.

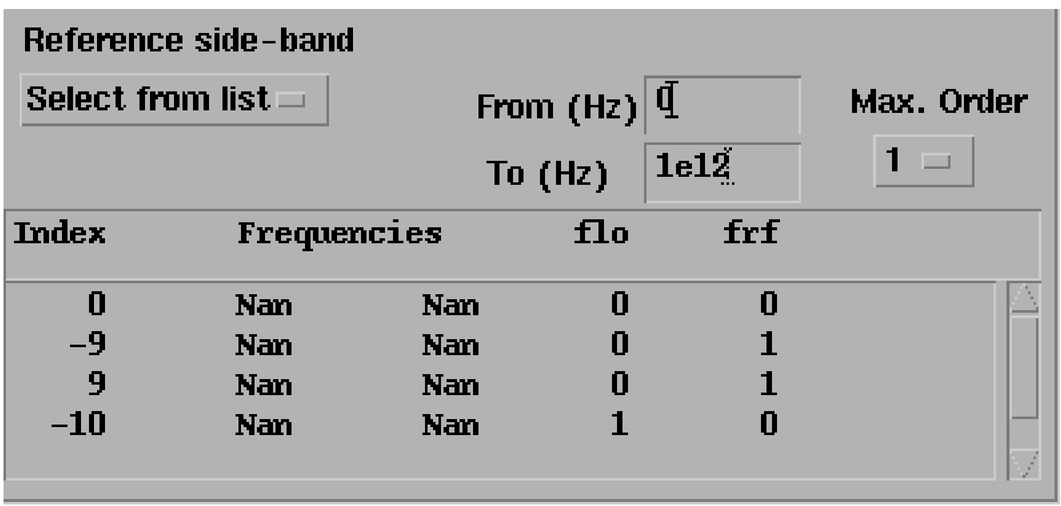
Enter in Field

- Select *Enter in field* from the *Reference side-band* cyclic field.
- Type an integer value in the field.
  - `0` for amplifiers and filters
  - `-K` for down converters
  - `+K` for up converters

  where `K` is the mixing harmonic.

Select From List

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.



Depending on the selection in the Sweeptype cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands:

- First, in the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range. The sideband frequencies displayed in the list box are within this range of frequencies.
- Then, from the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands. If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

To select from the listed sidebands:

Click on a sideband in the list box to select it. Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the Control key down while you click on the individual sidebands. (Deselect a sideband by clicking on a selected sideband while you hold the Control key down.)

Input Source and Reference Side-Band (QPnoise)

Identifies the noise generator and the reference sidebands vector for the qpnoise simulation.

Choices for *Input Source* are: *voltage, current, port,* or *none*.

The *Reference Side-Band* field specifies which conversion gain to use when the SpectreRF simulation computes the input-referred noise, noise factor, and noise figure.

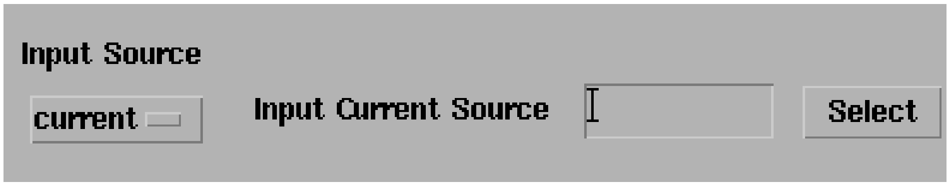
Important

For qpnoise analysis, the *Reference Side-Band* field value is a vector. For example, `1 0 0`.

Voltage

- Select *voltage* from the *Input Source* cyclic field.
- Either type the name of the noise voltage generator in the *Input Voltage Source* field or click on *Select* and then click on the generator in the schematic.

Current



- Select *current* from the *Input Source* cyclic field.
- Either type the name of a noise current generator in the *Input Current Source* field or click on *Select* and then click on the generator in the schematic.

Port

- Select *port* from the *Input Source* cyclic field.
- Either type the name of a port in the *Input Port Source* field or click on *Select* and then click on the port in the schematic.

None

➤ Select *none* from the *Input Source* cyclic field.

When you select none, there are certain calculations you cannot perform. For example, input referred noise. *Reference Side-Band* is not available when you select *none* for *Input Source.*

Reference Side-Band

Reference Side-Band specifies which conversion gain to use when the SpectreRF simulation computes the input-referred noise, noise factor, and noise figure. *Reference Side-Band* is available with all *Input Source* choices except *none.*

Important

For `qpnoise` analysis, the *Reference Side-Band* field value is a vector. For example, `1 0 0`.

Choices are *Enter in field* or *Select from list*.

Enter in Field

Select *Enter in field* from the *Reference side-band* cyclic field and type a vector value into the field.

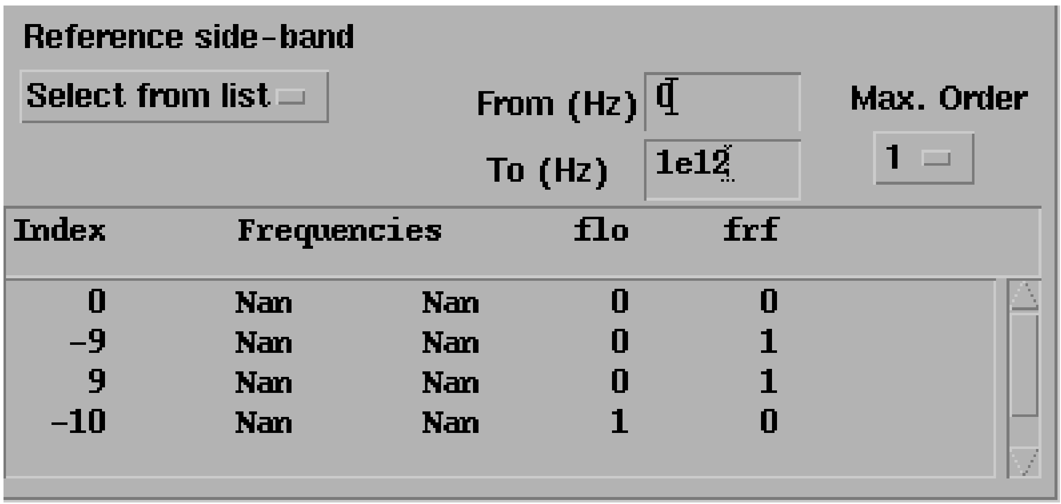
- When the input and output are at the same frequency, use the zero vector

  `[0 0 ...]`

- When you do not use the zero vector, the single sideband noise figure is computed.

Select From List

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.



Depending on the selection in the <u>*Sweeptype*</u> cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands:

- First, in the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range. The sideband frequencies displayed in the list box are within this range of frequencies.
- Then, from the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands. If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

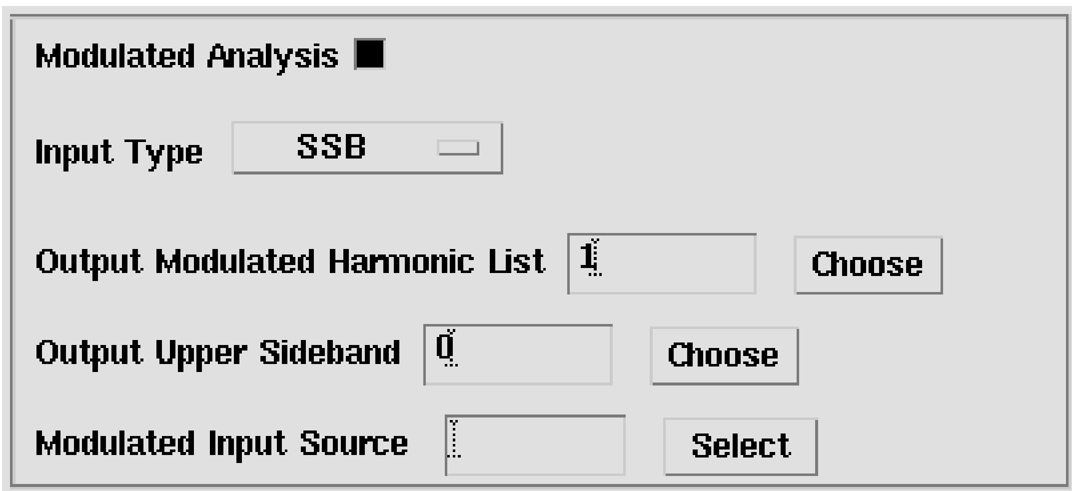
To select from the listed sidebands:

Click on a sideband in the list box to select it. Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the Control key down while you click on the individual sidebands. (Deselect a sideband by clicking on a selected sideband while you hold the Control key down.)

Modulated Analysis (PAC and PXF)

Measures AM and PM small signal effects for `pac` and `pxf` analyses. When you highlight *Modulated Analysis*, the form changes to let you enter more information.

The *Modulated Analysis* fields for the `pac` analysis include the following.

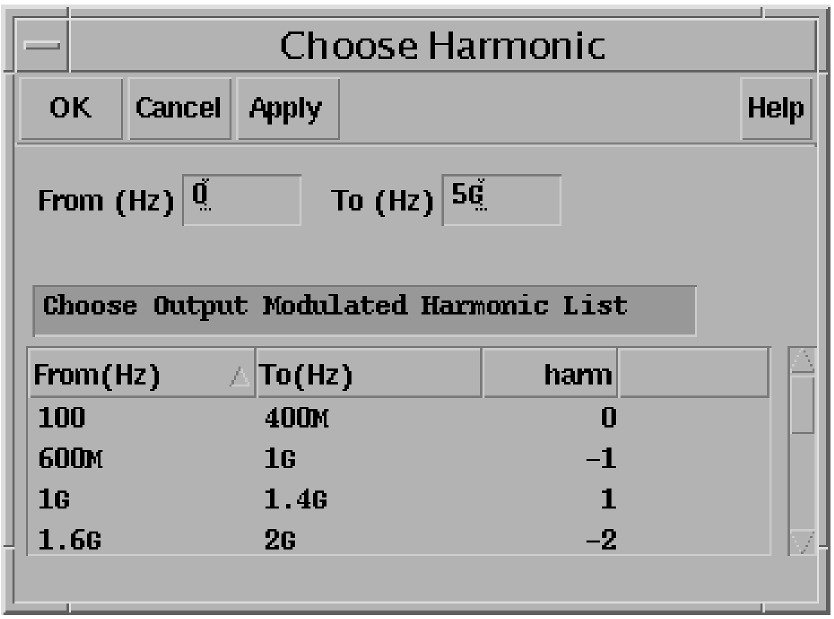


Modulated Analysis for PAC Terms and Data Entry Fields

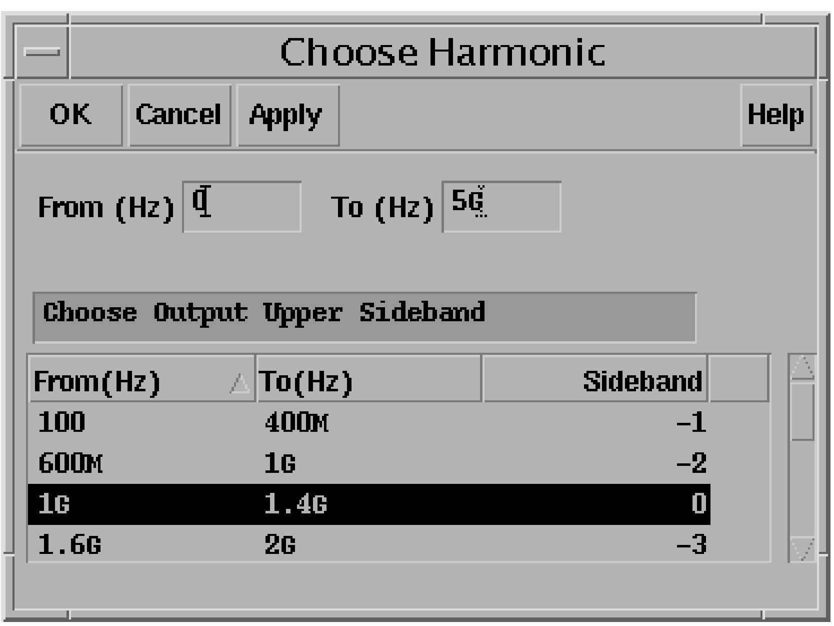
- **Input Type** – This cyclic field displays one of two values: *SSB* and *SSB/AM/PM.*
- **Output Modulated Harmonic List** – Lists the harmonic indexes.
- **Output Upper Sideband** – (Displays when you choose the *Input Type SSB*.) Click *Choose* to display the Choose Harmonic Pop Up.
- **Input Modulated Harmonic** – (Displays when you choose the *Input Type SSB/AM/PM*.) Click *Choose* to display the Choose Harmonic Pop Up.
- **Modulated Input Source** – Displays the instance name of the source in the schematic. Click *Select* and select the source in the schematic. The source name displays in the field.

Choose Harmonic Pop Up

Selects harmonics or sidebands for the analysis. Display the Choose Harmonics Pop Up from *Modulated Analysis* with the *Choose* button.



or



Choose Input/Output Modulated Harmonic List Box and Data Entry Fields

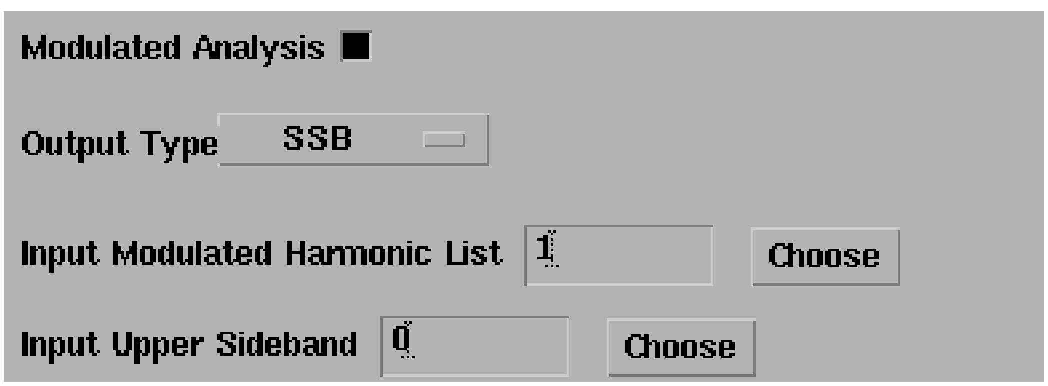
The *Choose Input* (or *Output*) *Modulated Harmonic* list box displays the harmonic indexes and associated frequency ranges to select from. Changing the values in the *From (Hz)* and *To(Hz)* fields, changes the harmonic indexes and frequency ranges displayed in the list box.

- **From (Hz) and To (Hz) fields -** The upper and lower bounds for the frequency range.
- **Harm** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency.

  For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.
- **From (Hz) -** The lower bound for the frequency range.
- **To(Hz) -** The upper bound for the frequency range.

Measures AM and PM small signal effects for `pac` and `pxf` analyses. When you highlight *Modulated Analysis*, the form changes to let you enter more information.

The Modulated Analysis fields for the pxf analysis include the following



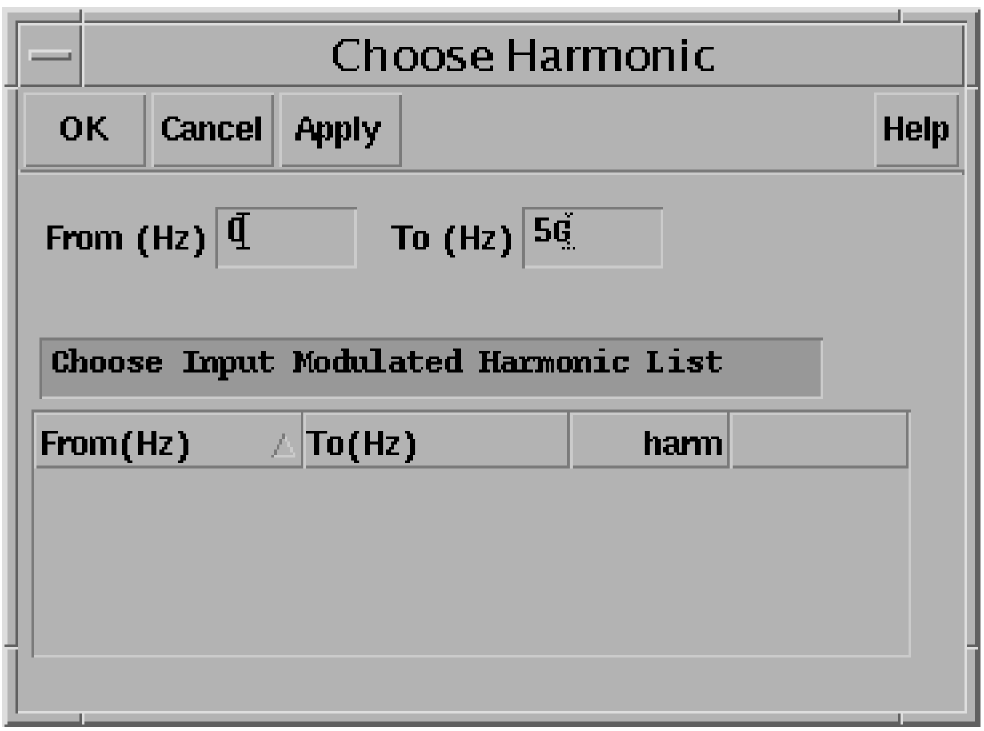
Modulated Analysis for PXF Terms and Data Entry Fields

- **Output Type** – This cyclic field displays one of two values: *SSB* and *SSB/AM/PM.*
- **Input Modulated Harmonic List** – Lists the harmonic indexes.

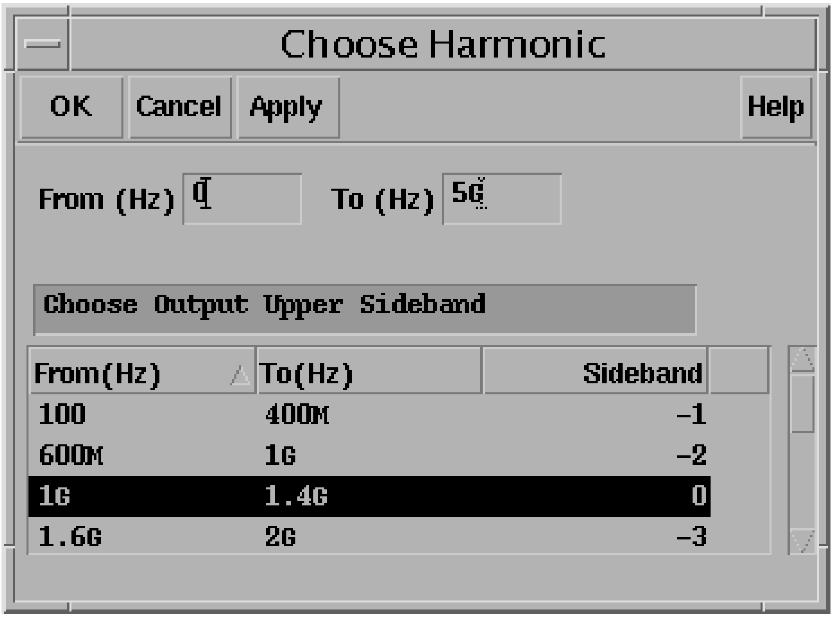
- **Input Upper Sideband** – (Displays when you choose the *Input Type SSB*.) Click *Choose* to display the Choose Harmonic Pop Up.
- **Output Modulated Harmonic** – (Displays when you choose the *Input Type SSB/AM/PM*.) Click *Choose* to display the Choose Harmonic Pop Up.

Choose Harmonic Pop Up

Selects harmonics or sidebands for the analysis. Display the Choose Harmonics Pop Up from *Modulated Analysis* with the *Choose* button.



or



Choose Input/Output Modulated Harmonic List Box and Data Entry Fields

The *Choose Input* (or *Output*) *Modulated Harmonic* list box displays the harmonic indexes and associated frequency ranges to select from. Changing the values in the *From (Hz)* and *To(Hz)* fields, changes the harmonic indexes and frequency ranges displayed in the list box.

- **From (Hz) and To(Hz) fields -** The upper and lower bounds for the frequency range.
- **Harm** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency.

  For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.
- **From (Hz) -** The lower bound for the frequency range associated with the harmonic.
- **To(Hz) -** The upper bound for the frequency range associated with the harmonic.

Choose Input Upper Sideband List Box and Data Entry Fields

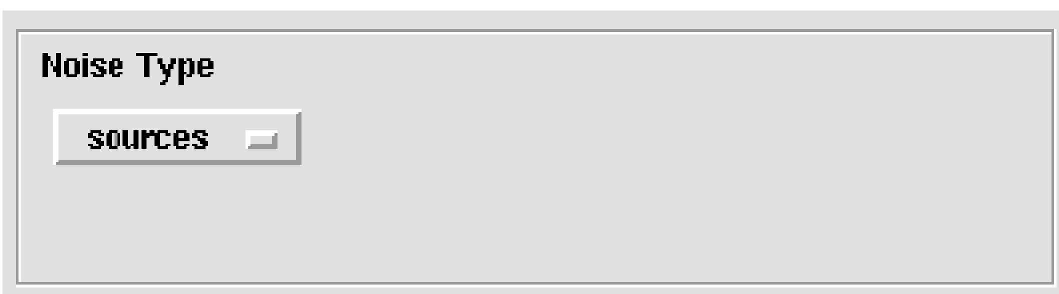
The *Choose Input Upper Sideband* list box displays the harmonic indexes and associated frequency ranges to select from. Changing the values in the *From (Hz)* and *To(Hz)* fields, changes the harmonic indexes and frequency ranges displayed in the list box.

- **From (Hz) and To(Hz) fields -** The upper and lower bounds for the frequency range.
- **Sideband** – Displays the sidebands, the integers which are the periodic small-signal output frequencies of interest.
- **From (Hz) -** The lower bound for the frequency range associated with the sideband.
- **To(Hz) -** The upper bound for the frequency range associated with the sideband.

Noise Type (Pnoise)

Specifies the type of noise to compute. Choices are: *sources, timedomain, correlations,* and *modulated*.

Sources



➤ Select *sources* from the *Noise Type* cyclic field.

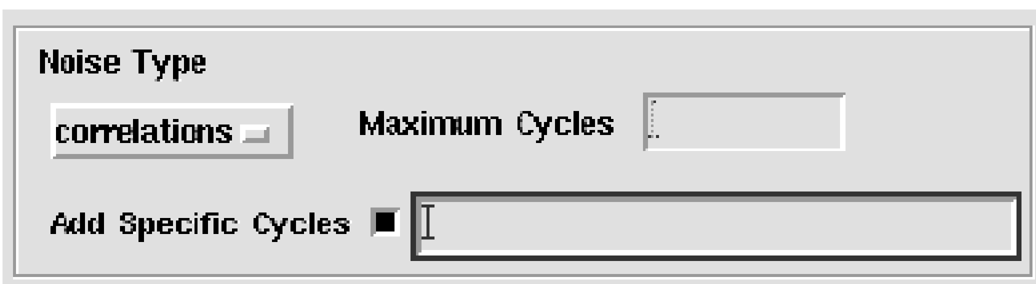
When you select *sources,* the simulation computes the total time-average noise at an output over a given frequency range. It computes the contribution of each noise source to the total noise at each frequency.

Timedomain

- Select *timedomain* from the *Noise Type* cyclic field.
- Type the noise skip count in the *Noise Skip Count* field.
- Optionally, highlight *Add Specific Points* and type one or more points.

Computes the time-varying instantaneous noise power in a circuit with periodically driven components.

Correlations



- Select *correlations* from the *Noise Type* cyclic field.
- Type the maximum number of cycles in the *Maximum Cycles* field.
- Optionally, highlight *Add Specific Cycles* and type one or more cycles.

Computes correlations between the noise at different ports of a multiport circuit. For example, you can compute the correlation between the noise at different outputs or the correlation between the noise at the input and output of a circuit that exhibits frequency conversion. You can extract equivalent noise sources from these calculations.

Modulated

- Select *modulated* from the *Noise Type* cyclic field.
- Select *relative* from the *Sweeptype* cyclic field.

Separates noise into AM and PM components. It also calculates USB and LSB noise.

Options

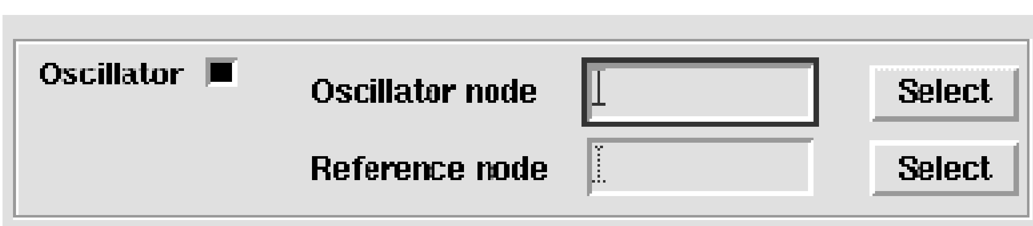
Opens the Options form.

Click the *Options* button to open the Options form for the highlighted analysis.

See Options Form for information on the Options forms.

Oscillator (PSS)

Specifies that the circuit is an oscillator. When you highlight Oscillator, the form changes to let you enter the *Oscillator node* and the *Reference node*. You can either type the appropriate node names or click on the appropriate *Select* and then click on the node in the schematic. If you leave the *Reference node* field empty, it defaults to gnd.

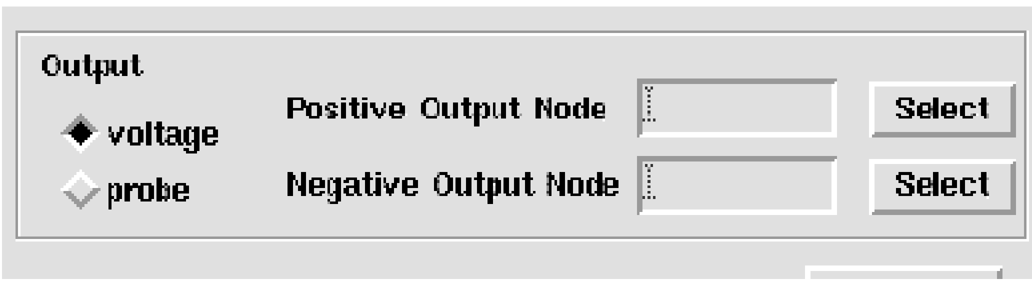


Output (PXF and QPXF)

Specifies the output transfer function for the `pxf` and `qpxf` analyses. Choices are *voltage* or *probe*.

Voltage

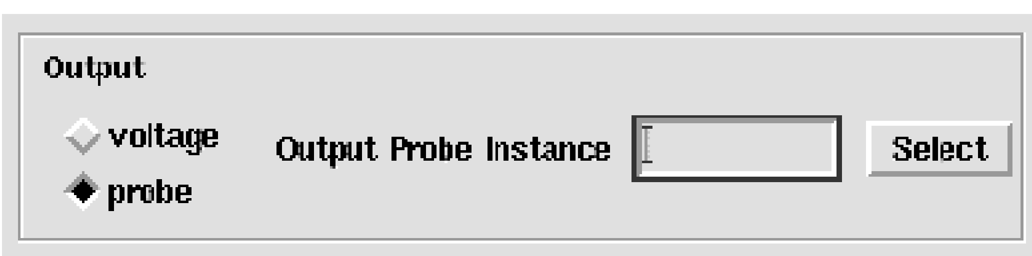
In `pxf` analysis, the voltage across the two nodes that you specify is the output for each transfer function.



Highlight *Voltage* and the form changes to let you specify a *Positive Output Node* and a *Negative Output Node*. You can specify this value either by typing the node name into the *Positive Output Node* or *Negative Output Node* field, or by clicking on the adjacent *Select* button and then clicking on the node in the schematic. If you leave the *Negative Output Node* field empty, it defaults to `gnd`.

Probe

In `pxf` analysis, the current through the point that you select is the output of each transfer function.



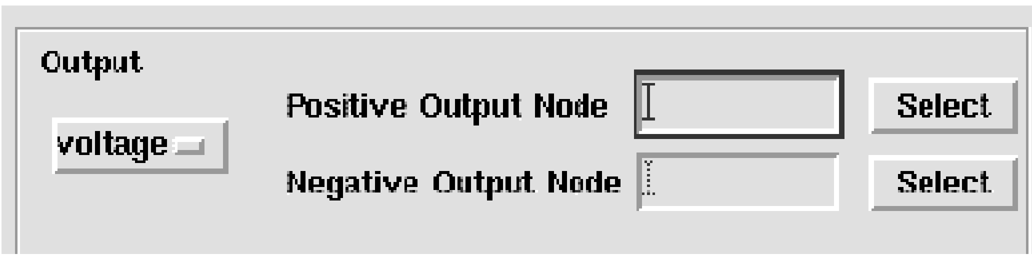
Highlight *Probe* and the form changes to let you specify an output voltage source. You can specify this value either by typing it into the *Output Probe Instance* field or by clicking the adjacent *Select* button and then clicking on the node in the schematic.

Output (Pnoise and QPnoise)

The *Output* cyclic field lets you specify the output for the `pnoise` and `qpnoise` analyses. Choices are *voltage* or *probe*.

Voltage

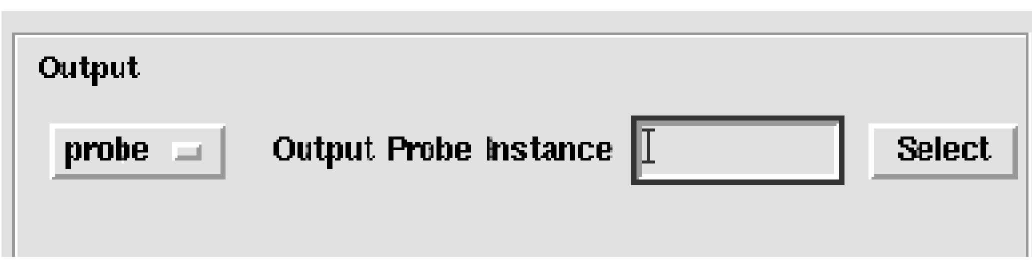
The `pnoise` or `qpnoise` analysis computes the noise voltage across the two nodes.



Select *Voltage* and the form changes to let you specify a *Positive Output Node* and a *Negative Output Node*. You can specify the nodes either by typing into the type-in field or by clicking on the adjacent *Select* button and then clicking on the appropriate node in the schematic. If you leave the *Negative Output Node* field empty, it defaults to `gnd`.

Probe

The `pnoise` or `qpnoise` analysis computes the noise voltage across the port. The noise contribution of the port is subtracted during the noise figure calculation.



Select *Probe* and the form changes to let you specify the *Output Probe Instance*. You can specify the node either by typing into the type-in field or by clicking on the adjacent *Select* button and then clicking on the appropriate node in the schematic.

Output Harmonics (PSS and Envlp)

Lets you select output harmonics. When you select from the *Output Harmonics* cyclic field, the form changes to let you specify appropriate data.

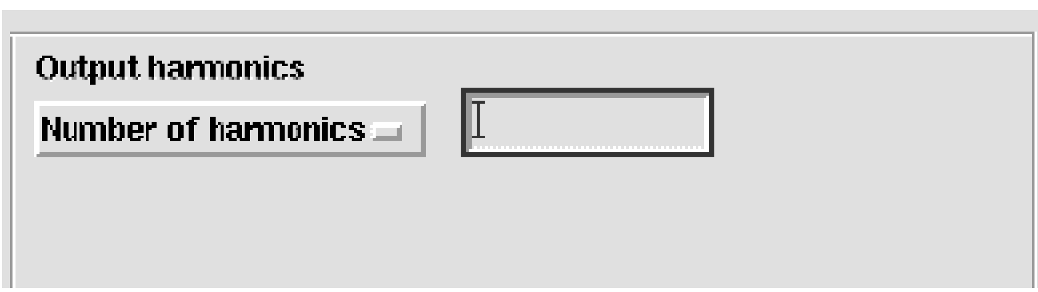
For pss analysis, choices are: *Number of Harmonics*, *Select from Range*, *Array of Coefficients*, *Array of Indices*.

For Envlp analysis, choices are: *Number of Harmonics*, and *Array of Indices*.

You can use the separate choices in the *Output Harmonics* cyclic field in combination. For example, if you add a harmonic using *Array of Coefficients*, the value you add appears in the *Select from Range* list box and as a currently active index field for the *Array of Indices*.

Number of Harmonics (PSS and Envlp)

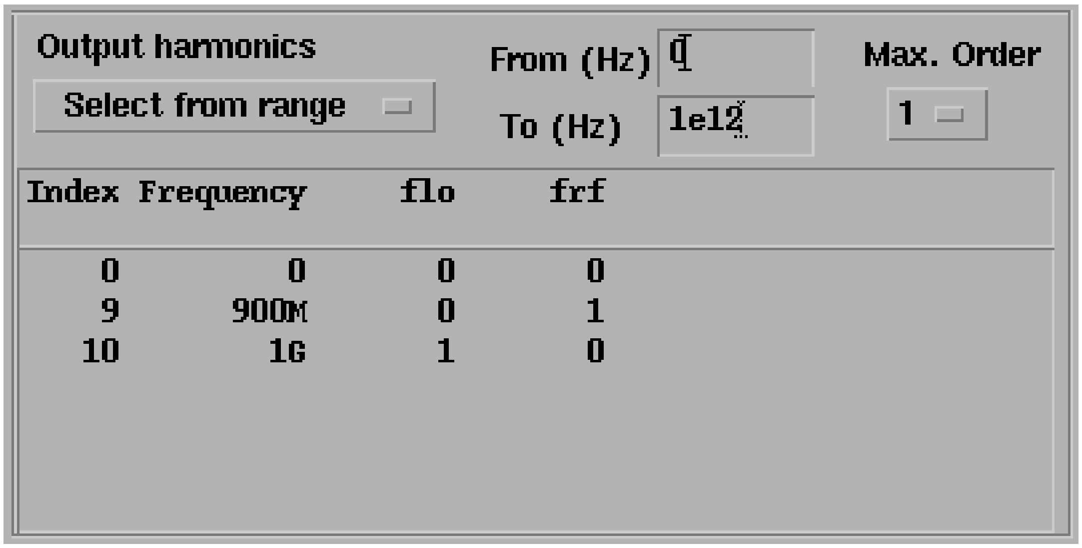
Specifies a single value for the number of output harmonics.



Select *Number of Harmonics* and the form changes to let you specify a single integer value. in the *Number of harmonics* field. Harmonics in this field are referred to the value of the fundamental. Type `0` into the *Number of harmonics* field to specify no harmonics.

Select from Range (PSS)

Lets you first enter a frequency range and then select harmonics from within this range.



Select *Select from Range* and the form changes to display a cyclic field and two data entry fields.

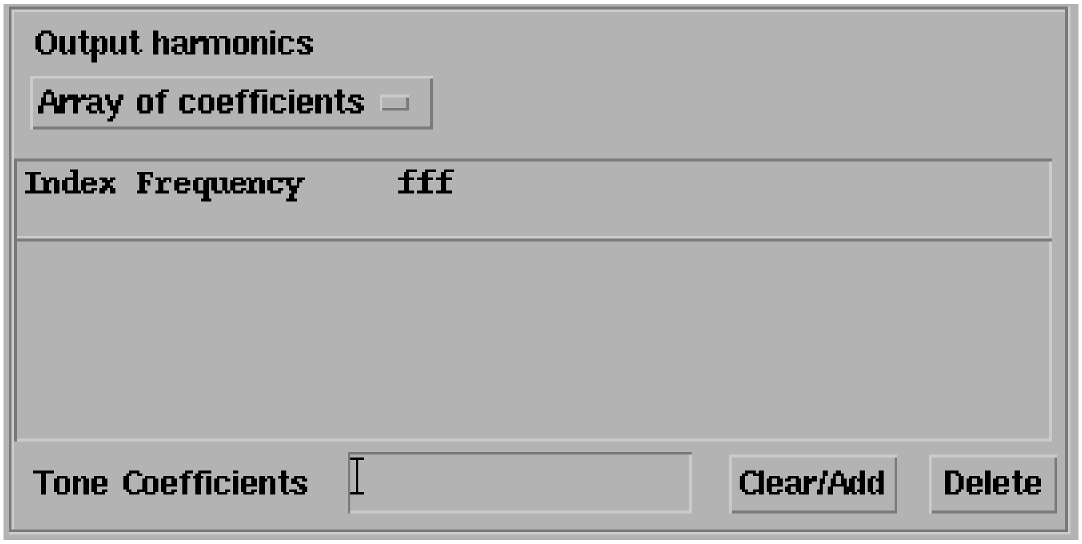
- First, in the *From (Hz)* and *To (Hz)* fields, type the lower and upper values for the frequency range.
- Second, from the *Max. Order* cyclic field, select the maximum order of harmonics that contribute to the output harmonics.

  The form changes to display harmonics matching these specifications in the list box.The first column is the index of a harmonic, the second column specifies its frequency, and the remaining columns specify tone coefficients for each fundamental tone that contributed to the listed harmonic.

  If, for example, you select 5 in the *Max. Order* cyclic field, the sum of the absolute values of the tone coefficients contributing to the listed output harmonics is less than or equal to five. Negative integers displayed in the list box represent the tone coefficients of harmonics below the fundamental and positive integers represent the tone coefficients of harmonics above the fundamental.
- Click on a harmonic in the list box to select it. Select adjacent harmonics by clicking and dragging with the mouse over the harmonics to select. Select harmonics that are not adjacent by holding the *Control* key down while you click on the individual harmonics. (Deselect a harmonic by holding the *Control* key down and clicking on it.)

Array of Coefficients (PSS)

Lets you specify output harmonics by entering their tone coefficients.



Select *Array of Coefficients* and the form changes to display a data entry field.

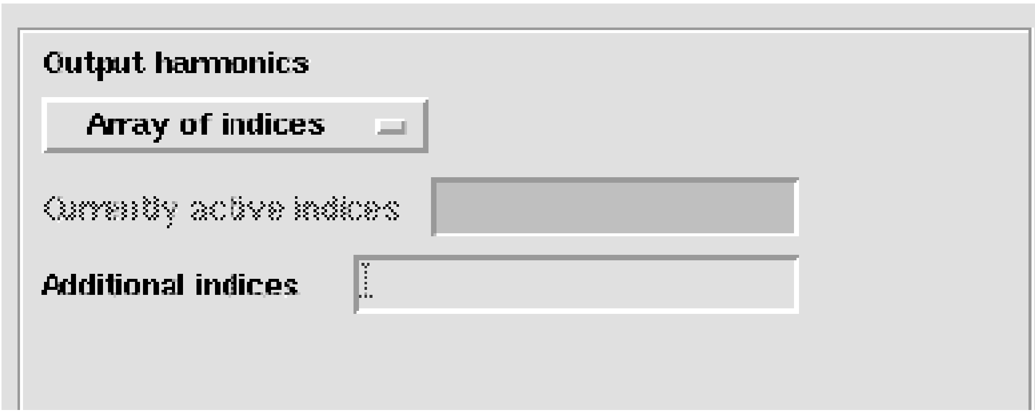
- Type tone coefficients separated by spaces in the *Tone Coefficients* field and click on *Clear/Add*.

  Values that appear in the list box are absolute values or relative to the fundamental, depending on the selection in the *Sweeptype* cyclic field.

- To delete a harmonic, select it in the list box and click on *Delete*.

Array of Indices (PSS and Envlp)

Lets you specify an array of harmonics by entering their indices.

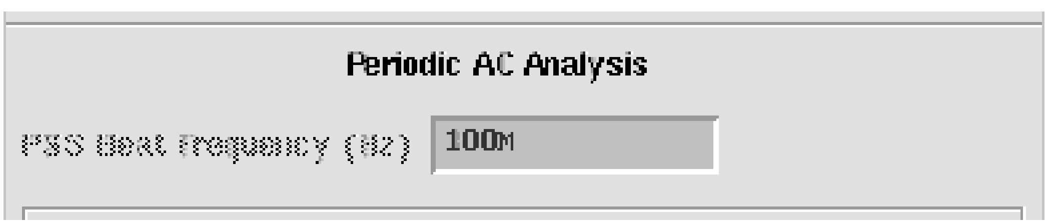


Select *Array of Indices* and the form changes to display a data entry field. Any currently selected indices appear in the *Currently active indices* field.

Type the additional harmonic indices separated by spaces and in any order in the *Additional Indices* type-in field.

PSS Beat Frequency (PAC, Pnoise, and PXF)

Displays the *Beat Frequency* for the initial PSS analysis.



This is a display-only field. You cannot edit this information here.

Save Initial Transient Results (PSS and QPSS)

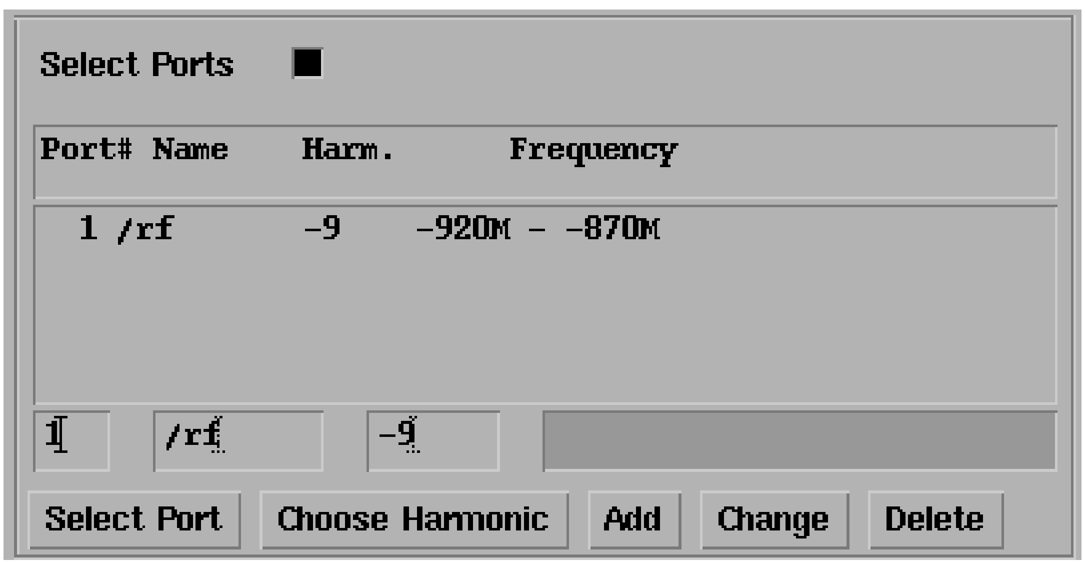
Saves the initial transient waveforms.

Highlight *yes* to save the initial transient waveforms.The default is *no*.

Select Ports (PSP and QPSP)

Specifies the active ports for psp and qpsp analyses.

The *Select Ports* fields include the following list box, data entry fields, and data entry buttons.



Select Ports List Box

The *Select Ports* list box displays information about all selected ports. Ports in the list box are arranged numerically by the port numbers (*Port#*) that you assign.

Select Ports List Box Terms and Data Entry Fields

The list box displays the ports that have already been specified for PSP or QPSP analysis. Enter data for new ports using the data entry fields and buttons below the list box. You can enter new ports or edit values for existing ports. Notice that the frequency field is grayed out and you cannot edit frequency values.

- **Port#** – Displays the port number assigned to the port. In the list box, ports are numbered sequentially.
- **Name** – Displays the name of the port in the schematic. Use the *Select Port* button to select a port from the schematic.The name of the selected port in the schematic displays in the editable field.
- **Harm.** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency. Use the *Choose Harmonic* button to display a list of harmonics and frequencies to select from. The selected harmonic index displays in the editable field. The associated frequency range also displays.

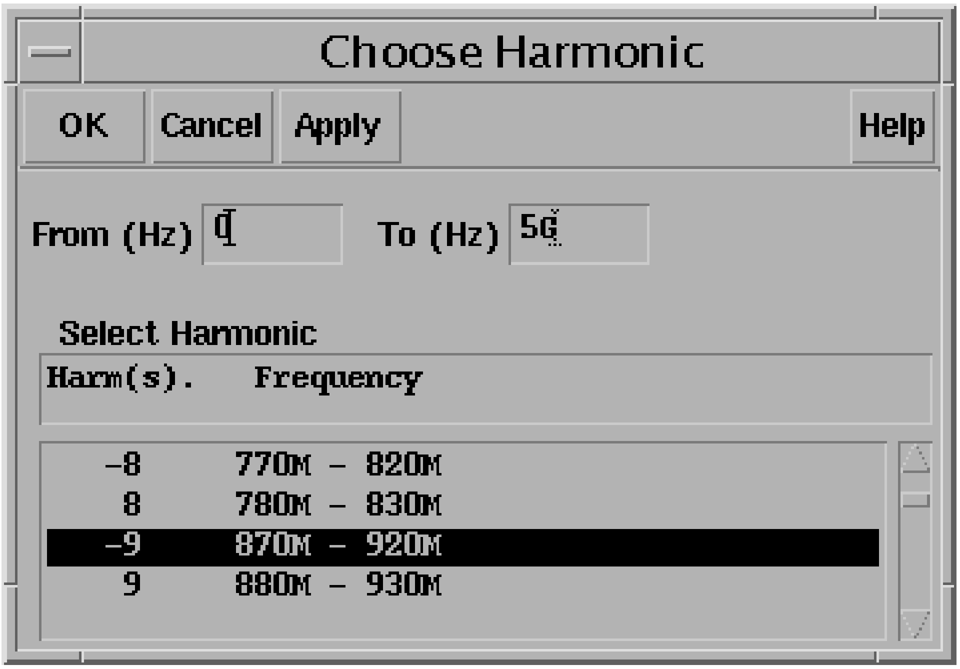
  For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.
- **Frequency** – Displays a range of frequency values associated with each harmonic index. Use the *Choose Harmonic* button to display a list of harmonics and frequencies to select from. The frequency range associated with the selected harmonic index value displays in the field. Notice that the Frequency field is grayed out and you cannot edit frequency range values.

Select Ports Data Entry Buttons

- **Select Port** – Prompts you to select a port from the schematic. Displays the port's name from the schematic in the editable field.
- **Choose Harmonic** – Opens the Choose Harmonic Pop UP where you can select a harmonic for the port.
- **Add** – Adds a port to the list box using the information in the data entry fields.
- **Change** – Adds a modified port to the list box using the modified information in the data entry fields. Highlight a port in the list box to move it's information to the data entry fields.
- **Delete** – Deletes a highlighted port from the list box.

Choose Harmonic Pop Up

Selects a harmonic for a port. Open the Choose Harmonic Pop UP from *Select Ports*.



Choose Harmonic List Box and Data Entry Fields

The *Select Harmonic* list box displays the harmonic indexes and associated frequency ranges to select from for the ports that have already been specified for PSP or QPSP analysis. Changing the values in the *From (Hz)*, *To(Hz)* and *Max. Order* fields, changes the harmonic indexes and frequency ranges displayed in the *Select Harmonic* list box.

- **From (Hz) and To(Hz) fields -** The upper and lower bounds for the frequency range.
- **Max. Order** - For QPSP analysis only. Displays the maximum order of harmonics that contribute to the harmonics.
- **Harm(s).** – Displays the harmonic index, the integer which is multiplied by the fundamental to calculate the harmonic frequency.

  For PSS analysis of port1 named RF with an harmonic index of 1, given the PSS fundamental of 900 MHz, port1 is analyzed from 901 MHz to 1000 MHz. For QPSP analysis, the computation is more complicated because there are more fundamentals and the harmonic specification is a vector of indexes.

- **Frequency** – Displays a range of frequency values associated with each harmonic index.

Using the Select Ports Data Entry Fields and Buttons

Use the data entry fields below the list box to edit the information in the *Select Ports* list box or to add new ports.

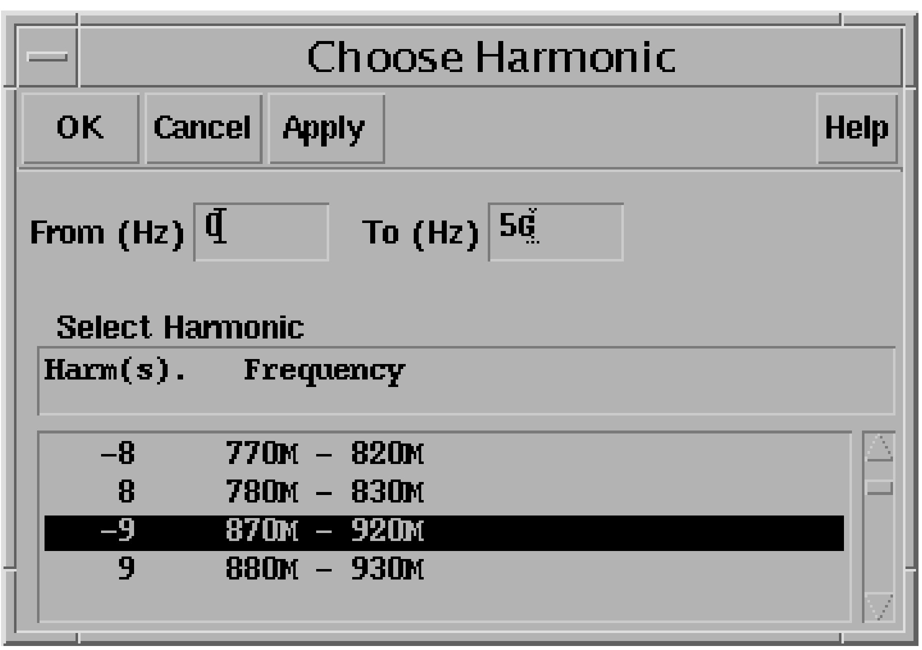
To specify a new port,

1. Type an integer in the first field, the *Port#* field. It is directly above the *Select Port* button.
2. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

   `Select source...`
3. In the Schematic window, click on an appropriate port, for example `/rf`.

   `/rf` appears in the *Name* field.
4. Click *Choose Harmonic.*

   The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rf* port.



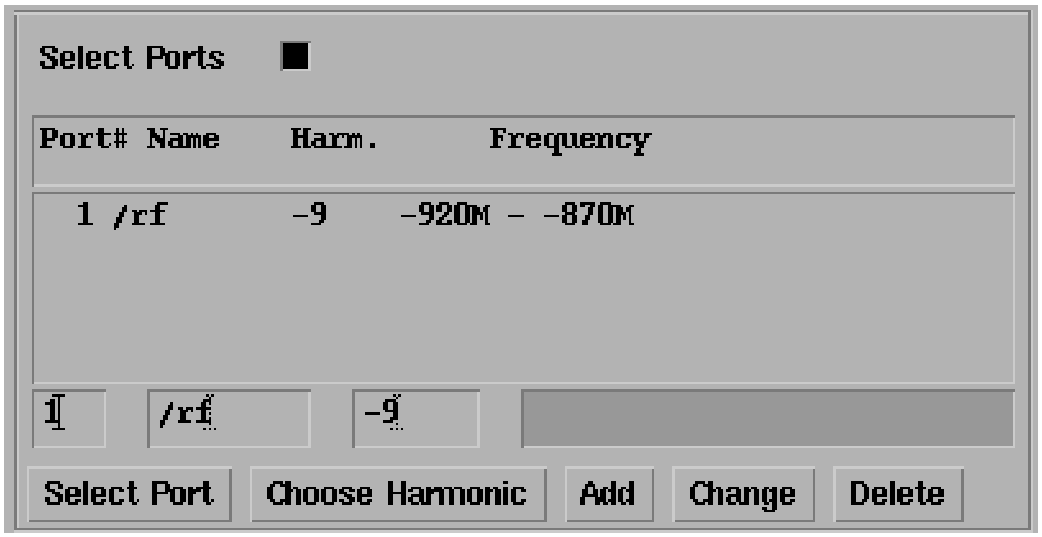
5. In the Choose Harmonic form, scroll through the list and highlight a harmonic to select it, for example `-9`.

6. Click *OK*.

   The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, `-9` displays in the *Harm.* field.

7. In the *Select Ports* area, click *Add*.

   Information for the input port displays in the *Select Ports* list box.



To edit an existing port,

1. Highlight the port in the list box.

   The port data appears in the data entry fields where you can edit it.

2. Modify a value in any one of the data entry fields. Values you cannot edit in the data entry fields (such as frequency range values), are grayed out.

3. Click *Change*.

   The modified port data replaces the port in the list box.

To delete an existing port,

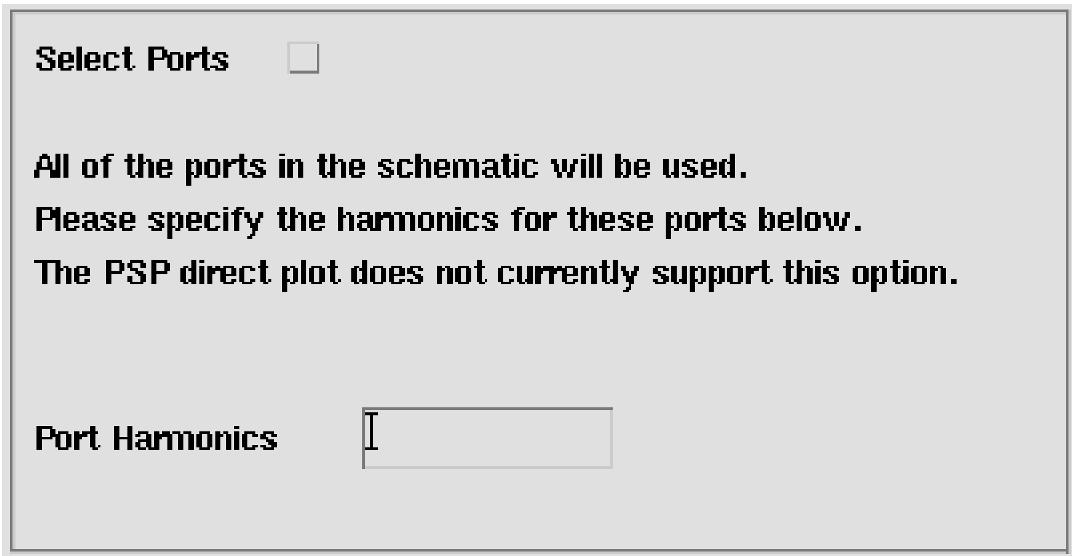
1. Highlight the port in the list box.

   The port data appears in the data entry fields.

2. Click *Delete*.

   The port is removed from the list box.

To use all ports in the schematic,



- Deselect *Select Ports*.
- In the *Port Harmonics* field, type the harmonics for these ports separated by spaces. (psp direct plot does not currently support this option.)

Sidebands (PAC, Pnoise, and PXF)

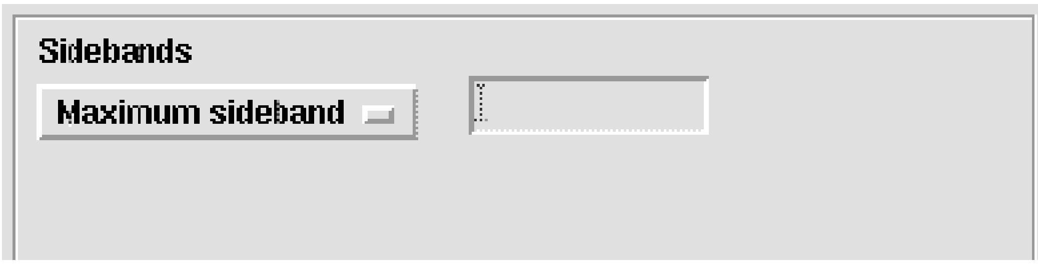
Lets you select the set of periodic small-signal output frequencies of interest. When you select from the *Sidebands* cyclic field, the form changes to let you specify appropriate data.

Choices are: *Maximum sideband*, *Select from Range*, *Array of Coefficients*, *Array of Indices*.

You can use the choices in the *Sidebands* cyclic field in combination. For example, if you add a sideband using *Array of Coefficients*, the sideband value you added appears in the *Select from Range* list box and as a *Currently active index* for the *Array of Indices*.

Maximum sideband

Specifies the number of frequency conversion terms (values of k) to take into account.



Prompts you for a *Maximum sideband* value and automatically generates a sideband array of the form:

`[–maximum sideband . . . 0 . . . +maximum sideband]`

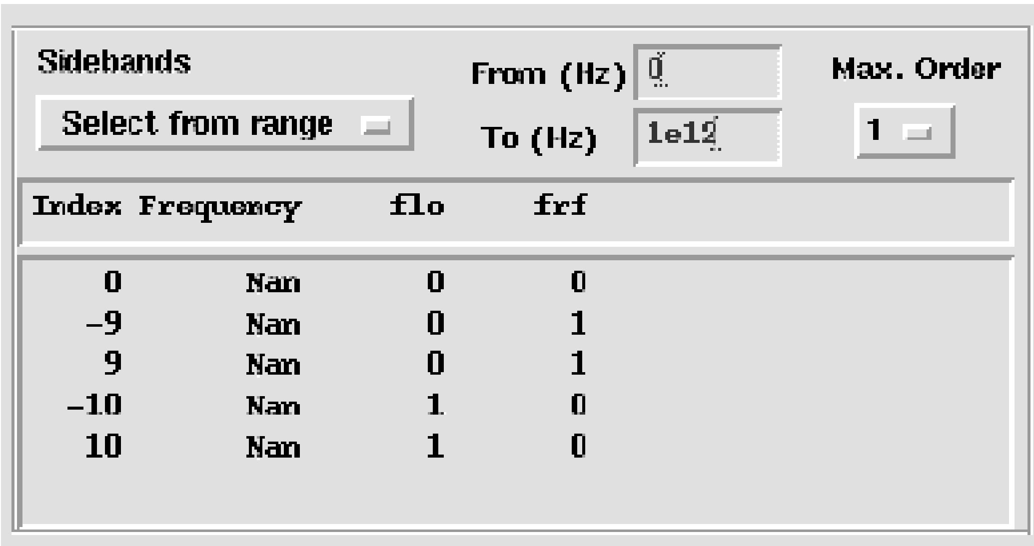
Begin by setting *Maximum sideband* to 7, the default value. Then increase the *Maximum sideband* value while observing the effect on output noise. If output noise changes, there is significant frequency conversion for values of k greater than 7. Continue to increase the *Maximum sideband* value until the output noise stops changing.

Important

When you set *Maximum sideband* to zero, the reported output noise will include no frequency conversion terms. For a fundamental oscillator, *Maximum sideband* must be at least 1 to see flicker noise upconversion. In general, small values for *Maximum sideband* are not recommended.

Select from range

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.



Depending on the selection in the _Sweeptype_ cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands:

- First, in the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range. The sideband frequencies displayed in the list box are within this range of frequencies.
- Then, from the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands. If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

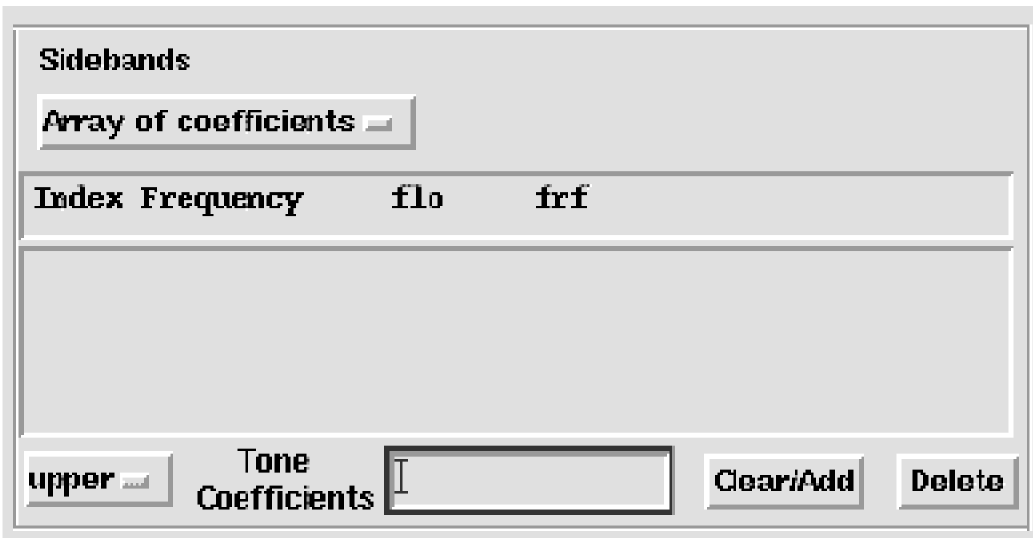
In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

To select from the listed sidebands:

Click on a sideband in the list box to select it. Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the Control key down while you click on the individual sidebands. (Deselect a sideband by clicking on a selected sideband while you hold the Control key down.)

Array of Coefficients

Lets you specify sidebands by typing their tone coefficients.



Lets you specify sidebands by typing tone coefficients, separated by spaces, in the *Tone Coefficients* type-in field. Click *Clear/Add* to add the tone coefficient to the list box.

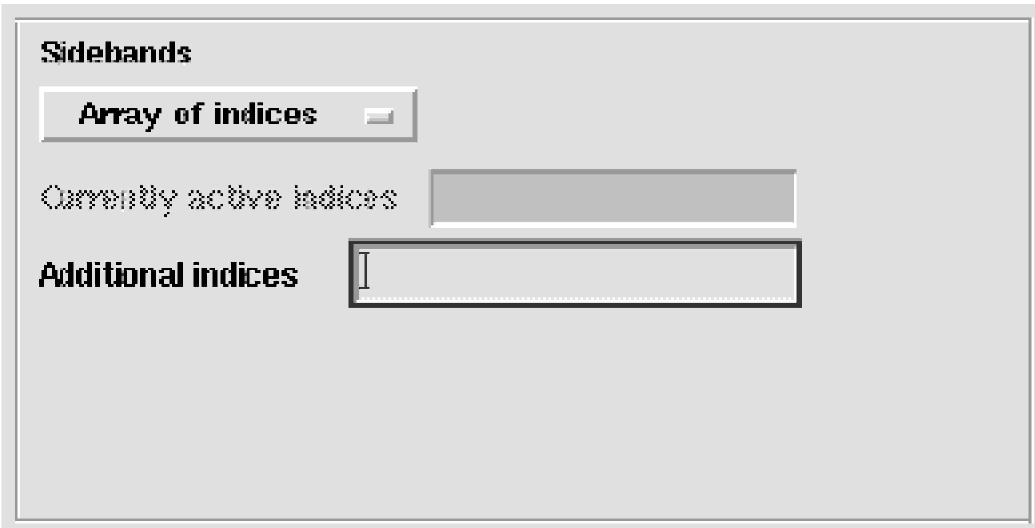
Values that appear in the list box are *absolute* values or are *relative* to the fundamental, depending on the selection in the *Sweeptype* cyclic field.

Place the specified sideband above or below the fundamental using the *upper* cyclic field.

To delete a sideband, select it in the list box and click on *Delete*.

Array of Indices

Lets you specify an array of sidebands by entering their indices.



Lets you specify an array of sidebands by typing their indices in the *Additional Indices* field. When you select this choice, currently selected indices appear in the *Currently active indices* field. You type the additional sideband indices you want in any order and separate these indices with spaces.

Sidebands (QPAC, QPnoise, and QPXF)

Lets you select the set of quasi-periodic small-signal output frequencies of interest for the `qpac`, `qpnoise`, or `qpxf` analysis.

Important

For the quasi-periodic analyses, sidebands are vectors specified with the `sidevec` and `clockmaxharm` parameters.

When you select from the *Sidebands* cyclic field, the form changes to let you specify appropriate data. Choices are: *Maximum clock order*, *Select from Range*, and *Array of Coefficients*.

You can use the choices in the *Sidebands* cyclic field in combination. For example, if you add a sideband using *Array of Coefficients*, the sideband value you added appears in the *Select from Range* list box.

Maximum clock order

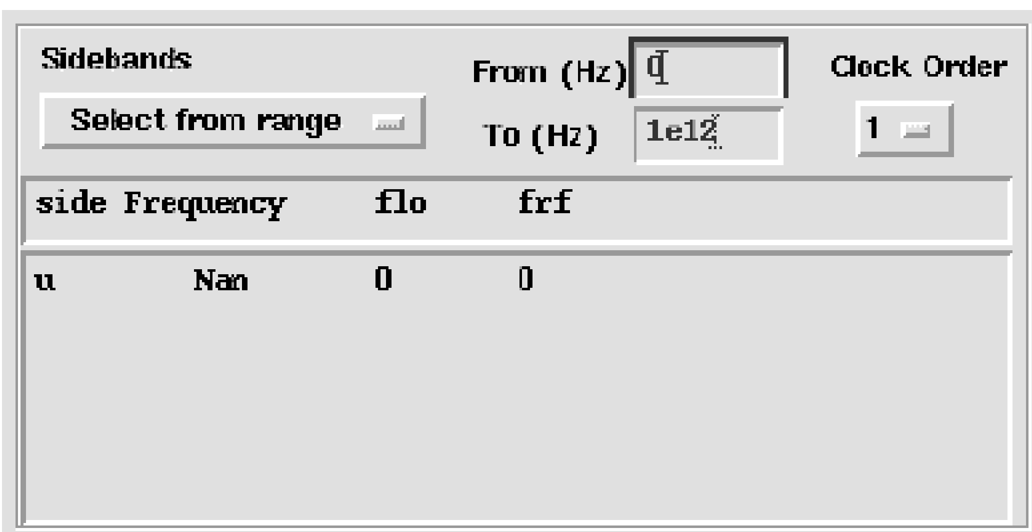
Specifies the largest sideband value and generates the sideband array.

Prompts you for a maximum sideband value and automatically generates a sideband array of the form:

```
[-max. sideband . . . 0 . . . + max. sideband]
```

Select from range

Lets you first enter a frequency range and then select sidebands from within this range of sideband values.



Depending on the selection in the <u>*Sweeptype*</u> cyclic field, both the values in the list box and the values you specify are either *absolute* values or they are *relative* to the fundamental.

To specify the listed sidebands:

- First, in the *From (Hz)* and *To (Hz)* type-in fields, type the lower and upper values for the frequency range. The sideband frequencies displayed in the list box are within this range of frequencies.
- Then, from the *Max. Order* cyclic field select the maximum order of harmonics that contribute to the sidebands. If, for example, you select 3 as the *Max. Order* value, the sum of the *absolute* values of the tone coefficients contributing to the sidebands in the list box must be less than or equal to three.

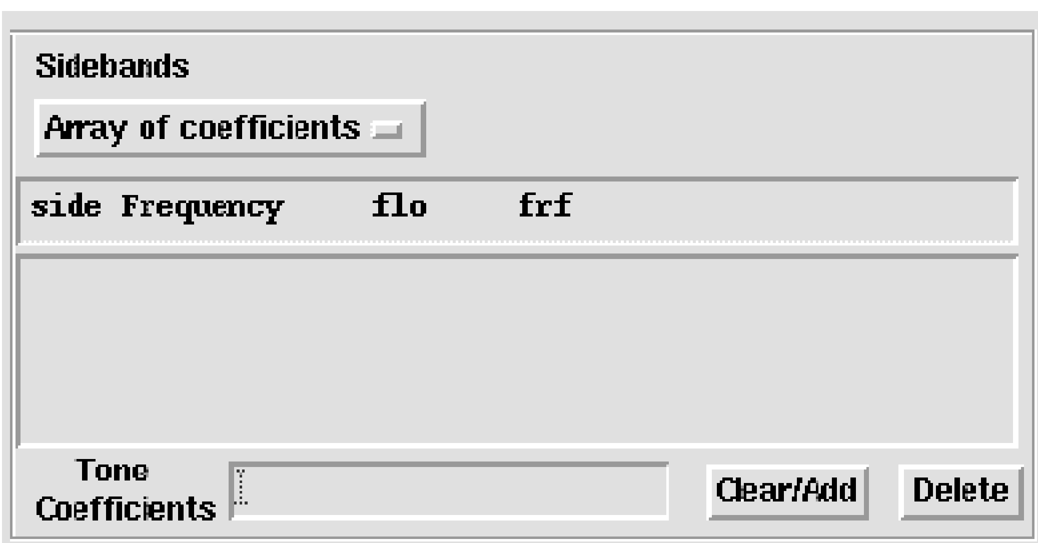
In the list box, the first column is the index of a sideband, the second and third columns list the frequency range of the sideband. The remaining columns list tone coefficients for each fundamental tone that contributed to the listed sideband.

To select from the listed sidebands:

Click on a sideband in the list box to select it. Select adjacent sidebands by clicking and dragging with the mouse over the sidebands to select. Select sidebands that are not adjacent by holding the Control key down while you click on the individual sidebands. (Deselect a sideband by clicking on a selected sideband while you hold the Control key down.)

Array of Coefficients

Lets you specify sidebands by typing their tone coefficients.



Lets you specify sidebands by typing tone coefficients, separated by spaces, in the *Tone Coefficients* type-in field. Click *Clear/Add* to add the tone coefficient to the list box.

Values that appear in the list box are *absolute* values or are *relative* to the fundamental, depending on the selection in the Sweeptype cyclic field.

Place the specified sideband above or below the fundamental using the *upper* cyclic field.

To delete a sideband, select it in the list box and click on *Delete*.

Start ACPR Wizard (Envlp)

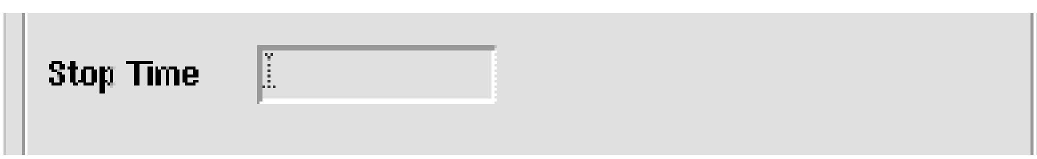
Opens the ACPR Wizard form. The ACPR Wizard helps you through the complex process of measuring ACPR (Adjacent Channel Power Ratio) and PSD (Power Spectral Density).

See“The ACPR Wizard” on page 211 for related information.

You can also open the ACPR Wizard from the Simulation window by choosing *Tools – RF – Wizards – ACPR.*

Stop Time (Envlp)

Specifies the end point in an Envelope Following Analysis.



Type a time value in the *Stop Time* field. Make the time interval long enough to let slow signals complete at least one cycle. See “Clock Name and Select Clock Name Button (Envlp)” on page 117 for related information.

Sweep (PSS and QPSS)

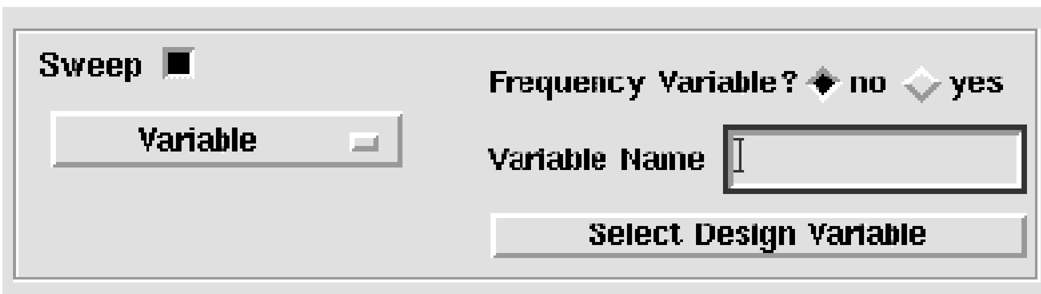
Specifies how a sweep is performed. Choices are: *Variable*, *Temperature*, *Component Param*, and *Model Param*.

Important

When you activate *Sweep* on the pss or qpss analysis form, the *Frequency Sweep Range* on the small-signal analysis forms is restricted to a single point.

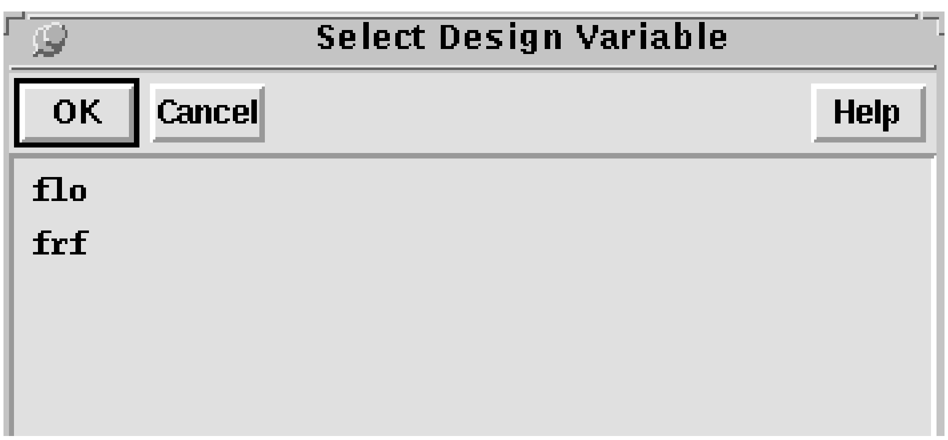
Variable

Sweeps a design variable.



Select *Variable* from the cyclic field and the form changes to accept data for the variable sweep.

- To select the variable to sweep, click *Select Design Variable* to display the Select Design Variable form.



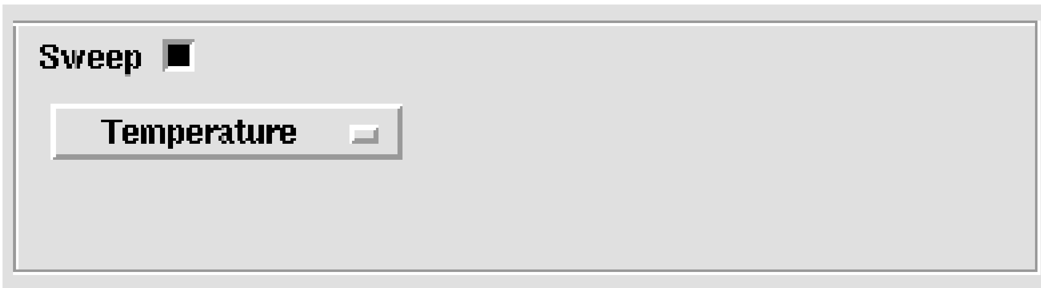
  - ❑ In the Select Design Variable form, select a variable and click *OK*.

    The variable name appears in the *Variable Name* field. (You can also simply type the name in the *Variable Name* field.)

- If the selected variable is a frequency variable, highlight *yes* for *Frequency Variable*.

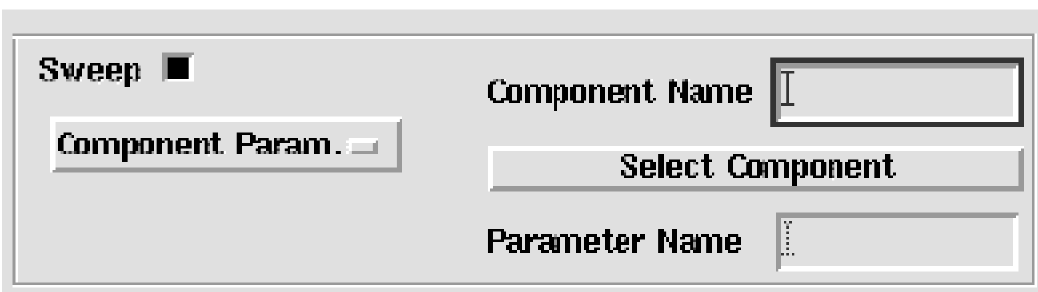
Temperature

Collects temperature data during the sweep.



Component Param

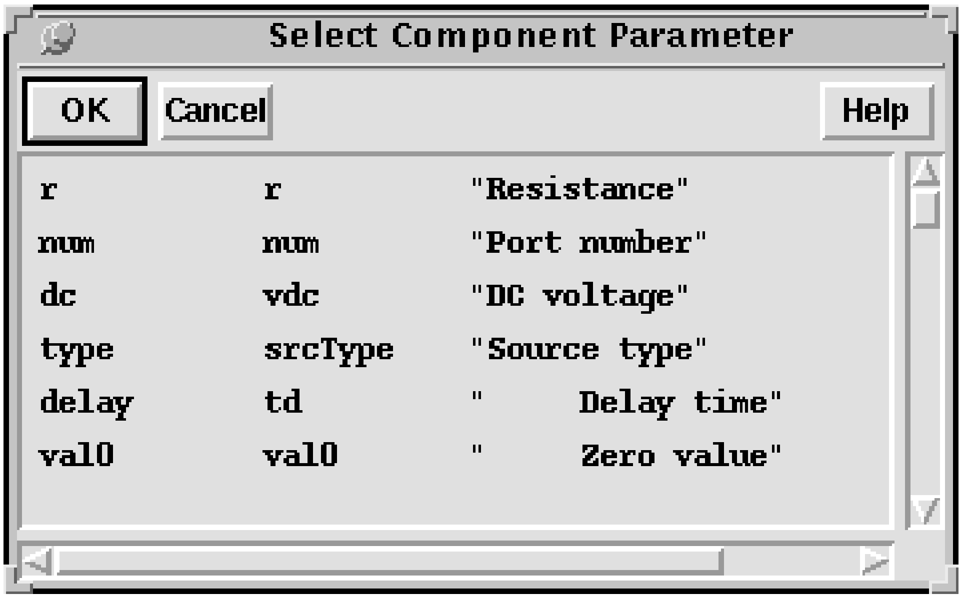
Sweeps a component parameter.



Select *Component Param* and the form changes to accept data for the component parameter sweep.

- To select the component, click *Select Component* and click on the component in the schematic.

After you click on the component, the Select Component Parameter form appears.

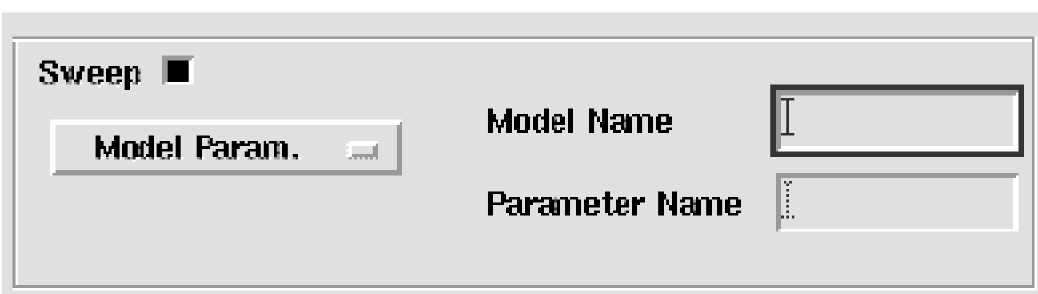


- ❑ You select a parameter in this form and then click *OK*.

  Both the component and parameter names appear in their respective fields. (You can also simply type the component and parameter names in their respective fields.)

- ■ **Model Param**

Sweeps a model parameter.



Select *Model Param* and the form changes to accept data for the model parameter sweep.

You type the model name and the parameter name in their respective type-in fields.

Sweep Range, Sweep Type, and Add Specific Points (PSS and QPSS)

Defines the analysis sweep range, the type of sweep, and any additional individual sweep points for the large-signal `pss` analysis or the medium-signal `qpss` analysis that precedes a small-signal analysis

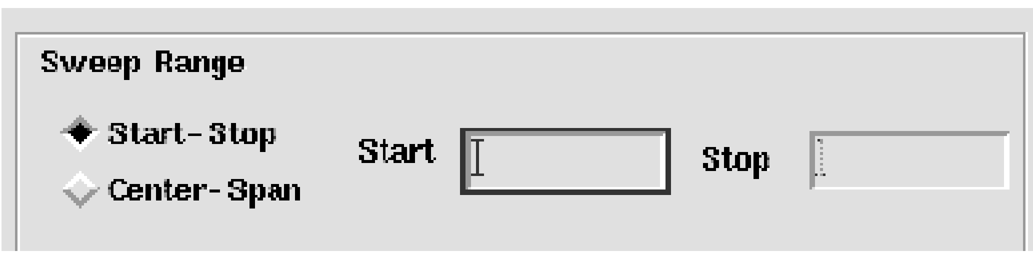
When you highlight *Sweep*, the *Sweep Range*, *Sweep Type*, and *Add Specific Points* fields open below the *Sweep* fields.

Sweep Range

Specifies the bounds for the sweep either as beginning and ending points or as a center point and a span.

Start – Stop

Defines the beginning and ending points for the sweep.



- Highlight *Start-Stop*.

  The form changes to let you type the start and stop points.
- Type the initial point for the sweep in the *Start* field.
- Type the final point in the *Stop* field.

  The *Start* and *Stop* sweep values can be frequencies, periods, or design variable values that correspond to your selection in the *Sweep* cyclic field.

Center – Span

Defines the center point for the sweep and its span.

- Highlight *Center-Span*.

  The form changes to let you type the center point and span.

- Type the midpoint for the sweep in the *Center* field.
- Type the span in the *Stop* field.

Sweep Type

Specifies whether the sweep is linear or logarithmic.

Linear

Specifies a linear sweep.

- Highlight *Linear*.

  The form changes to let you type either the step size or the total number of points (steps).

  - ❑ Highlight *Step Size* and type the size of each step in the field.

    or

  - ❑ Highlight *Number of Steps* and type the number of steps in the field.

Logarithmic

Specifies a logarithmic sweep.

- Highlight *Logarithmic*.

  The form changes to let you type either the number of points per decade or the total number of points (steps).

  - Highlight *Points per Decade* and type the number of points per decade in the field.

    or

  - Highlight *Number of Steps* and type the number of steps in the field.

Add Specific Points

Specifies additional sweep points for the analysis.

Highlight *Add Specific Points*, and type the additional sweep point values into the field. Separate them with spaces.

Sweeptype (Pnoise)

Controls the inclusion of the *sweeptype* parameter in the Spectre netlist. Choices are: *absolute*, *relative*, and *blank*, with *blank* selecting the appropriate Spectre default.

The results vary depending on whether you are simulating an autonomous circuit (an oscillator) or a driven circuit (a mixer) as determined by the *Oscillator* button selection on the `pss` Choosing Analysis form.

In general,

- When you simulate an autonomous circuit (the *Oscillator* section of the `pss` Choosing Analysis form is active), you can select *relative* for *Sweeptype*. If you leave *Sweeptype* blank, it will default to *relative*.
- When you simulate a driven circuit (the *Oscillator* section of the `pss` Choosing Analysis form is not active), you can either select either *relative* or *absolute* for *Sweeptype*. If you leave *Sweeptype* blank, it will default to *absolute*.

Absolute

Puts the *Sweeptype* as *absolute* in the Spectre netlist.

In the output for the `pnoise` sweep, the x axis corresponds to the *Start* and *Stop* values. There is no indication on the plot that you selected *Sweeptype* as *absolute.*

Relative

Puts the *Sweeptype* as *relative* in the Spectre netlist.

When you select *Sweeptype* as *relative*, you must also enter a value for *Relative Harmonic*. If you enter a `1`, it appears in the netlist as `relharmnum=1`.

In the output for a `pnoise` analysis, the x axis corresponds to the *Start* and *Stop* values in the `pnoise` sweep. The plot label indicates the selected relative harmonic (for example, `relharmnum=1`). In prior versions of the software, there is no indication.

Blank

Does not put a *Sweeptype* parameter in the Spectre netlist. Sets the appropriate Spectre default depending on the circuit type.

Sets the appropriate Spectre default depending on whether the circuit is autonomous or driven.

- For autonomous circuits, Spectre automatically sets the *Sweeptype* to *relative* and the *Relative Harmonic* to `1`.
- For driven circuits, Spectre automatically sets the *Sweeptype* to *absolute* and displays the message `Sweep is Currently Absolute.`

In the output, the `x` axis corresponds to the *Start* and *Stop* values in the `pnoise` sweep.

- For autonomous circuits, the plot label indicates the default *Relative Harmonic* (`relharmnum=1`). In prior versions of the software, there is no indication.
- For driven circuits, Spectre automatically sets the *Sweeptype* to *absolute*. There is no indication you selected *Sweeptype* as *blank*.

Sweeptype (PAC and PXF)

Controls the inclusion of the *sweeptype* parameter in the Spectre netlist. Choices are: *absolute*, *relative*, and *blank*, with *blank* selecting the appropriate Spectre default.

The results vary depending on whether you are simulating an autonomous circuit (an oscillator) or a driven circuit (a mixer) as determined by the *Oscillator* button selection on the `pss` Choosing Analysis form.

In general,

- When you simulate an autonomous circuit (the *Oscillator* section of the `pss` Choosing Analysis form is active), you can select *relative* for *Sweeptype*. If you leave *Sweeptype* blank, it will default to *relative*.

- When you simulate a driven circuit (the *Oscillator* section of the `pss` Choosing Analysis form is not active), you can either select either *relative* or *absolute* for *Sweeptype*. If you leave *Sweeptype* blank, it will default to *absolute*.

Absolute

Puts the *Sweeptype* as *absolute* in the Spectre netlist.

There is no indication on the plot that you selected *Sweeptype* as *absolute.*

Relative

Puts the *Sweeptype* as *relative* in the Spectre netlist.

When you select *Sweeptype* as *relative*, you must also enter a value for *Relative Harmonic*. If you type `1`, it appears in the netlist as `relharmnum=1`.

For `pxf` analysis with *sweeptype* set to *relative* and freqaxis=in, the simulation output is shifted frequency with the x axis clearly labeled `relative frequency (offset from xx HZ)`. For `pac` analysis with *sweeptype* set to *rel*ative and freqaxis=out, the simulation output is shifted frequency with the x axis clearly labeled `relative frequency (offset from xx HZ)`.

For `pxf` analysis with *sweeptype* set to *relative* and freqaxis=absin, the simulation output is absolute frequency. For `pac` analysis with *sweeptype* set to *relative* and freqaxis=absout, the simulation output is absolute frequency.

At the bottom of the direct plot form for the analysis, the `freqaxis` value displays along with one of the following messages-`relative frequency (offset xxx)` or `relative freq`. The plot label also indicates the selected relative harmonic (for example, `relharmnum=1`). In prior versions of the software, there is no indication.

Blank

Does not put a *Sweeptype* parameter in the Spectre netlist.

Sets the appropriate Spectre default depending on whether the circuit is autonomous or driven.

- For autonomous circuits, Spectre automatically sets the *Sweeptype* to *relative* and the *Relative Harmonic* to `1`.
- For driven circuits, Spectre automatically sets the *Sweeptype* to *absolute* and displays the message `Sweep is Currently Absolute.`

In the output, the `x` axis corresponds to the *Start* and *Stop* values in the `pnoise` sweep.

- For autonomous circuits, the plot label indicates the default *Relative Harmonic* (`relharmnum=1`). In prior versions of the software, there is no indication.
- For driven circuits, Spectre automatically sets the *Sweeptype* to *absolute*. There is no indication you selected *Sweeptype* as *blank*.

Sweeptype (PSP and QPSP)

Controls the inclusion of the *sweeptype* parameter in the Spectre netlist. Choices are: *absolute, relative*, and *blank,* with *blank* selecting the appropriate Spectre default.

Since the computations for `psp` analysis involve inputs and outputs at frequencies that are relative to multiple harmonics, *Sweeptype* behaves differently in `psp` and `qpsp` analysis than it does in `pac`, `pnoise`, and `pxf` analyses. With `psp` and `qpsp` analysis, the frequency of the input and the frequency of the response are usually different.

Absolute

Puts the *Sweeptype* as *absolute* in the Spectre netlist.

Specifying *Sweeptype* as *absolute*, sweeps the absolute input source frequency. There is no indication on the plot that you selected *Sweeptype* as *absolute.*

Relative

Puts the *Sweeptype* as *relative* in the Spectre netlist.

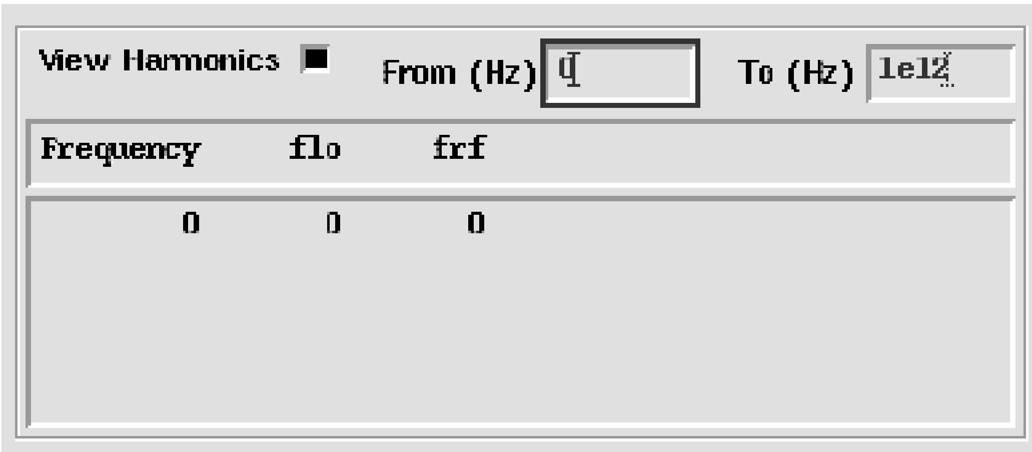
Specifying *Sweeptype* as *relative*, indicates to sweep relative to the analysis harmonics (rather than the `pss` or `qpss` fundamental).

Blank

Does not put a *Sweeptype* parameter in the Spectre netlist. Sets the appropriate Spectre default depending on the circuit type.

View Harmonics (QPSS)

Displays harmonics within a specified range of frequencies.



➤ In the *From (Hz)* and *To (Hz)* fields, type the lower and upper values for the frequency range.

The form changes to display all input frequencies, their harmonics, and the intermodulations of the frequencies and harmonics for named, top-level, large and moderate input signals in the circuit whose frequencies fall within the specified frequency range.

The first column is the frequency, and the remaining columns specify tone coefficients for each fundamental tone that contributed to the listed harmonic.

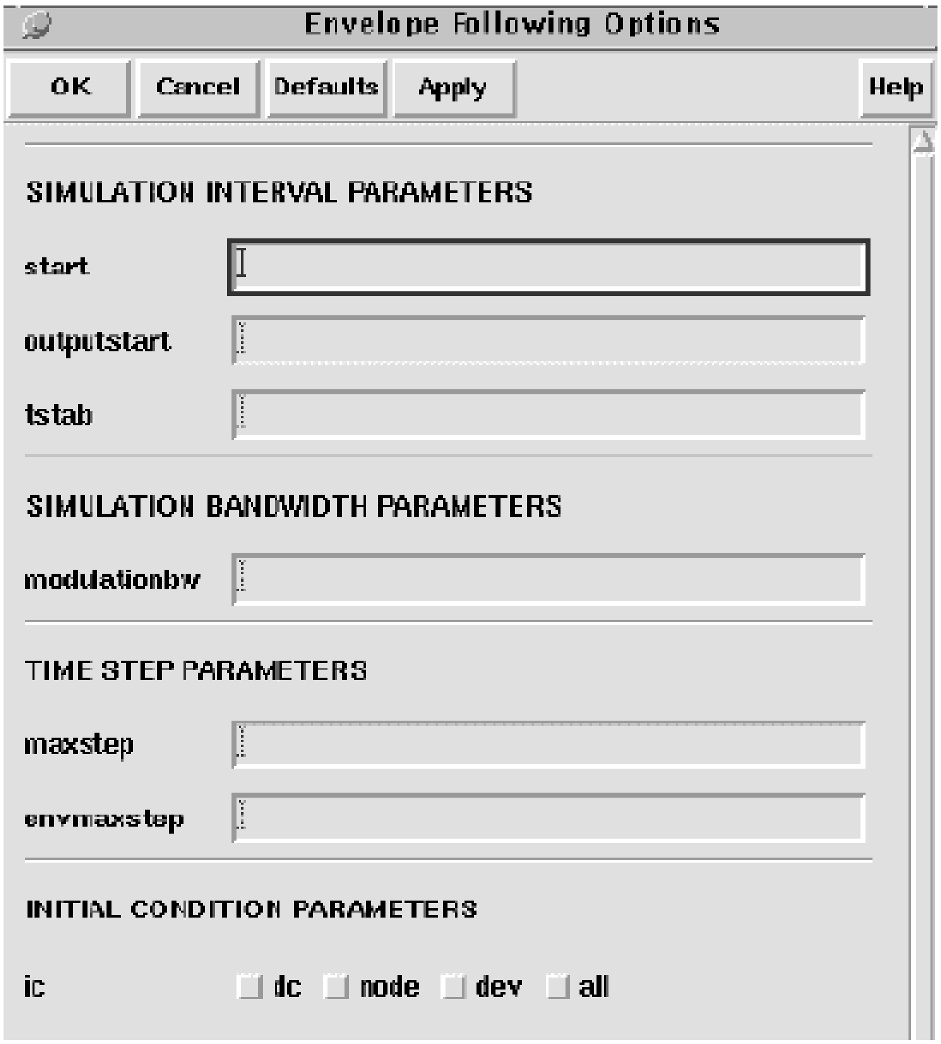
Options Forms

Each Options form lets you specify parameter values for a SpectreRF analysis. Options that are not relevant for a particular analysis do not appear on its Options form.

Opening the Options Forms

Click the *Options* button on the Choosing Analysis form to open the Options form corresponding to the analysis type that is currently highlighted on the Choosing Analysis form. Figure 2-2 on page 174 shows the Options form for the Envelope Following analysis.

Figure 2-2 The Envelope Following Options Form



Use the Options form for an analysis to define its parameter values. Only those options that are relevant for a particular analysis are available on its Options form.

Field Descriptions for the Options Forms

The following sections describe all the simulation parameters whose values you specify on Options forms. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the form.

Accuracy Parameters (PSS, QPSS, and Envlp)

Important

In most cases, the `errpreset` parameter is the only accuracy parameter you should set.

Use the following links to locate detailed descriptions of how the `errpreset` parameter works to set accuracy parameters.

“The errpreset Parameter in PSS Analysis” on page 38

“Using the errpreset Parameter With Driven Circuits” on page 38

“Using the errpreset Parameter With Autonomous Circuits” on page 39

“The errpreset Parameter in QPSS Analysis” on page 78

“The errpreset Parameter in Envlp Analysis” on page 110

relref is the reference used for relative convergence criteria. Your **Accuracy Defaults** choice sets the default value.

pointlocal compares the relative errors in quantities at each node to that node alone.

alllocal compares the relative error at each node to the largest values ever found for that node.

sigglobal compares relative errors in each signal to the maximum value for all signals.

allglobal is the same as sigglobal except that it also compares the residues for each node to the historical maximum.

lteratio is the ratio the simulator uses to compute LTE tolerances from the Newton tolerance. The default value is based on the accuracy default.

envlteratio is the ratio the simulator uses to compute envelope LTE tolerances for the Envelope Following Analysis. The default value is based on the accuracy default.

steadyratio is the ratio the simulator uses to compute steady-state tolerances from the LTE tolerance. This parameter adjusts the maximum allowed mismatch in node voltages and current branches during the steady-state period. The default is based on the accuracy default.

maxacfreq (not QPSS) is the maximum frequency used in a subsequent periodic small-signal analysis. This parameter automatically adjusts *maxstep* to reduce errors due to aliasing in frequency-domain results. The default is based on *maxstep* and *harms* values. See “SpectreRF Theory” for more information about specifying this parameter.

maxperiods is the maximum number of simulated periods allowed for the simulation to reach steady-state. Default is `20`.

finitediff specifies the use of the finite difference (FD) refinement method which is used after the shooting method to refine the simulation results. `finitediff` is only meaningful when `highorder` is set to no. The possible settings are *no*, *yes* or *refine*.

no turns off the finite difference refinement method.

yes applies the finite difference refinement method to the pss analysis. The finite difference method tries to improve the initial small time steps if necessary.

refine applies the finite difference refinement method to the pss analysis. The finite difference method tries to refine the time steps. When the simulation uses the *gear2* method, uniform 2nd order gear is used.

When you use *readpss* and *writepss* to re-use the pss analysis results, *finitediff* automatically changes from *no* to *yes*. If you are concerned that the accuracy of your simulation might be affected by a loose *steadyratio*, you might want to try *finitediff*.

You might also set *finitediff* to *yes* to get more uniform time steps and reduce the noise floor. This does not always work. At lower power levels, the finite difference method might sometimes reduce the noise floor. We recommend that you reduce the noise floor by setting *highorder* to *yes*.

The finite difference method works for driven circuits only. If you use the finite difference method with an oscillator circuit, it will serve only as a loading routine. Even when a circuit or analysis parameter has changed, the old solution is loaded and may be inaccurate.

highorder specifies the use of the high-order Chebyshev refinement method (MIC) which is used after the shooting method to refine the shooting method's steady-state solution. *highorder* works for driven circuits only and when `errpreset` is set to either `moderate` or `conservative`. The possible settings are *no* or *yes*.

no turns off the MIC method.

yes turns on the MIC method and tries harder to converge.

When you set `errpreset` to either `moderate` or `conservative` and you have not set *highorder*, MIC is used but it does not aggressively try to converge. MIC does try harder to converge when *highorder* is explicitly set to *yes*.

Annotation Parameters (All)

stats tells the simulator to generate analysis statistics. Default is no.

annotate lets you specify what information is printed at the beginning of the output to identify the results. Default is `status`. Choices are `no`, `title`, `sweep`, `status` and `steps`.

Convergence Parameters (All)

readns lets you specify the name of a file that contains an estimate of the initial transient solution. Enter the complete path to the file. No default.

cmin is the minimum capacitance from each node to ground. Default is 0.

tolerance is the relative tolerance for the linear solver when solving for convergence. Default is 10^{-9}.

gear_order is the order used for Gear-type interpolations. Default is 2 (second order).

solver

> **std** specifies that Krylov subspaces not be used. Simulations with this setting are slower but more robust than simulations using the turbo setting. Use this setting if simulation with *`turbo`* is unsuccessful.
>
> **turbo** specifies the use of Krylov subspaces and makes small-signal analyses run more quickly. However, it is less robust than *`std`*. This setting is the default.

oscsolver

> **std** specifies that Krylov subspaces not be used. The setting is more robust than *`turbo`* for oscillator small-signal analyses, although the simulation is slower. Use this setting if simulation with *`turbo`* is unsuccessful.
>
> **turbo** specifies the use of Krylov subspaces and makes oscillator small-signal analyses run more quickly. However, it is less robust than *`std`*. This setting is the default.

Initial Condition Parameters (PSS, QPSS, and Envlp)

ic specifies the methods used to set the initial condition (`dc`, `node`, `dev` and `all`). For an explanation of each of these methods, consult the Virtuoso Spectre Reference manual. Default is all.

skipdc

> **yes** omits the DC analysis from the transient analysis.
>
> **no** includes a DC analysis with the transient analysis. This is the default.

readic lets you specify the name of the file that contains the initial conditions.

Integration Method Parameters (PSS, QPSS, and Envlp)

method specifies the integration method. Your accuracy default choice sets the default value. The possible settings are

euler is backward Euler.

trap is the backward Euler and trapezoidal methods.

traponly is the trapezoidal rule only.

gear2 is the backward Euler and second-order Gear methods.

gear2only is Gear's second-order backward difference method only.

The trapezoidal rule is best when you want high accuracy, but it can exhibit point-to-point ringing, which you can control with tighter error tolerances. Euler and Gear work better with looser tolerances for quick simulation, but they can make systems appear more stable than they actually are.

Use the *gear_order* option on the small-signal analyses Options form to set the order of a Gear-type interpolation.

tstabmethod specifies the tstab integration method. Your accuracy default choice sets the default value. The possible settings are

euler is backward Euler.

trap is the backward Euler and trapezoidal methods.

traponly is the trapezoidal rule only.

gear2 is the backward Euler and second-order Gear methods.

gear2only is Gear's second-order backward difference method only.

The trapezoidal rule is best when you want high accuracy, but it can exhibit point-to-point ringing, which you can control with tighter error tolerances. Euler and Gear work better with looser tolerances for quick simulation, but they can make systems appear more stable than they actually are.

Use the *gear_order* option on the small-signal analyses Options form to set the order of a Gear-type interpolation.

envmethod specifies the integration method for the envlp analysis. Your accuracy default choice sets the default value. The possible settings are

euler is backward Euler.

trap is the backward Euler and trapezoidal methods.

traponly is the trapezoidal rule only.

gear2 is the backward Euler and second-order Gear methods.

gear2only is Gear's second-order backward difference method only.

trapgear2 is the backward Euler, trapezoidal and second-order Gear methods.

The trapezoidal rule is best when you want high accuracy, but it can exhibit point-to-point ringing, which you can control with tighter error tolerances. Euler and Gear work better with looser tolerances for quick simulation, but they can make systems appear more stable than they actually are.

Multitone Stabilization Parameter (QPSS)

stabcycles specifies the number of stabilization cycles to perform when both large and moderate sources are enabled. The default is 2.

Newton Parameters (PSS, QPSS, and Envlp)

maxiters is the maximum number of iterations per time step.

restart tells the simulator not to use the previous DC solution as an initial guess. Default is yes (means do not use).

envmaxiters lets you specify the maximum number of Newton iterations per envelope step for the envlp analysis. The default is 3.

Output Parameters (All)

save tells the simulator what signals to save. You have the following choices:

selected saves only the signals you request on the *Outputs* menu in the Simulation window. This is the default setting.

lvlpub saves all normally useful signals up to *nestlvl* deep in the subcircuit hierarchy. *lvlpub* is equivalent to *allpub* for subcircuits. Normally useful signals include shared node voltages and currents through voltage sources and iprobes.

Click on *lvlpub* to activate the *nestlvl* type-in field. Then enter a value for the *nestlvl* parameter.

lvl saves all signals up to *nestlvl* deep in the subcircuit hierarchy. *lvl* is relevant for subcircuits.

Click on *lvl* to activate the *nestlvl* type-in field. Then enter a value for the *nestlvl* parameter.

allpub saves only signals that are normally useful. Normally useful signals include shared node voltages and currents through voltage sources and iprobes.

all saves all signals.

Use *lvl* or *all* (instead of *lvlpub* or *allpub*) to include internal node voltages and currents through other components that compute current.

Use *lvlpub* or *allpub* to exclude signals at internal nodes on devices (the internal collector, base, emitter on a BJT, the internal drain and source on a FET, etc). *lvlpub* and *allpub* also exclude the currents through inductors, controlled sources, transmission lines, transformers, etc.

oppoint specifies whether the simulator outputs the operating point information. You can send the information to a rawfile, the logfile, or the screen. Default is *no*.

skipstart specifies when the simulator starts skipping output data. Default is *starttime s* (seconds).

skipstop specifies when the simulator stops skipping output data. Default is *stoptime s* (seconds).

skipcount specifies how many points to skip before saving a point. No default.

strobeperiod is the output strobe interval in seconds. The actual strobe interval is rounded to an integer multiple of the clock period.

Strobing is crucial to getting the very low noise floors required for ACPR measurements. Some users require noise floors 70 to 80 dB below the peak, which is not possible if the Fourier analysis has to interpolate between unevenly spaced data points.The *strobeperiod* option forces the output envelope to have evenly spaced data points. The subsequent Fourier analysis can proceed without interpolation.

strobedelay is the delay (phase shift) between the skipstart time and the first strobe point. Default is 0.

compression directs the simulator to perform data compression on the output. Default is *no.*

stimuli specifies what PXF uses for inputs to the transfer functions.

sources specifies that the sources present in the circuit are used as the inputs to the transfer functions.

nodes_and_terminals specifies that all possible transfer functions are computed.

freqaxis specifies what version of the frequency to plot the output against in spectral plots.

- For the `pnoise` analysis

  absout is the absolute value of the output frequency.

  in is the input frequency.

  out is the output frequency.

- For the `pac` and `qpac` analysis

  absout is the absolute value of the output frequency.

  in is the input frequency.

  out is the output frequency.

- For the `pxf` and `qpxf` analysis

  absin is the absolute value of the input frequency.

  in is the input frequency.

  out is the output frequency.

- For the `psp` and `qpsp` analysis

  - *absin* is the absolute value of the frequency swept at the input
  - *in* is the scattered frequency at the input
  - *out* is the scattered frequency at the output

outputperiod lets you specify the time-domain output period. The time-domain small-signal response is computed for the period specified, rounded to the nearest integer multiple of the PSS analysis period.

outputtype lets you specify the output type for Envelope Following Analysis. Possible values are *both*, *envelope*, or *spectrum*. The default is both.

saveallsidebands lets you save all sidebands for Pnoise and Qpnoise analyses.

Simulation Interval Parameters

tstart is the start time you specify for transient analysis. It can be negative or positive. Default is `0`.

start lets you specify the start time for the Envelope Following Analysis.

outputstart lets you specify the timepoint when the simulator starts to save output.

tstab is the initial stabilization time for the Envelope Following Analysis. Default is `0`.

Simulation Bandwidth Parameters

modulationbw lets you specify the modulation bandwidth.

State File Parameters

write lets you specify the name of the file to which the SpectreRF simulation writes the initial transient solution.

writefinal lets you specify the name of the file to which the SpectreRF simulation writes the final transient solution.

swapfile is a temporary file that holds steady-state information. If you enter a filename, the Spectre circuit simulator stores the operating point in that file rather than in virtual memory. Use this option if you receive a warning about not having enough memory to complete the analysis. Enter the complete path to the file.

writepss specifies the file to which the steady-state solution is written. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

writeqpss specifies the file to which the steady-state solution is written. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

readpss specifies the file from which the steady-state solution is read. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

readqpss specifies the file from which the steady-state solution is read. Small-signal analyses can read the steady-state solution from this file so rerunning the PSS analysis is unnecessary.

checkpss Options are yes and no.

Time Step Parameters

step is the smallest simulator time step used to improve the look of the results. Default is 0.001 x fundamental period seconds.

maxstep is the largest allowable time step. The default is set by the accuracy default.

liberal = 0.1/max AC frequency

moderate = 2 x liberal

conservative = 4 x liberal

envmaxstep specifies the maximum outer envelope size. The default is derived from the *errpreset* setting.

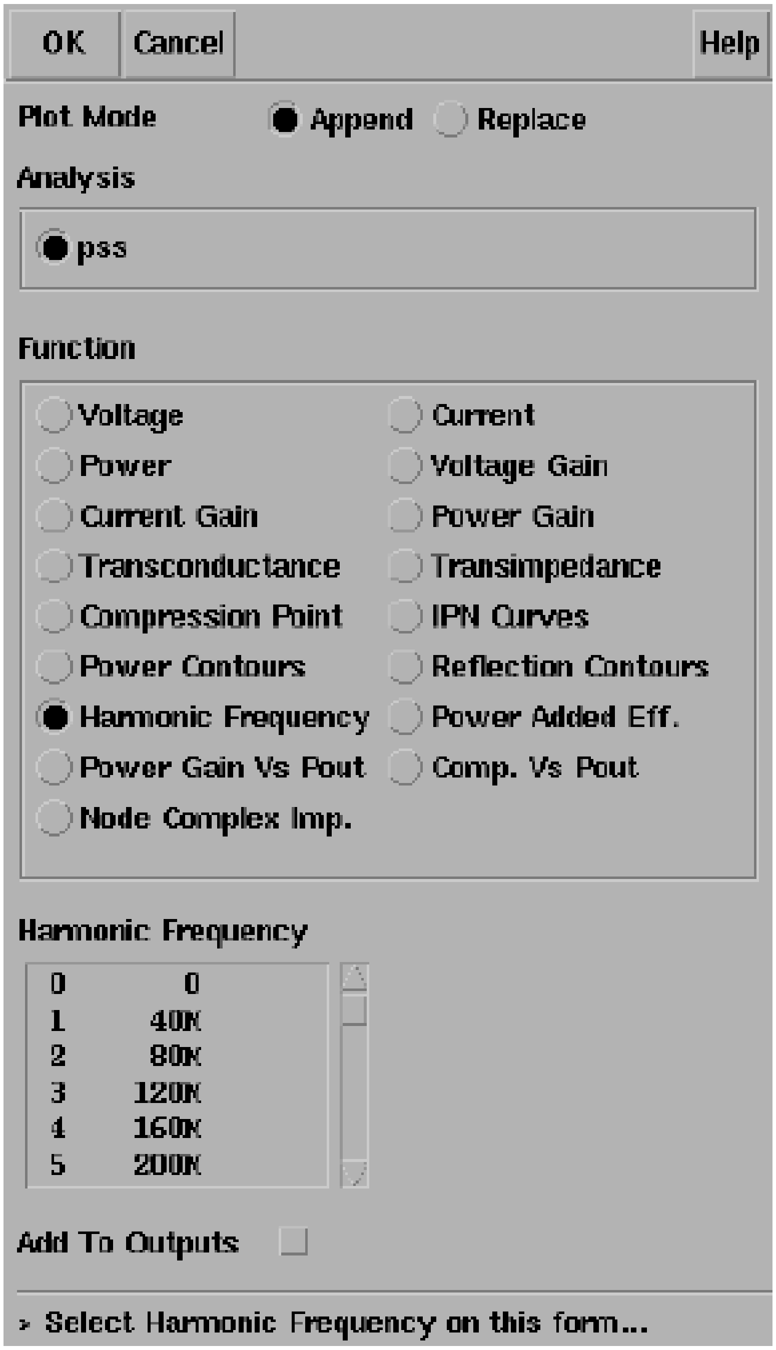
Direct Plot Form

Use the *Direct Plot* command in the Simulation window to plot most RF simulation results.

Opening the Direct Plot Form

Use the *Results – Direct Plot – Main Form* command in the Simulation window, to access results for the most recently performed analyses.The Direct Plot form will be similar to the one shown in Figure 2-3.

Figure 2-3 A Direct Plot Form



The *Analysis* section lists one or more analyses with available data.

- For Periodic large and small signal analyses—Choose *pss* to access the results from the initial *pss* analysis. Choose *pac*, *pnoise*, *pxf*, or *psp* to access the results of any available periodic small-signal analyses that ran after the *pss* analysis.
- For Quasi-Periodic large and small signal analyses—Choose *qpss* to access the results from the initial *qpss* analysis. Choose *qpac*, *qpnoise*, *qpxf*, or *qpsp* to access the results of any available quasi-periodic small-signal analyses that ran after the *qpss* analysis.
- Choose *Envlp* to access the results from an Envelope Following analysis.

Defining Measurements in a Plot Form

You define the RF simulation measurements to display by entering and selecting items in the Direct Plot form such as functions, plots, and modifiers and selecting nets, terminals, or other objects on the schematic.

While making selections in the Direct Plot form, follow the messages at the bottom of the form for instructions and prompts. For example,

> *Select Net on schematic...*

and

> *To plot, press Sij-button on this form...*

When you press the plot button (or perform another specified action), a simulation plot appears, by default, in the current Waveform Window. If the Waveform window or Schematic window is not open, selecting a direct plot function automatically opens both windows.

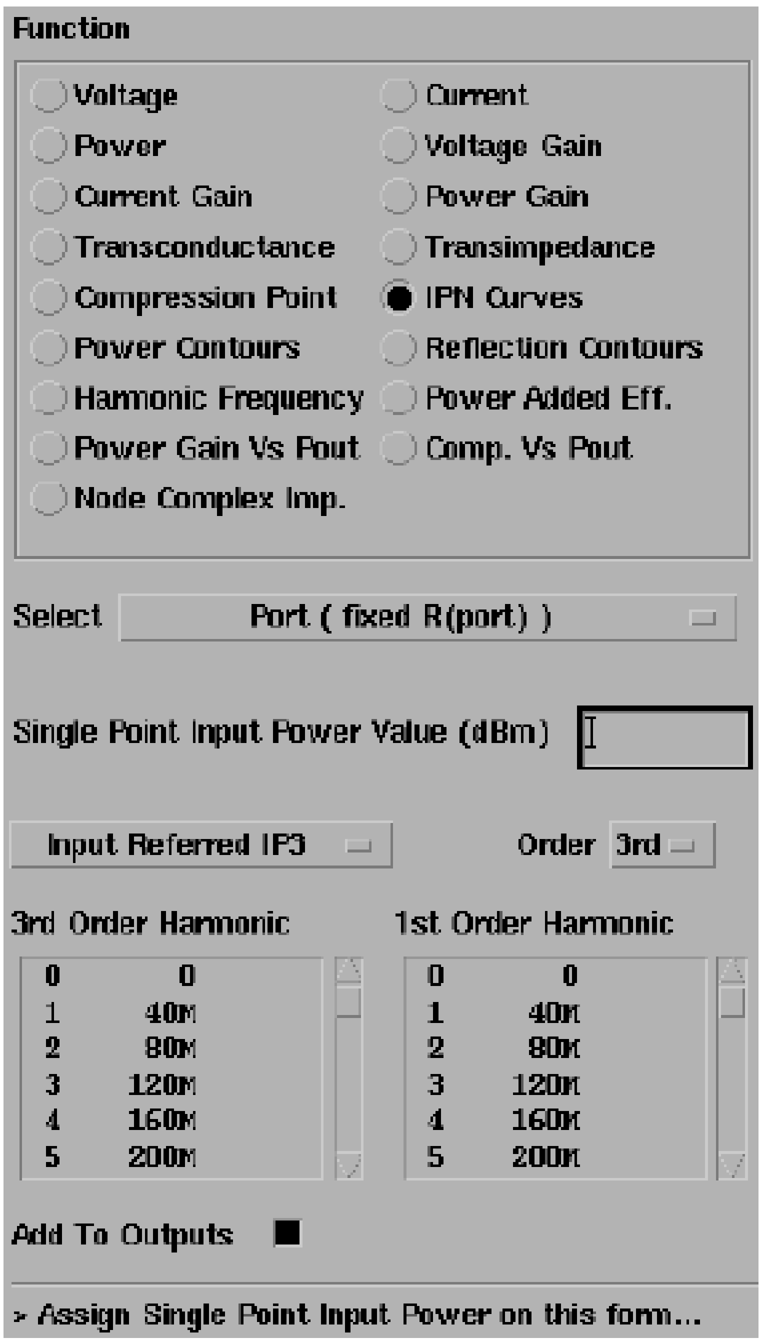
Informative messages often also display at the bottom of the Waveform and Schematic windows.

Plotting Data for Swept Simulations

For swept analyses, the last section in the Direct Plot form includes one or more list boxes which display different values of the design variable you selected when setting up the swept analysis. Figure 2-4 on page 186, the Direct Plot form for an IP# measurement, has two such list boxes. Following the messages at the bottom of the Direct Plot form, in the list box, click on the design variable values you want to plot. The name and type of the variable are displayed at the top of the list box.

For some measurements, for example, second and third order intercept functions, the Direct Plot form displays a second list of variable values for the reference variable. The following example shows the Direct Plot form for 3rd order IPN curves.

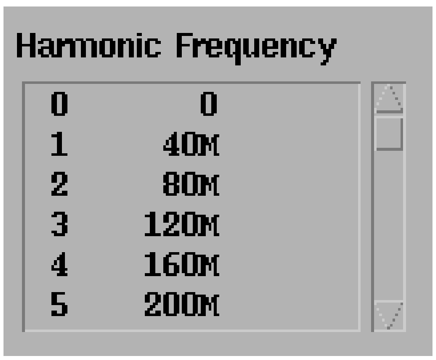
Figure 2-4 Direct Plot form for IP3 Curve



Follow the messages at the bottom of the Direct Plot form and click on the design variable values you want to plot.

Selecting Sidebands and Harmonics

When Sidebands or Harmonics are available, they are displayed in one or more list boxes at the bottom of the Direct Plot form.



As before, follow the messages at the bottom of the Direct Plot form and click on sideband or harmonic values in the list box to select them.

- To select one value, simply click on it.
- To select two or more adjacent values, click and drag with the mouse over the values you want to select.
- To select two or more values that are not adjacent, hold the *Control* key down while you click on the individual values.
- To deselect a value, hold the Control key down while you click on a selected value.

Generating a Spectral Plot

Spectral plots show the function level at each frequency component of a single analysis point (one step in a sweep of frequency, input power, or design variable).

1. In the Simulation window, choose *Results – Direct Plot – Main Form* to open the Direct Plot form, the Waveform window, and the design in the Schematic window.

   Follow the prompts displayed at the bottom of the Direct Plot form for instructions. Sometimes additional information is also displayed elsewhere in the form.

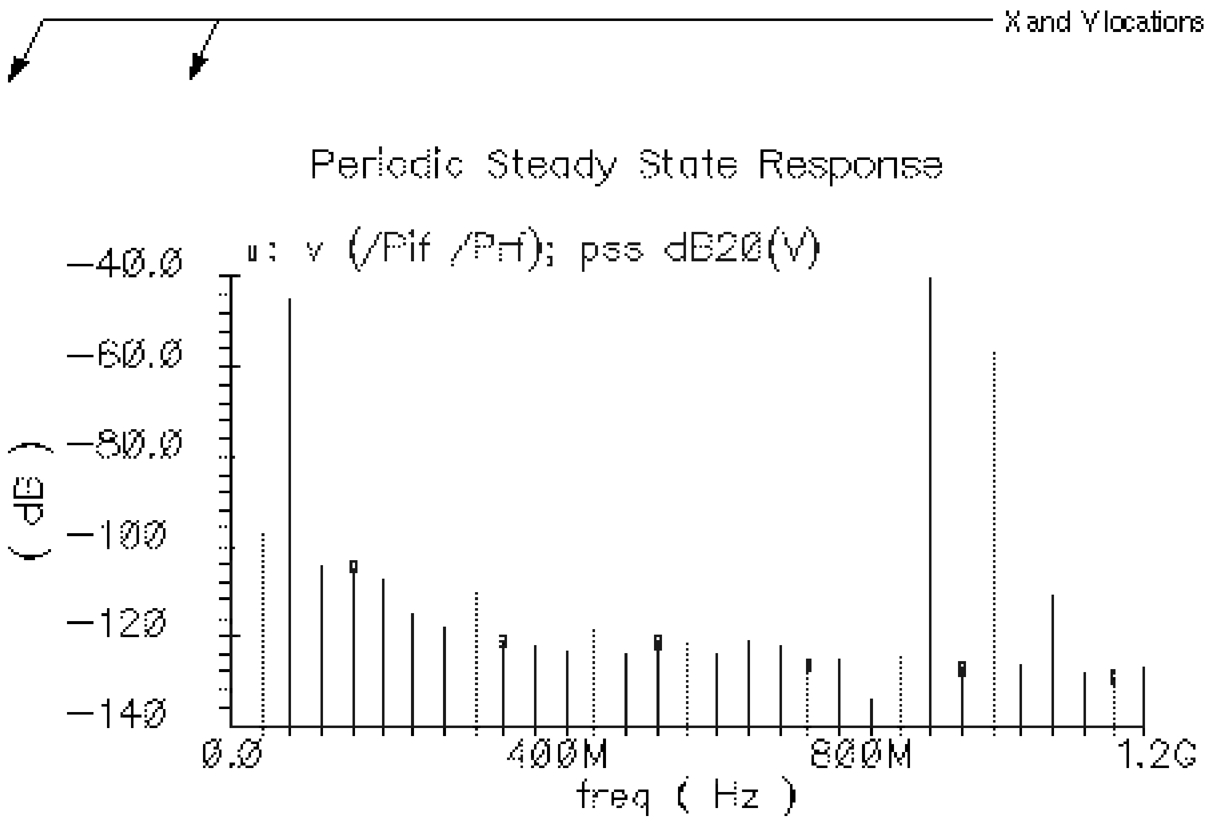
2. In the Direct Plot form, click on a *Plot Mode* button to specify whether the curves you plot are appended to or replace any existing curves in the Waveform window.

3. Click on an *Analysis* button to select the simulation you want to plot results for.
4. Click on a *Function* button to select the function or measurement you want to plot.

   The *Function* values displayed vary with the simulations run. If you need a function that is not included on the Direct Plot form, use the RF version of the analog design environment calculator. See the Waveform Calculator documentation for more information.
5. In the *Select* cyclic field, determine what to select in the Schematic window.

   All *Function* values now have several *Select* options. For example, for the *Voltage* function, you can now choose between *Net*, *Differential Nets* and *Instance with 2 Terminals*. Follow the prompts displayed at the bottom of the Direct Plot form for more information.
6. For PSS sweeps, set *Sweep* to *spectrum*.
7. For some RF functions, set *Signal Level,* choices are *peak* or *rms*.
8. For some RF functions, set a plot *Modifier*, choices are *Magnitude*, *Phase*, *dB20*, *Real*, or *Imaginary.*
9. Follow the prompts at the bottom of the Direct Plot form to draw the curve in the Waveform window.

The following waveform is the spectral plot used to calculate harmonic distortion for the ne600p mixer circuit in "Harmonic Distortion Measurement with PSS" on page 240.



Saving a Displayed Output and Displaying Saved Outputs.

- In the Plot form, first click the *Add to Outputs* button and then click the *Replot* button to add a plot displayed in the Waveform window to the *Outputs* list in the Simulation window.

- When you select a curve to plot from the list of curves in the *Outputs* section of the Simulation window, the new expression is *always* plotted as the *Magnitude* of the signal, not the plot with your original *Modifier* selection. If you want a specific scale, for example *dB20*, you have two choices. You can either
  - Modify the expression using *Curve – Edit – Scale* in the Waveform window

- ❑ Paste the expression from the *Outputs Setup* form into the calculator buffer and edit it.

Changing the Noise Floor of a Spectral Plot

1. Use the *Axes – Y axis* command in the Waveform window to change the noise floor of a spectral plot. In the Axes – Y axis form, set *Range* to *Min-Max* and specify a *Min* value.

Generating a Time Waveform

Time waveforms plot voltage and current against time.

1. Choose *Results – Direct Plot – Main Form* to open the Waveform window, the design in the Schematic window, and the Direct Plot form.

   Follow the prompts displayed at the bottom of the Direct Plot form for information on what to do next. Sometimes additional information is also displayed elsewhere in the form.

2. In the Direct Plot form, click on a *Plot Mode* button to specify whether the curves you plot are appended to or replace any existing curves in the Waveform window.

3. Click on an *Analysis* button to select the simulation you want to plot results for.

4. Click on a *Function* button to select the function or measurement you want to plot.

   The *Function* values displayed vary with the simulations run. If you need a function that is not included on the Direct Plot form, use the RF version of the analog design environment calculator. See the Waveform Calculator documentation for more information.

5. In the *Select* cyclic field, determine what to select in the Schematic window.

   All *Function* values now have several *Select* options. For example, for the *Voltage* function, you can now choose between *Net*, *Differential Nets* and *Instance with 2 Terminals*. Follow the prompts displayed at the bottom of the Direct Plot form for more information.

6. For PSS sweeps, set *Sweep* to *time*.

7. For some RF functions, set *Signal Level,* choices are *peak* or *rms*.

8. For some RF functions, set a plot *Modifier*, choices are *Magnitude*, *Phase*, *dB20*, *Real*, or *Imaginary.*

9. Follow the prompts at the bottom of the Direct Plot form to draw the curve in the Waveform window.

Saving a Displayed Output and Displaying Saved Outputs.

- In the Plot form, first click the *Add to Outputs* button and then click the *Replot* button to add a plot displayed in the Waveform window to the *Outputs* list in the Simulation window.

- When you select a curve to plot from the list of curves in the *Outputs* section of the Simulation window, the new expression is *always* plotted as the *Magnitude* of the signal, not the plot with your original *Modifier* selection. If you want a specific scale, for example *dB20*, you have two choices. You can either
  - Modify the expression using *Curve – Edit – Scale* in the Waveform window
  - Paste the expression from the *Outputs Setup* form into the calculator buffer and edit it.

Field Descriptions for the Plot Form

The following sections describe the fields on the Direct Plot form. The sections are arranged alphabetically, according to the top-level headings on the forms. The top-level headings are usually found along the leftmost margin of the forms.

1st Order Harmonic

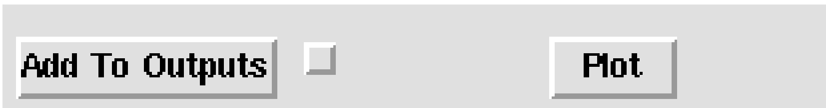
See First-Order Harmonic.

2nd-7th Order Harmonic

See Nth Order Harmonic.

Add To Outputs

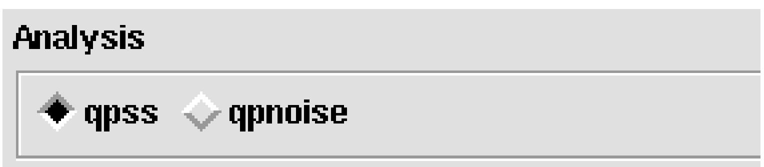
Adds the expression plotted in the Waveform window to the *Outputs* list.in the Simulation window.



- Highlight *Add To Outputs* to automatically add each new plot to the *Outputs* list whenever you click *Replot*.
- With *Add To Outputs* dehighlighted, click the *Add To Outputs* button to add only the current plot to the *Outputs* list.

Analysis

Selects the analysis to plot results for.

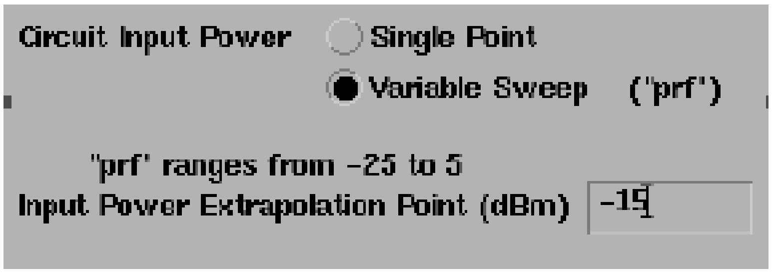


The *Analysis* area lists analyses for which results are available. If you performed one or more small-signal analysis, each small signal analysis has its own button separate from the button for the large-signal analysis performed as a prerequisite.

Highlighting an analysis changes the Direct Plot form to display fields for that analysis.

Circuit Input Power (QPSS, PAC)

Selects between *Single Point* and an appropriate *Sweep*.



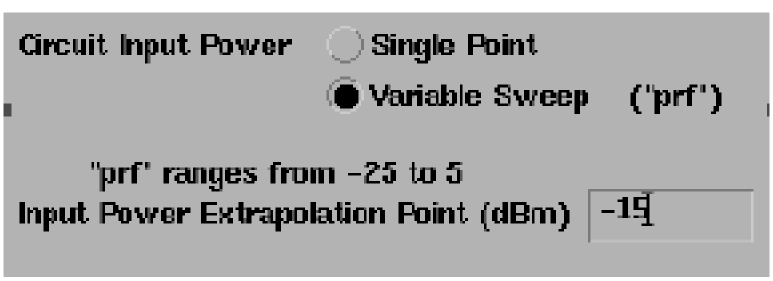
- If you choose *Single Point*, the *Input Power Value (dBm)* field appears, and you must type a value into it.
- If you choose a *Sweep*, the *Extrapolation Point* field appears, and you can type a value. Placing a value in the *Extrapolation Point* field is optional.

Close Contours (PSS and Envlp)

See “Maximum Reflection Magnitude (Envlp)” on page 201.

Extrapolation Point (PSS and QPSS)

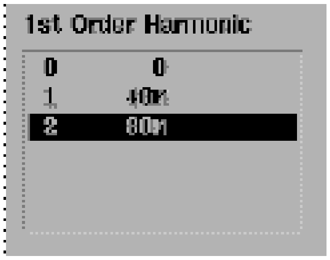
Specifies the value of the design variable from which the straight line approximations of first- and third-order harmonics are produced. See Figure 2-4 on page 193 for more information.



The *Extrapolation Point* field is optional. It specifies the value of the design variable from which the straight line approximations of first- and third-order harmonics are produced. If you do not specify a value, the default value is the smallest x axis sweep value.

First-Order Harmonic (PSS)

Lists available first-order harmonics when you select the *IPN Curves Function* for a PSS analysis.

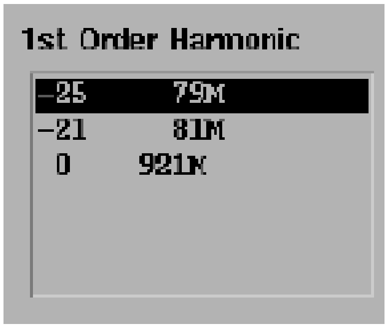


Lists the harmonics available for plotting by number and associated frequency values.

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

First Order Harmonic

Lists available first-order harmonics when you select the *IPN Curves Function* for an analysis.

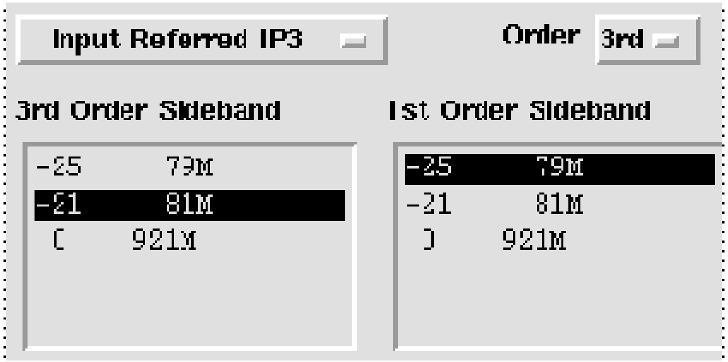


In the list box, the first column lists the frequency value of a harmonic, the following columns list the tone coefficient for each fundamental tone that contributed to the harmonic.

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

First Order Sideband (PAC)

Lists available first-order sidebands when you select the *IPN Curves Function* for a PAC analysis.

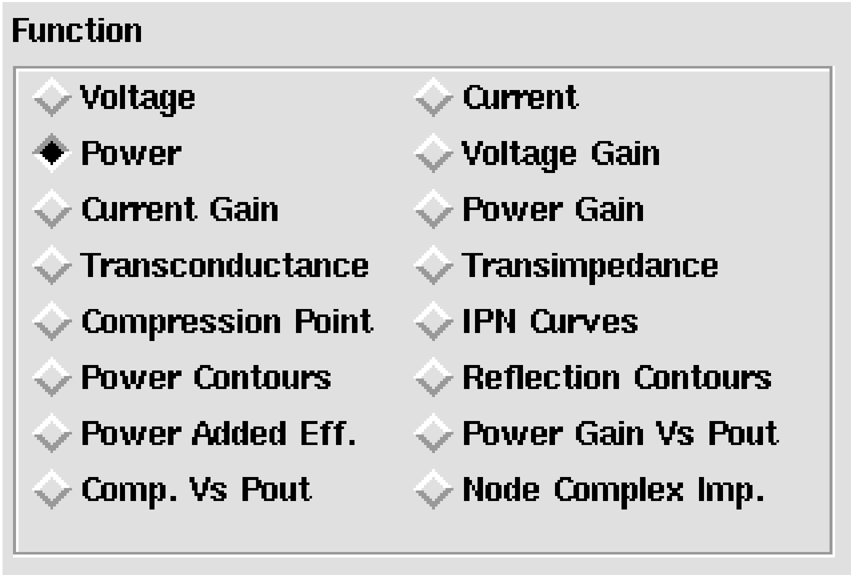


Lists the first-order sidebands available for plotting by number and associated frequency values.

➤ Select one sideband from the list box and then select the appropriate net on the schematic.

Function

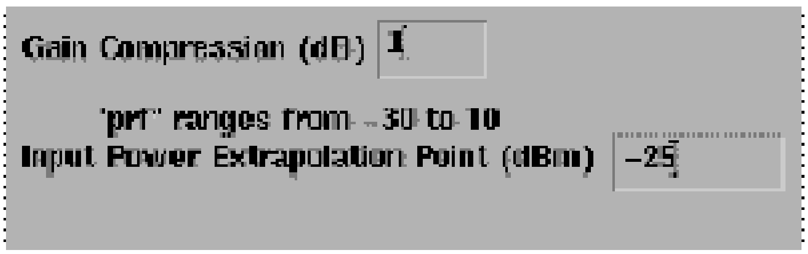
Specifies a quantity to plot.



Each *Function* button specifies a different quantity that you can plot. The available *Functions* vary depending on the *Analysis* you select. For some functions, you must select one or two objects on the schematic after selecting the *Function* button.

Gain Compression (PSS and QPSS)

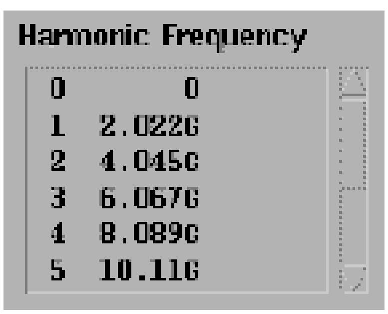
Specifies the *Gain Compression* when you plot the *Compression Point*.



Harmonic Frequency (PSS)

Lists available harmonic frequencies s by number when you select the *Harmonic Frequency* function.

.

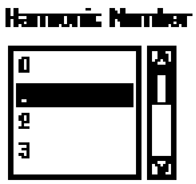


Lists the harmonics available for plotting by number and associated frequency values.

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

Harmonic Number (Envlp)

Lists available harmonics by number when you select the *Power Function* for an Envelope Following analysis.

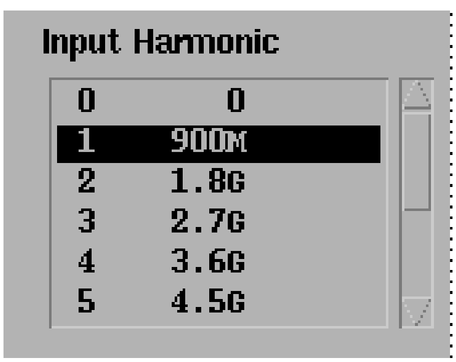


Lists the harmonics available for plotting by number and associated frequency values.

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

Input Harmonic (PSS)

Lists available input harmonics by number.

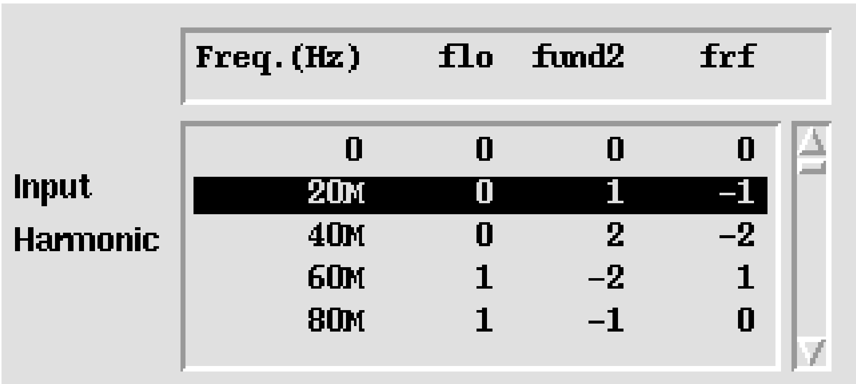


Lists available input harmonics by number. the list box appears on the PSS Plot form when you select one of the following functions: *Voltage Gain*, *Current Gain*, *Power Gain*, *Transconductance*, or *Transimpedance Functions*.

The values in the list box are those you requested in the *Output Harmonics* specification in the Choosing Analyses form.

Input Harmonic (QPSS)

Lists available input harmonics.



In the list box, the first column lists the frequency value of a harmonic. The following columns list the tone coefficients for each fundamental tone that contributed to the harmonic.

Input Power Value (dBm) (PSS, QPSS, and PAC)

Specifies an input power value.

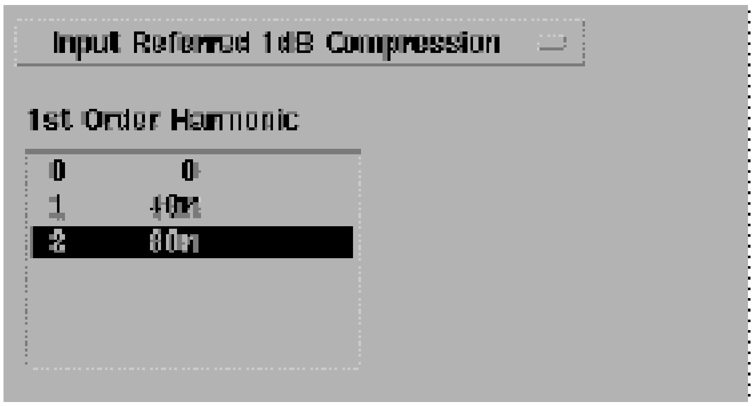
This field appears

- when you select *Single Point* for *Circuit Input Power*
- When you select the IPN Curves Function
- When you want information about a single point after running a power sweep.

➤ Assign an input power value.

Input or Output Referred 1dB Compression (PSS and QPSS)

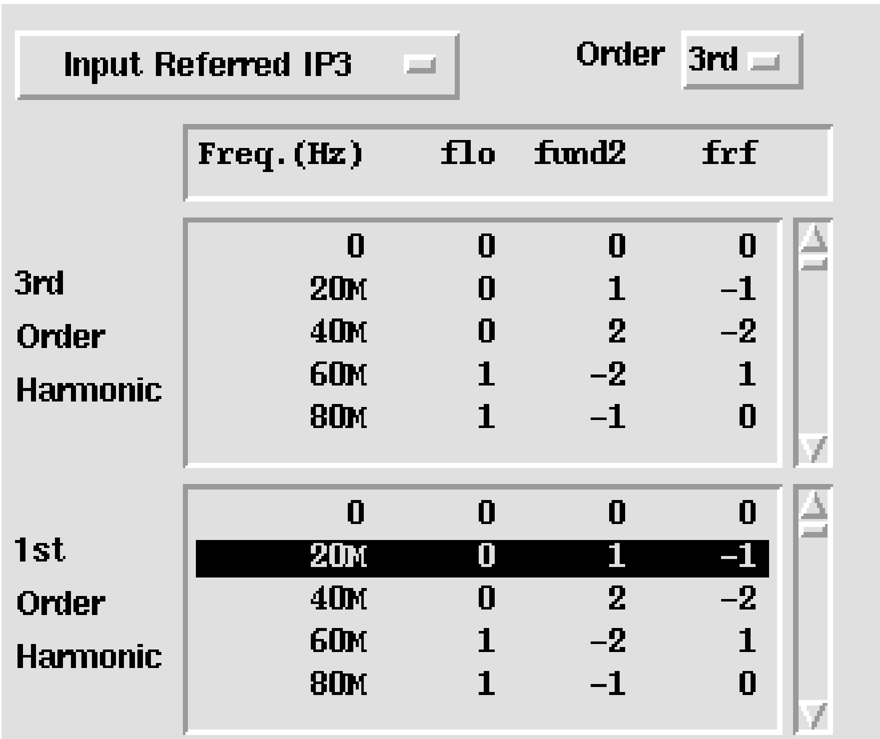
Selects input or output referred 1 dB compression.



- Output referred compression is referred to the y axis.
- Input referred compression is referred to the x axis.

Input or Output Referred IPN and Order (PSS, QPSS, and PAC)

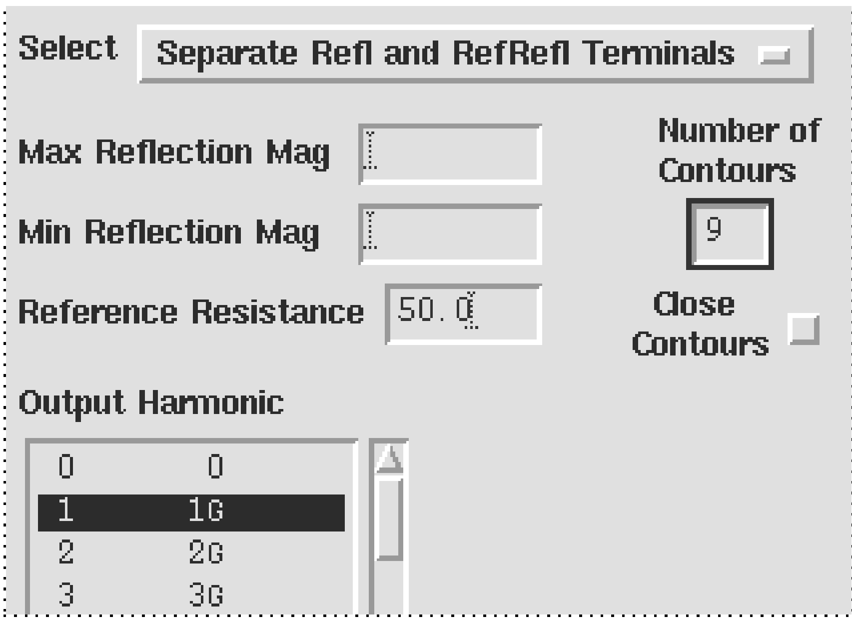
Selects Input or Output Referred Nth-Order Intercept point.



- Select the *Order*, *2nd* through *7th,* in the *Order* cyclic field.
- Select *Input Referred IPN* or *Output Referred IPN* in the Input/Output Referred IPN cyclic field.
    - Output referred IPN is referred to the y axis.
    - Input referred IP3 is referred to the x axis.

Maximum Reflection Magnitude (Envlp)

When you select the *Reflection Contours Function* in the envelope following analysis:



- Specifies a maximum reflection magnitude.
- Specifies a minimum reflection magnitude.
- Sets the resistance of the port adapter when you plot *Power* or *Reflection Contours*.
- Specifies the number of *Power* or *Reflection Contours* to plot.
- Sets open or closed contours for *Power* and *Reflection Contours*.
  - When you highlight *Close Contours*, the plot appears as a closed figure.
  - When you dehighlight *Close Contours*, the plot appears as an open figure.

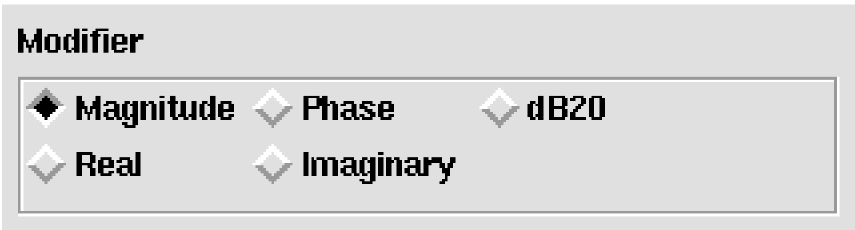
    The default is to leave the two most distant points in the plot unconnected.
- Selects the Output harmonic.

Min Reflection Mag

See “Maximum Reflection Magnitude (Envlp)” on page 201.

Modifier

Sets the units for the y axis of the plot.



Choices vary depending on the *Function* highlighted.

Magnitude is the raw value, in volts, amps, or no units at all.

Phase sets the y axis to degrees.

dB20 sets the y axis to decibels with tick marks every 20 dB.

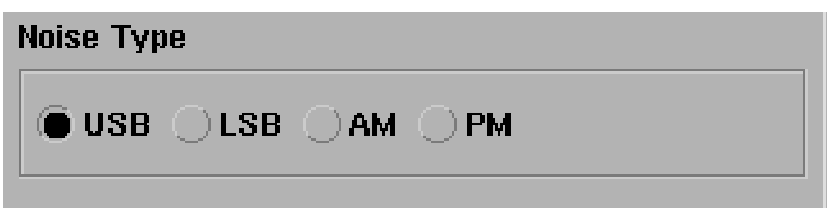
dBm sets the y axis to dB 10 plus 30.

dB10 sets the y axis to decibels with tick marks every 10 dB.

Real and **Imaginary** restrict plots to only the real or imaginary range of the curve.

Noise Type

Lists the types of noise calculated following a `Pnoise` analysis with *modulated Noise Type* selected.



Choices may vary.

USB is the upper sideband noise.

LSB is the lower sideband noise.

AM is the amplitude modulated noise.

PM is the phase modulated noise.

Number of Contours

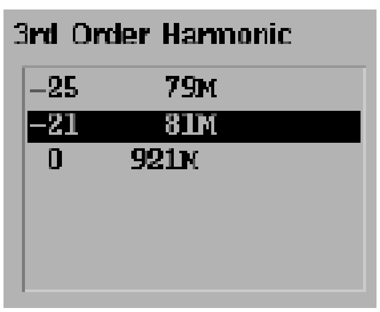
See "Maximum Reflection Magnitude (Envlp)" on page 201.

Lists the harmonics available for plotting by number and associated frequency values. (Select the *Order*, *2nd* through *7th,* in the *Order* cyclic field.)

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

Nth Order Harmonic (QPSS)

Lists available Nth Order Harmonics when you select the *IPN Curves Function* for the qpss analysis.

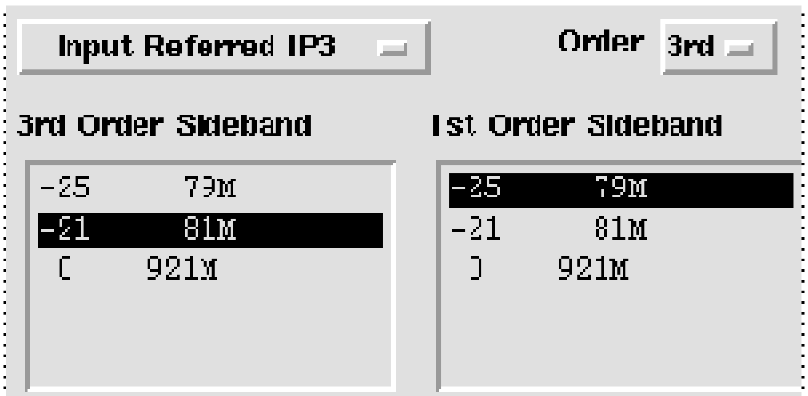


In the list box, the first column lists the frequency value of a harmonic. The following columns list the tone coefficient for each fundamental tone that contributed to the harmonic.

➤ Select one harmonic from the list box and then select the appropriate net on the schematic.

Nth Order Sideband (PAC)

Lists available Nth Order sidebands when you select the *IPN Curves Function* for a PAC analysis.



Lists the sidebands available for plotting by number and associated frequency values. (Select the *Order*, *2nd* through *7th,* in the *Order* cyclic field.)

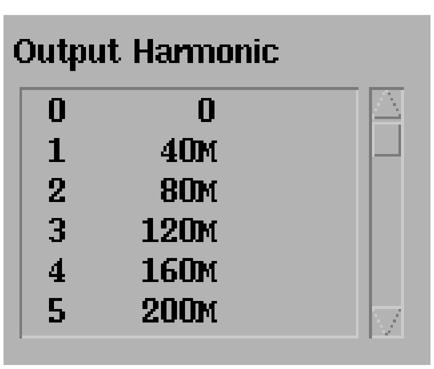
➤ Select one sideband from the list box and then select the appropriate net on the schematic.

Order

See “Input or Output Referred IPN and Order (PSS, QPSS, and PAC)” on page 200.

Output Harmonic (PSS)

Lists available Output Harmonics for PSS analysis.



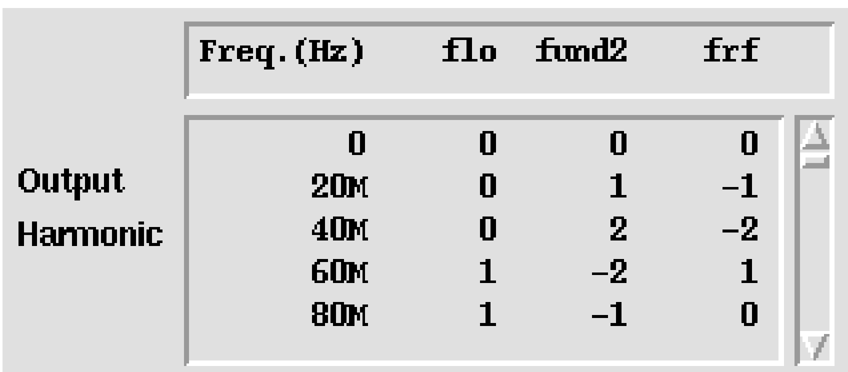
Lists the harmonics available for plotting by number and associated frequency.

Click on harmonics in the list box to select them.

- Select adjacent harmonics by clicking and dragging with the mouse over the harmonics you want to select.
- Select harmonics that are not adjacent by holding the `Control` key down while you click on the individual sidebands.
- Deselect harmonics by holding the `Control` key down while you click on a selected harmonic.

Output Harmonic (For QPSS)

Lists available Output Harmonics for qpss analysis.



Output Harmonic

Freq.(Hz)	flo	fund2	frf
0	0	0	0
20M	0	1	-1
40M	0	2	-2
60M	1	-2	1
80M	1	-1	0

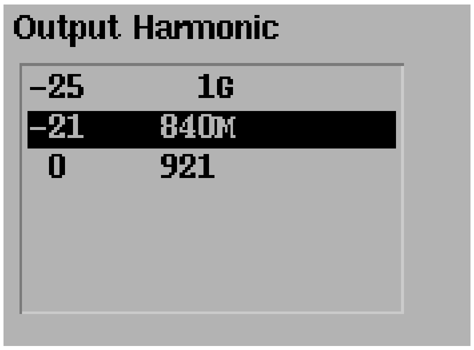
In the list box, the first column is the frequency of a harmonic. The second and third columns specify the tone coefficients for each fundamental tone that contributed to the listed harmonic.

Click on harmonics in the list box to select them.

- Select adjacent harmonics by clicking and dragging with the mouse over the harmonics you want to select.
- Select harmonics that are not adjacent by holding the `Control` key down while you click on the individual sidebands.
- Deselect harmonics by holding the `Control` key down while you click on a selected harmonic.

Output Sideband (PAC and PXF)

Lists the sidebands you requested on the small-signal Choosing Analyses form.



This list box appears when you choose variable for *Sweep* on a small-signal Direct Plot form. It lists all the sidebands you requested on the small-signal Choosing Analyses form.

Click on sidebands in the list box to select them.

- Select adjacent sidebands by clicking and dragging with the mouse over the sidebands you want to select.
- Select sidebands that are not adjacent by holding the `Control` key down while you click on the individual sidebands.
- Deselect sidebands by holding the Control key down while you click on a selected sideband.

Plot and Replot

Displays a plot in the Waveform window.

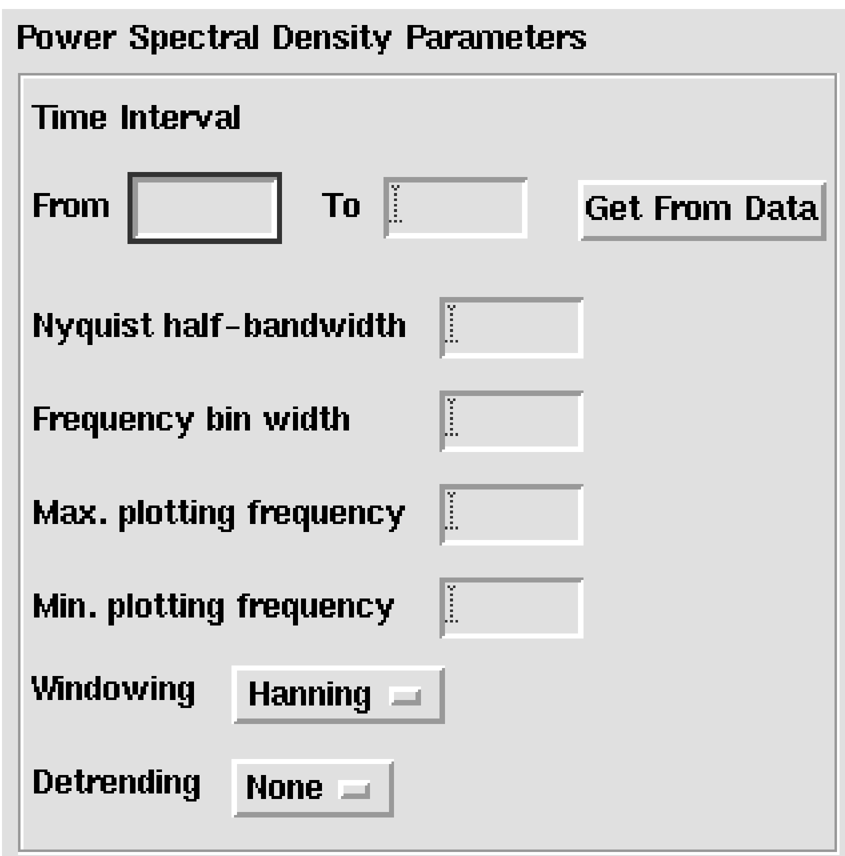
Plot Mode

Determines whether to add the next plot to those currently displayed in the Waveform window or to clear the window and display only the next plot.

- *Append* combines the next plot with other curves already plotted in the Waveform window
- *Replace* clears the Waveform window just prior to displaying the next plot.

Power Spectral Density Parameters (Envlp)

Determines how the Power Spectral Density is calculated.



- **Time Interval** – The starting and ending times for the spectral analysis interval. They are usually the start and stop times for the simulator.
- **Get From Data** --
- **Nyquist half-bandwidth** – The maximum frequency at which there are signals of interest. This is usually three to five times the maximum band frequency.
- **Frequency bin width** – The frequency resolution, such as the width of the frequency bins.
- **Max. plotting frequency** – Sets the maximum x axis value for the waveform you want to plot.
- **Min. plotting frequency** – Sets the minimum x axis value for the waveform you want to plot.
- **Windowing** – A preset list of available windowing functions used during the spectrum calculation.
- **Detrending** – Removes trends from the data before the spectral analysis.

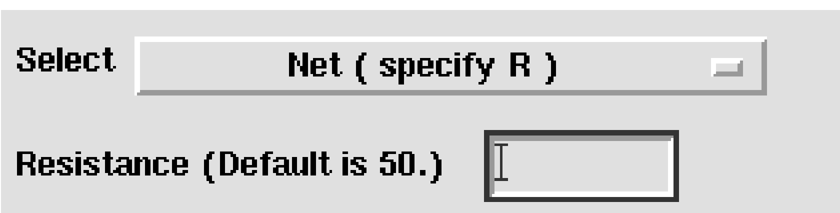
Reference Resistance (Envlp)

See “Maximum Reflection Magnitude (Envlp)” on page 201.

Resistance

Sets the resistance of the port adapter when you plot *Power* or *Reflection Contours*.

Sets the resistance of the port adapter when you select Net (specify R) and plot Power or Reflection Contours.



Select

Determines the type and number of objects to select on the schematic.

Choices available in the *Select* cyclic field vary depending on the highlighted *Function*. The message at the bottom of the form prompts you to make an appropriate selection.

- **Differential Nets** -- Select differential nets on the schematic.
- **Differential Nets (dB, 1ohm reference)** -- Select differential nets on the schematic.
- **Differential Terminal** -- Select a differential terminal on schematic.
- **Instance with 2 Terminals** -- Select an instance with two terminals on the schematic.
- **Net** -- Select a net on the schematic.
- **Net (dB, 1ohm reference)** -- Select a net on the schematic.
- **Out. and In. Ports (fixed R(OutPort))** -- Select output and input ports on the schematic.
- **Out. and In. Instances with 2 Terminals** -- Select output and input instances with two terminals on the schematic.
- **Output and Input Nets** -- Select output and input nets on the schematic.
- **+- Output and +- Input Nets** -- Select output and input nets on the schematic.
- **Output and Input Terminals** -- Select output and input terminals on the schematic.
- **+- Output and +- Input Terminals** -- Select output and input terminals on the schematic.
- **Output Net and Input Terminal** -- Select an output net and an input terminal on the schematic.
- **+- Output Net and Input Terminal** -- Select an output net and an input terminal on the schematic.

- **Port (fixed R(port))** -- Select a port on the schematic.
- **+- Power and +- Refl Terminals** -- Select power and reflection terminals on the schematic.
- **Separate Power and Refl Terminals** Select separate power and reflection terminals on the schematic.
- **Single Power/Refl Terminal** Select one power or reflection terminal on the schematic.
- **Single Power/Refl Term and ref Term** Select one power or reflection terminal and a reference terminal on the schematic.
- **Terminal** -- Select a terminal on the schematic.
- **Terminal and V-Reference Terminal** -- Select a terminal and a V-Reference terminal on the schematic.

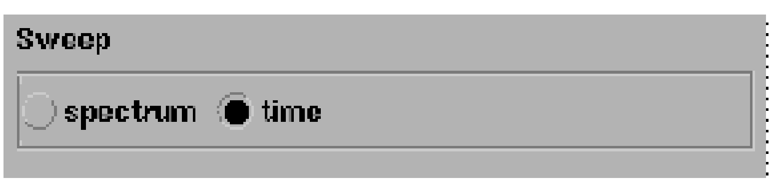
Signal Level (PSS, QPSS)

Determines the signal value to plot.

- **Peak** -- Plots the maximum value of the signal.
- **rms** -- Plots the root-mean-square value, or effective value, of the signal.

Sweep (PSS, PXF, and Envlp)

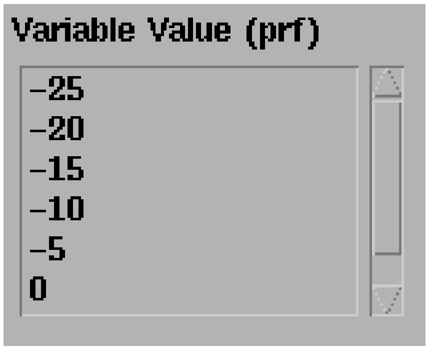
Sets the units for the x axis of the plot. Choices vary depending on the *Function* highlighted.



- *spectrum*, *sideband*, and *frequency* set the x axis to display frequency
- *time* sets the x axis to display seconds
- *variable* sets the x axis to display the value of a design variable.

Variable Value (PSS, QPSS, and PAC)

Lists the swept variable values that you can plot for a PSS, QPSS, or PAC analysis.



The PSS, QPSS, and PAC Results forms display this list of sweep values that you can specify for a specific variable, temperature, component parameter, or model parameter. the variable name is included in the title.

The range of values is determined by the *Sweep Range* specification in the Choosing Analyses form. The number of values is determined by the *Sweep Type* specification in the Choosing Analyses form.

In the sample figure, which shows the PSS version, the values are listed for the design variable `prf`.The QPSS version of the form is formatted slightly differently but gives the same information.

The ACPR Wizard

The ACPR Wizard simplifies the procedure for measuring ACPR and PSD.

The following sections describe the fields on the ACPR Wizard form. The sections are arranged alphabetically, according to the top-level headings on the ACPR Wizard form. The top-level headings are usually found along the leftmost margin of the form.

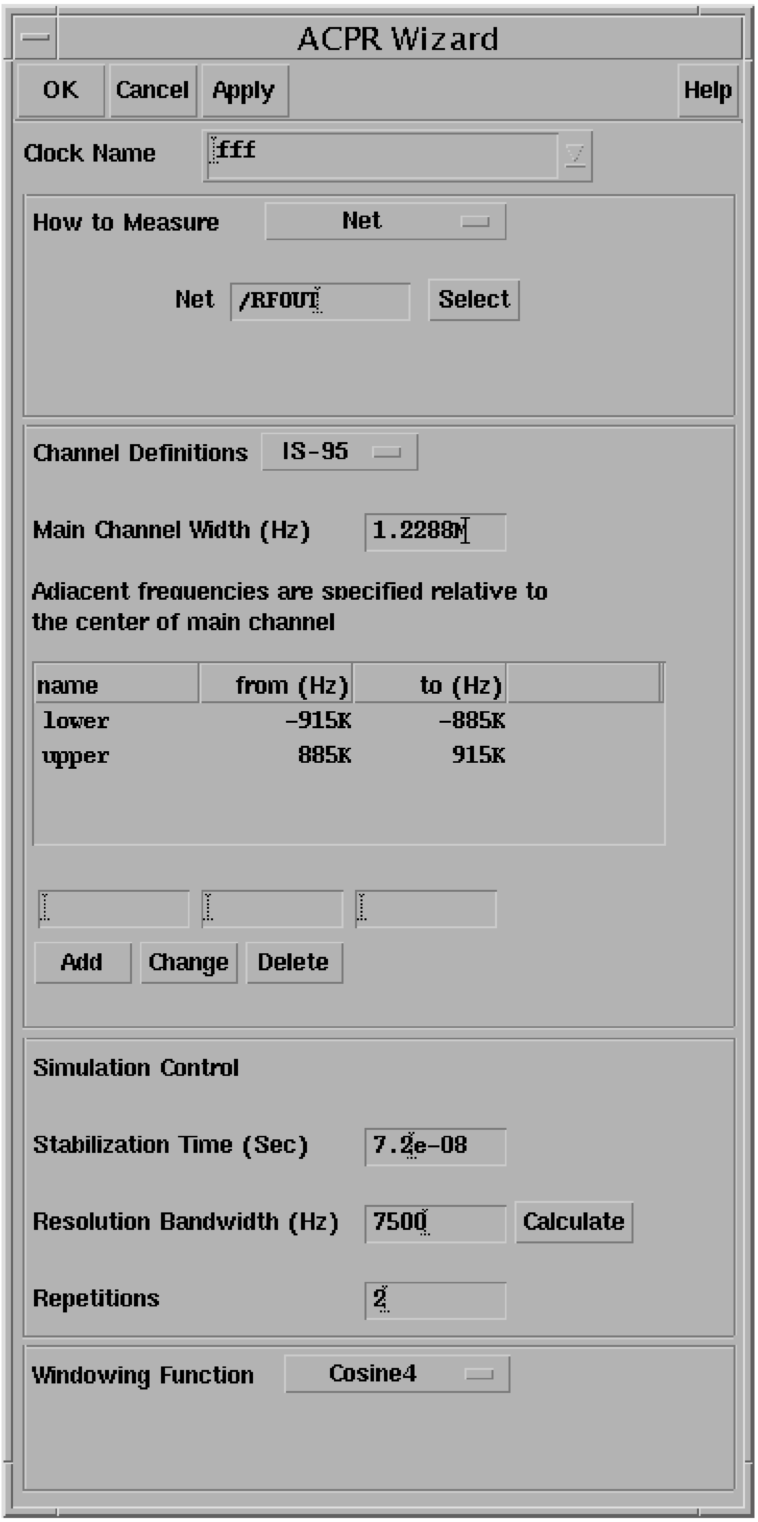
Open the ACPR wizard in one of two ways.

- In the Simulation window, choose *Tools – RF – Wizards – ACPR*

  or

- In the *envlp* Choosing Analyses form, press *Start ACPR Wizard*.

The ACPR wizard form opens.



Clock Name

In the *Clock Name* cyclic field, select the clock signal from those listed in the cyclic field.

The *Clock Name* identifies the source of the modulated signal.

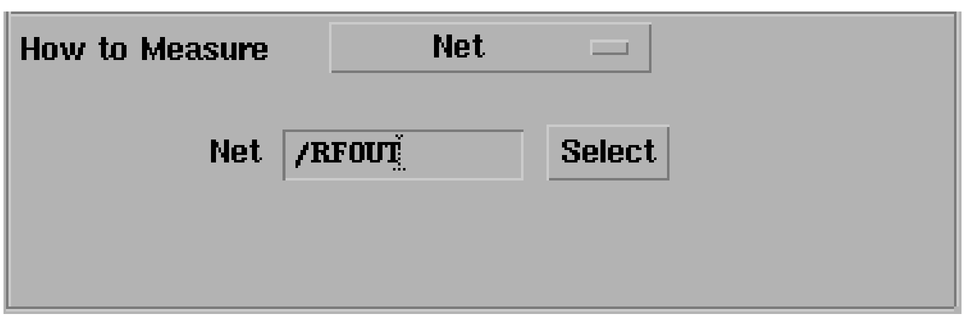
How to Measure

The *How to Measure* section allows you to choose whether to measure ACPR for a single *Net* or between *Differential Nets*. Use the *How to Measure* cyclic field to choose a single *Net* or *Differential Nets*.

To measure ACPR for a single net

1. Select *Net* in the *How to Measure* cyclic field.
2. Press *Select*. Then select the output net in the Schematic window.

   RFOUT displays in the *Net* field.

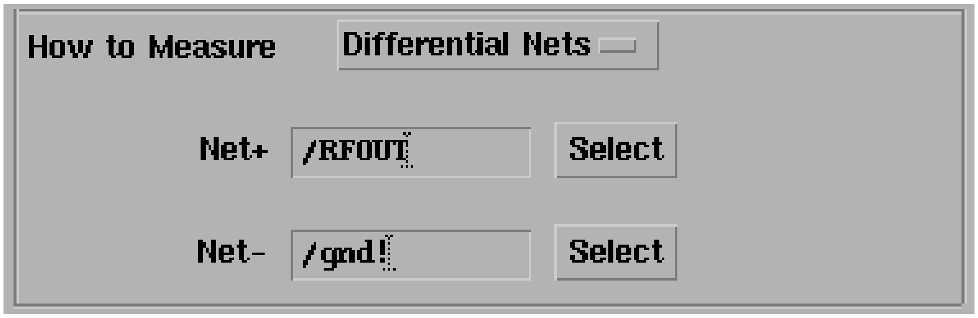


To measure ACPR for differential nets

1. Select *Differential Nets* in the *How to Measure* cyclic field.
2. Press *Select* next to the *Net+* field. Then select the positive net in the Schematic window.

   The signal name displays in the *Net+* field.
3. Press *Select* next to the *Net-* field. Then select the negative net in the Schematic window.

The signal name displays in the *Net-* field.



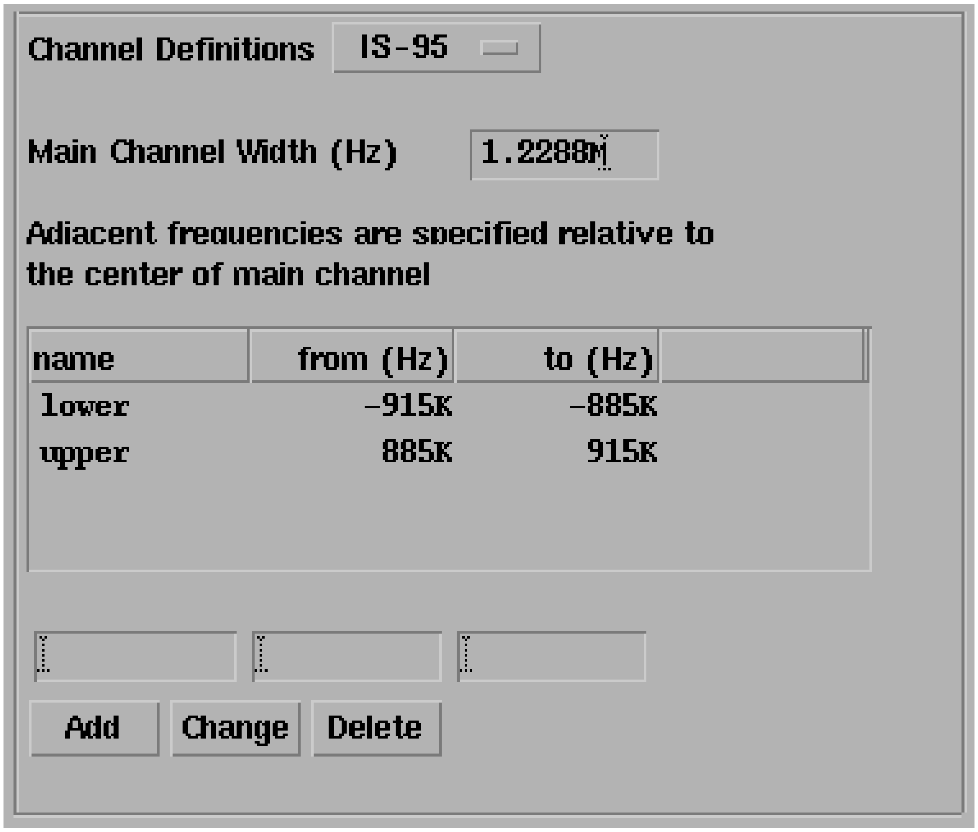
Channel Definitions

In the *Channel Definitions* cyclic field, select a preset channel definition. Choices include *Custom*, *IS-95*, and *W-CDMA*.

When you select *IS-95*

- The *Main Channel Width (Hz)* field is calculated. This is the width of the main channel in Hz. Enter a number greater than zero. When you click OK or Apply, the content of this field is verified.
- The *adjacent frequencies* are determined and display in the list box. Note that adjacent frequencies are specified relative to the center of the main channel.

  Use the edit fields and the *Add*, *Change* and *Delete* buttons to enter or modify channel definitions in the list box. You can hand edit or enter adjacent frequency names and upper and lower boundaries. Specify adjacent frequencies relative to the center of the main channel. All channel widths must be greater than zero.



Simulation Control

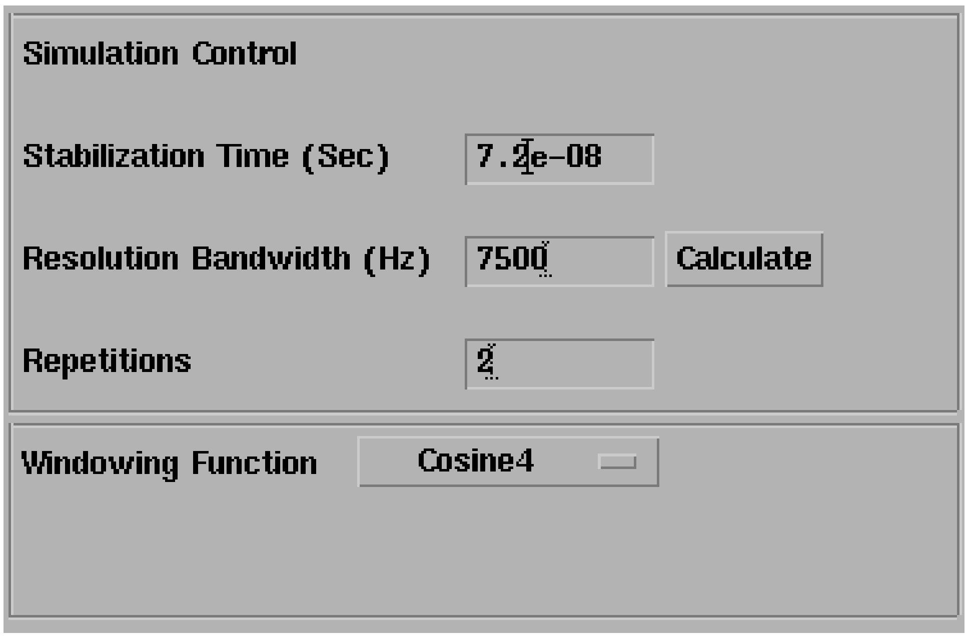
Stabilization Time (Sec.) specifies the number of seconds to wait before using the data for analysis.

Resolution Bandwidth (Hz) specifies the spacing of data points on the on the power density curve, in Hz. Use the Calculate button to calculate the resolution bandwidth. When you decrease the resolution bandwidth, simulation time is longer and the data file is larger.

Repetitions specifies the number of times to repeat the DFT for averaging. When you increase the number of repetitions, the power density curve is smoother, simulation time is longer and the data file is larger.

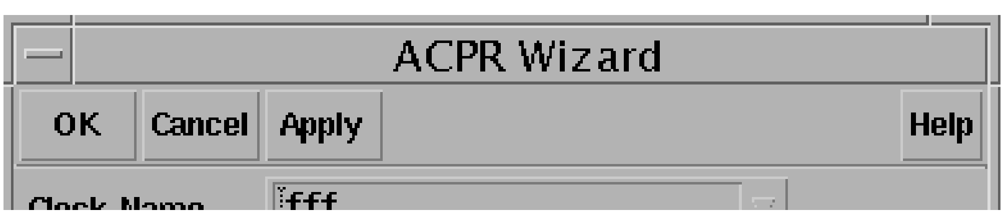
In the *Windowing Function* cyclic field, select a preset windowing function. Choices include *Windowing Function* presets include *Blackman, Cosine2, Cosine4, ExtCosBell,* *HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular.*

A *Windowing Function* tapers the signal before performing the DFT to reduce the effect of any edge discontinuities.



Apply and OK

When you click *Apply* or *OK*, the values you entered in the ACPR wizard are used to determine values for the required `envlp` analysis and the `envlp` choosing analysis form is filled in.



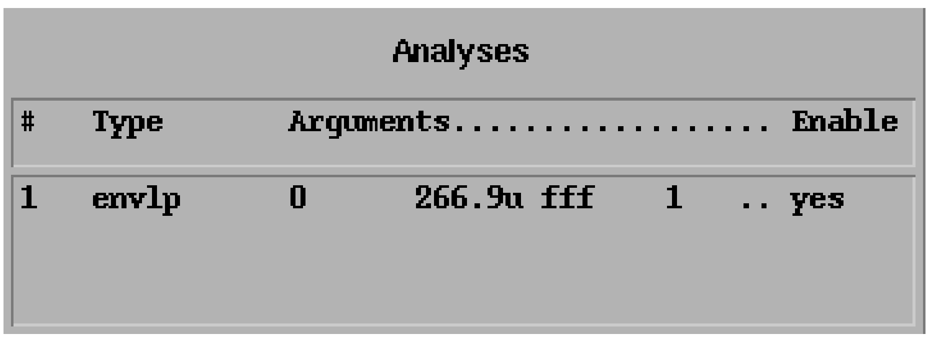
In the `envlp` choosing analysis form, values are calculated as follows.

- The *Clock Name* field in the `envlp` form is the same as the *Clock Name* field on the ACPR wizard.
- The *Stop Time* value in the `envlp` form is calculated.

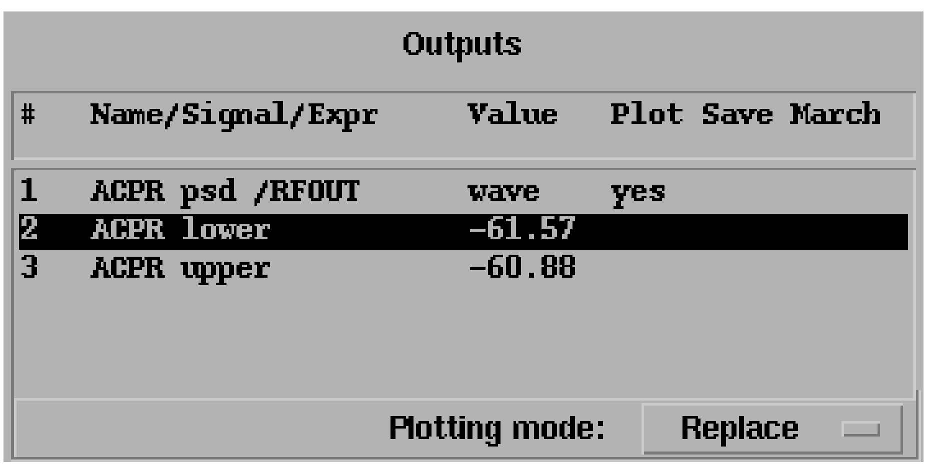
- The *Output Harmonics* field in the `envlp` form is set to `1`.
- The `envlp` choosing analysis form is enabled.
- The *Start* field in the `envlp` Options form is left blank.
- The *modulationbw* value for the `envlp` Options form is calculated.
- The *strobeperiod* value for the `envlp` Options form is calculated.

You can modify these values entered on the `envlp` analysis form, but your changes will not be propagated back to the ACPR wizard.

The *Analyses* area in the Simulation window reflects the `envlp` analysis.



When the `envlp` analysis completes, the ACPR values for each channel and the PSD waveform display in the *Output* area in the Simulation window.



The RF Simulation Forms Quick Reference

The SpectreRF simulation forms include the following:

- Choosing Analysis Form
- Option Forms (One for each analysis)
- Direct Plot Form
- ACPR Wizard

The simulation forms change to display only the fields relevant for the currently selected analysis. The field description topics for each analysis form, are briefly described here and linked to the detailed description in this chapter.

Choosing Analysis Form

Periodic Steady-State (PSS) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *pss* to display the Periodic Steady State Analysis form fields.)

Fundamental Tones displays and edits information for top level tones in the circuit.

Beat Frequency, **Beat Period**, and **Auto Calculate** determine whether the pss analysis uses *Beat Frequency* or *Beat Period*.

Output Harmonics selects and defines output harmonics.

Accuracy Defaults quickly adjusts simulation parameters.

Oscillator defines the simulation for an oscillator circuit. (Displays additional fields to specify oscillator analysis.)

Sweep selects swept PSS analysis. (Displays additional fields to specify sweep.)

Enabled includes this analysis in the next simulation.

Options displays the PSS Options form.

Modifications to PSS Form for Oscillator Analysis

Oscillator Node and **Reference Node** specify how the PSS oscillator analysis is performed.

Modifications to PSS Form for Swept PSS Analysis

Sweep, **Sweep Range**, **Sweep Type**, and **Add Specific Points** specify how the PSS sweep is performed.

Quasi-Periodic Steady State (QPSS) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *qpss* to display the Quasi-Periodic Steady State Analysis form fields.)

Fundamental Tones displays and edits information for top level tones in the circuit.

View Harmonics displays output harmonics. (Displays additional fields to specify harmonics.)

Accuracy Defaults quickly adjusts simulation parameters.

Sweep selects swept QPSS analysis. (Displays additional fields to specify sweep.)

Enabled includes this analysis in the next simulation.

Options displays the QPSS Options form.

Modifications to QPSS Form to Display and Select Harmonics

View Harmonics displays available harmonics and *From* and *To* fields for selection.

Modifications to QPSS Form for Swept QPSS Analysis

Sweep, **Sweep Range**, **Sweep Type**, and **Add Specific Points** specifies how the QPSS sweep is performed.

Envelope Following (Envlp) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *envlp* to display the Envelope Following Analysis form fields.)

Clock Name and **Select Clock Name** Selects the clock signal for the Envlp analysis.

Stop Time Specifies the end time for the Envlp analysis.

Output Harmonics selects and defines output harmonics.

Accuracy Defaults quickly adjusts simulation parameters.

Enabled includes this analysis in the next simulation.

Options displays the Envlp Options form.

Periodic AC (PAC) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *pac* to display the Periodic AC Analysis form fields.)

PSS Beat Frequency (Hz) displays the *Beat Frequency* for the associated PSS analysis.

Sweeptype, **Frequency Sweep Range**, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Sidebands selects the set of periodic small-signal output frequencies of interest.

Enabled includes this analysis in the next simulation.

Options displays the PAC Options form.

Modifications to PAC Form for Swept PSS Analysis

Sweeptype, Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept PSS analysis.

Periodic Noise (Pnoise) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *pnoise* to display the Periodic Noise Analysis form fields.)

PSS Beat Frequency (Hz) displays the *Beat Frequency* for the associated PSS analysis.

Sweeptype, **Frequency Sweep Range**, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Sidebands selects the set of periodic small-signal output frequencies of interest.

Output, **Input Source**, and **Reference Side-Band** selects the output, noise generator, and reference sidebands for the PNOISE analysis. (Displays additional fields.)

Noise Type selects the type of noise to compute. (Active only when PSS analysis is *not* swept.)

Enabled includes this analysis in the next simulation.

Options displays the Pnoise Options form.

Modifications to Pnoise Form for Swept PSS analysis

Sweeptype, Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept PSS analysis.

(**Noise Type** is not active when PSS analysis is swept.)

Periodic Transfer Function (PXF) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *pxf* to display the Periodic Noise Analysis form fields.)

PSS Beat Frequency (Hz) displays the *Beat Frequency* for the associated PSS analysis.

Sweeptype, **Frequency Sweep Range**, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Sidebands selects the set of periodic small-signal output frequencies of interest.

Output selects the output.

Enabled includes this analysis in the next simulation.

Options displays the PXF Options form.

Modifications to PXF Form for Swept PSS Analysis

Sweeptype, Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept PSS analysis.

Periodic S-Parameter (PSP) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *psp* to display the Periodic S-Parameter Analysis form fields.)

Sweeptype, **Frequency Sweep Range**, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Select Ports selects the active ports for the PSP analysis.

Do Noise selects whether or not to measure noise during the PSP analysis.

Enabled includes this analysis in the next simulation.

Options displays the PSP Options form.

Modifications to PSP Form for Swept PSS Analysis

Sweeptype, Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept PSS analysis.

Quasi-Periodic Noise (QPnoise) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *qpnoise* to display the Quasi Periodic Noise Analysis form fields.)

Frequency Sweep Range, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Sidebands selects the set of periodic small-signal output frequencies of interest.

Output, **Input Source**, **Reference Side-Band**, and **refsidebandoption** selects the output, noise generator, reference sidebands, and refsidebandoption for the QPnoise analysis. (Displays additional fields.)

Enabled includes this analysis in the next simulation.

Options displays the QPnoise Options form.

Modifications to QPnoise Form for Swept QPSS Analysis

Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept QPSS analysis.

Quasi-Periodic AC (QPAC) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *qpac* to display the Quasi Periodic AC Analysis form fields.)

Frequency Sweep Range, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Sidebands selects the set of periodic small-signal output frequencies of interest.

Enabled includes this analysis in the next simulation.

Options displays the QPnoise Options form.

Modifications to QPAC Form for Swept QPSS Analysis

Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept QPSS analysis.

Quasi-Periodic Transfer Function (QPXF) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *qpxf* to display the Quasi-Periodic XF Analysis form fields.)

Frequency Sweep Range, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Sidebands selects the set of periodic small-signal output frequencies of interest.

Output selects the output.

Enabled includes this analysis in the next simulation.

Options displays the QPXF Options form.

Modifications to QPXF Form for Swept QPSS Analysis

Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept QPSS analysis.

Quasi-Periodic S-Parameter (QPSP) Choosing Analysis Form

Analysis selects the type of analysis to set up. (Highlight *qpsp* to display the Quasi Periodic S-Parameter Analysis form fields.)

Sweeptype, **Frequency Sweep Range**, **Sweep Type**, and **Add Specific Points** sets up the sweep for the small signal analysis.

Select Ports selects the active ports for the QPSP analysis.

Do Noise selects whether or not to measure noise during the QPSP analysis.

Enabled includes this analysis in the next simulation.

Options displays the QPSP Options form.

Modifications to QPSP Form for Swept QPSS Analysis

Sweeptype, Frequency Sweep Range, **Single-Point**, and **Freq** specify the frequency for the small signal analysis that follows each swept QPSS analysis.

Option Forms

PSS Analysis Options Form

Time Step Parameters define the time step used for the PSS analysis.

Initial Condition Parameters define the initial conditions for the PSS analysis.

Convergence Parameters provide an initial transient solution and minimum capacitance for the PSS analysis.

State File Parameters provides the locations of files associated with the PSS analysis.

Integration Method Parameters define the integration method used for the PSS analysis.

Accuracy Parameters define the level of accuracy to use for the PSS analysis. Enable/disable/refine use of the Finite difference method of the PSS analysis.

Annotation Parameters define statistics and other information recorded and displayed for the PSS analysis.

Output Parameters defines information related to the results of the PSS analysis.

Newton Parameters defines information about the Newton iterations and previous DC solution for the PSS analysis.

Simulation Interval Parameters define the start time for the initial transient analysis for this PSS analysis.

QPSS Analysis Options Form

Time Step Parameters define the time step used for the QPSS analysis.

Initial Condition Parameters define the initial conditions for the QPSS analysis.

Convergence Parameters provide an initial transient solution and minimum capacitance for the QPSS analysis.

State File Parameters provides the locations of files associated with the QPSS analysis.

Integration Method Parameters define the integration method used for the QPSS analysis.

Accuracy Parameters define the level of accuracy to use for the QPSS analysis.

Annotation Parameters define statistics and other information recorded and displayed for the QPSS analysis.

Output Parameters defines information related to the results of the QPSS analysis.

Newton Parameters defines information about the Newton iterations and previous DC solution for the QPSS analysis.

Simulation Interval Parameters define the start time for the initial transient analysis for this QPSS analysis.

Envelope Following Analysis Options Form

Simulation Interval Parameters defines the start time, output start time, and the stabilization time period for the Envlp analysis.

Simulation Bandwidth Parameters define the modulation bandwidth for the Envlp analysis.

Time Step Parameters define the time step and the outer envelope size used for the Envlp analysis.

Initial Condition Parameters define the initial conditions for the Envlp analysis.

Convergence Parameters provide an initial transient solution and minimum capacitance for the Envlp analysis.

State File Parameters provides the locations of files associated with the Envlp analysis.

Integration Method Parameters define the integration method used for the Envlp analysis.

Accuracy Parameters define the level of accuracy to use for the Envlp analysis.

Annotation Parameters define statistics and other information recorded and displayed for the Envlp analysis.

Output Parameters defines information related to the results of the Envlp analysis.

Newton Parameters defines information about the Newton iterations and previous DC solution for the Envlp analysis.

Periodic Small-Signal Analyses Options Forms

Convergence Parameters provide convergence information for the small signal analysis.

Annotation Parameters define statistics and other information recorded and displayed for the small signal analysis.

Output Parameters defines information related to the results of the small signal analysis.

Quasi-Periodic Small-Signal Analyses Options Forms

Convergence Parameters provide convergence information for the small signal analysis.

Annotation Parameters define statistics and other information recorded and displayed for the small signal analysis.

Output Parameters defines information related to the results of the small signal analysis.

Direct Plot Form

See Direct Plot Form on page 183 for information on using the Direct Plot Form.

See Field Descriptions for the Plot Form on page 191 for descriptions of fields on the Direct Plot Form.

ACPR Wizard

Clock Name Specifies the clock signal for the Envlp analysis

How to Measure Selects whether to measure ACPR for a single net or between two differential nets.

Channel Definitions Select one of three channel definitions: Custom, IS-95 or W-CDMA.

Main Channel Width Specify channel width in Hz.

Adjacent frequencies list box Specifies the adjacent channel frequencies. Us the Add, Change and Delete buttons and the editing fields to modify adjacent frequencies.

Simulation Control Enter the **Stabilization Time** and **Repetitions** in the adjacent fields.

Resolution Bandwidth (Hz) and **Calculate** Click *Calculate* to determine the Resolution Bandwidth.

Windowing Function Selects the windowing function to use.

3

Setting Up for the Examples

This chapter tells you how to set up your software so you can follow the instructions and run the examples in the chapters about simulating specific components.

Setting Up the Software

Before you perform the SpectreRF simulator analyses, you must set up the component files and start the Cadence® software.

Copying the SpectreRF Simulator Examples

Copy the *rfExamples* library into your account so that you can edit the sample design.

The library is located at

`<CDSHOME>/tools/dfII/samples/artist/rfExamples`

where *`CDSHOME`* is the installation directory for your Cadence software.

Setting Up the Cadence Libraries

The Cadence Libraries are defined in the UNIX text file *`cds.lib`*. You can edit this file in a UNIX shell window or by using the Library Path Editor while running the Cadence software.

Using a UNIX Shell Window

To set up the libraries in a UNIX shell window, use the following procedure:

1. In a UNIX shell window, open the `cds.lib` file for editing using *vi*, *emacs* or a similar text editor.

   The *`cds.lib`* file is in your installation directory.

2. In the *cds.lib* file, specify definitions for the *basic*, *analogLib*, *sample,* and *spectreSModels* libraries. In addition, define a user library where the sample circuits can be tested. The definitions in the *cds.lib* file look like the following:

```
DEFINE rf_test my_dir/rf_test
DEFINE basic <CDSHOME>/tools/dfII/etc/cdslib/basic
DEFINE analogLib <CDSHOME>/tools/dfII/etc/cdslib/artist/analogLib
DEFINE sample <CDSHOME>/tools/dfII/samples/cdslib/sample
DEFINE spectreSModels <CDSHOME>/tools/dfII/samples/artist/
                        spectreSModels
```

   You can label your test library any name you choose. This example assumes that you have called the library *rf_test*. The name *my_dir* represents the directory into which you copied the design library.

   You must use the names *basic*, *sample*, *analogLib,* and *spectreSModels*. You cannot rename those four libraries.

Using the Library Path Editor

To access the Library Path Editor, use the following procedure.

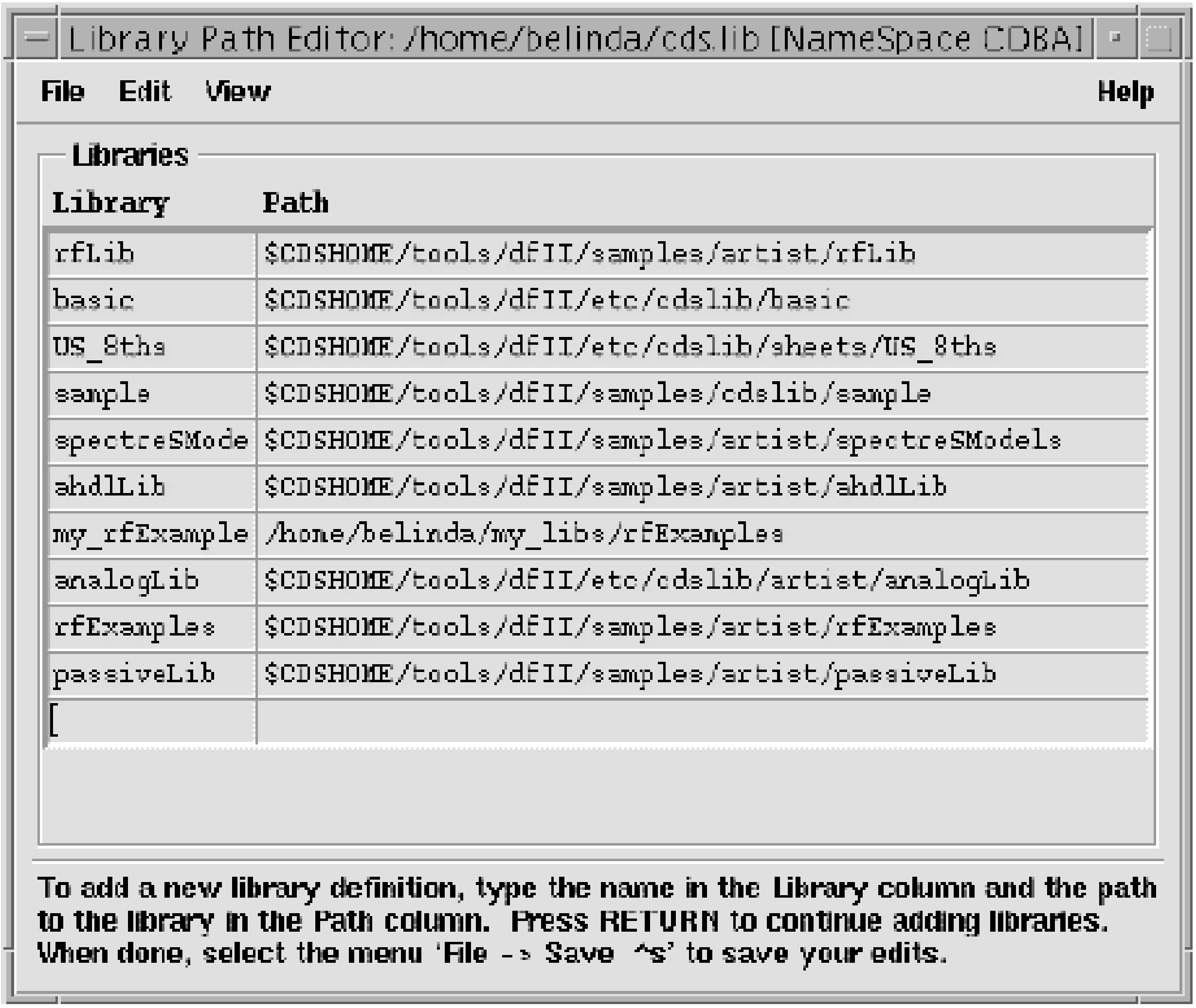
1. In a UNIX window, type `icms` to start the Cadence software.

   The Command Interpreter Window (CIW) appears.

2. In the CIW, choose *Tools – Library Path Editor*.

   The Library Path Editor form appears.

3. In the Library Path Editor, follow the instructions at the bottom of the form.



4. Type in each required library and its associated path. See the previous section for a list of the required libraries.

Library Conversion of SpectreS Libraries

If you have libraries that you have been using with SpectreS simulation, you need to convert them before you can use them with Spectre direct simulation. To learn more about the necessary conversion, see the information about library conversion in the *Compatibility Guide*.

4

Simulating Mixers

To use this chapter, you must be familiar with the SpectreRF simulator analyses as well as know about mixer design. For more information about the SpectreRF simulator analyses, see Chapter 1.

The SpectreRF simulator can simulate circuits, such as mixers, that show frequency conversion effects. This chapter uses a commercially available integrated circuit mixer, the ne600p, configured as a down converter to illustrate how the SpectreRF simulator can determine the characteristics of your design. The ne600p circuit examples illustrate both the capabilities and the requirements of the SpectreRF simulator. The examples show you how to perform a simulation with an RF to IF ratio of almost 12 to 1. The examples also show you how to be sure that all time-varying independent signal sources have a common integral multiple (40 MHz in this case).

In the mixer examples, you plot the following nonlinear characteristics of the ne600p mixer:

Measurements	Analyses
Harmonic distortion	PSS
Noise figure	PSS and Pnoise
Noise figure	PSS and PSP
Periodic S-parameters	PSS and PSP
Conversion gain	PSS and PXF
Power supply rejection	PSS and PXF
1 dB compression point	Swept PSS
Intermodulation distortion	Swept PSS
Third-order intercept point	Swept PSS and PAC
Intermodulation distortion	QPSS
Noise figure	QPSS and QPnoise

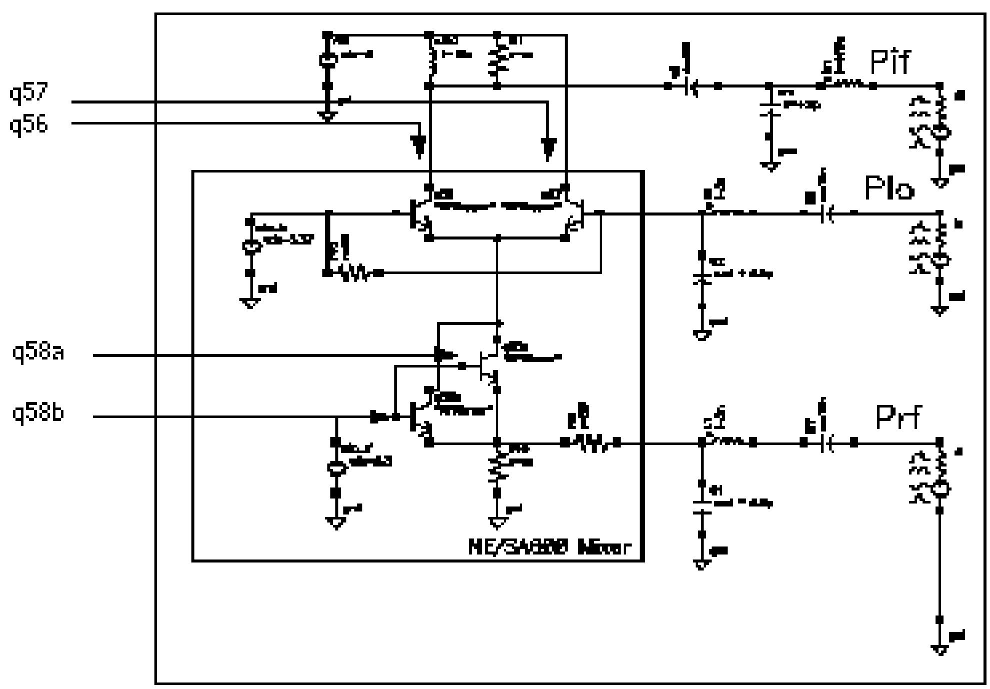
In order to make these measurements, you run one of the following combination of analyses.

- A Periodic Steady-State analysis (PSS), and the following periodic small-signal analyses: Periodic AC (PAC) analysis, Periodic Transfer Function (PXF) analysis, Periodic Noise (Pnoise) analysis, and Periodic S-Parameter (PSP) analysis.
- A Quasi-Periodic Steady-State (QPSS) Analysis, and the quasi-periodic small-signal analysis: Quasi-Periodic Noise (QPnoise) analysis.

The ne600p Mixer Circuit

The ne600p integrated circuit is commercially available and is designed for low power communication systems from 800-1200 MHz. The receiver is a single balanced mixer containing four bipolar transistors. the schematic for the ne600p mixer circuit is shown in Figure 4-1.

Figure 4-1 Schematic for the ne600p Mixer Circuit



The high-level local oscillator (LO) signal (~0 dBm at 50 Ohms) enters through the emitter follower q57. The transistor q56 is in common base configuration. The q56 emitter follows the q57 emitter. When the LO signal goes high, both emitters also go high and the B-E voltage on q56 decreases. As a result, the two BJTs form a balanced pair.

The collector currents of q56 and q57 are 180 degrees out of phase. The q58 pair feed the low-level (-30 to -20 dBm) RF signal to the balanced pair. The IF signal is drawn from the open collector output of q56 through a bandpass passive network.

The following tables list some measured values for different aspects of the ne600p mixer.

Measurement	Measured
LO frequency (Hz)	1 GHz
RF frequency (Hz)	920 MHz
IF frequency (Hz)	80 MHz
LO power	0 dBm
RF power	-30 dBm
Conversion gain	-2.6 dB
Noise figure	16 dB
Input 1dB compression point	-4 dBm
Input IP3 (from swept power)	6 dBm
Input IP3 (from PAC analysis)	6 dBm

Design Variable	Default Value
prf (RF power)	-30 dBm
vlo (LO magnitude)	316.2 m
frf (RF frequency)	920 MHz
flo (LO frequency)	1 GHz

Before you use the ne600p circuit from the sample library, be sure that the *frf* design variable is set to the appropriate value, either 920 MHz or 900 MHz, depending on the simulation. If you adjust the RF frequency to the center of the IF tuned circuit, approximately 100 MHz, the conversion gain becomes -2.57dB and the other values change slightly

Simulating the ne600p Mixer

Before you start, perform the setup procedures described in Chapter 3, “Setting Up for the Examples.”

Opening the ne600p Mixer Circuit in the Schematic Window.

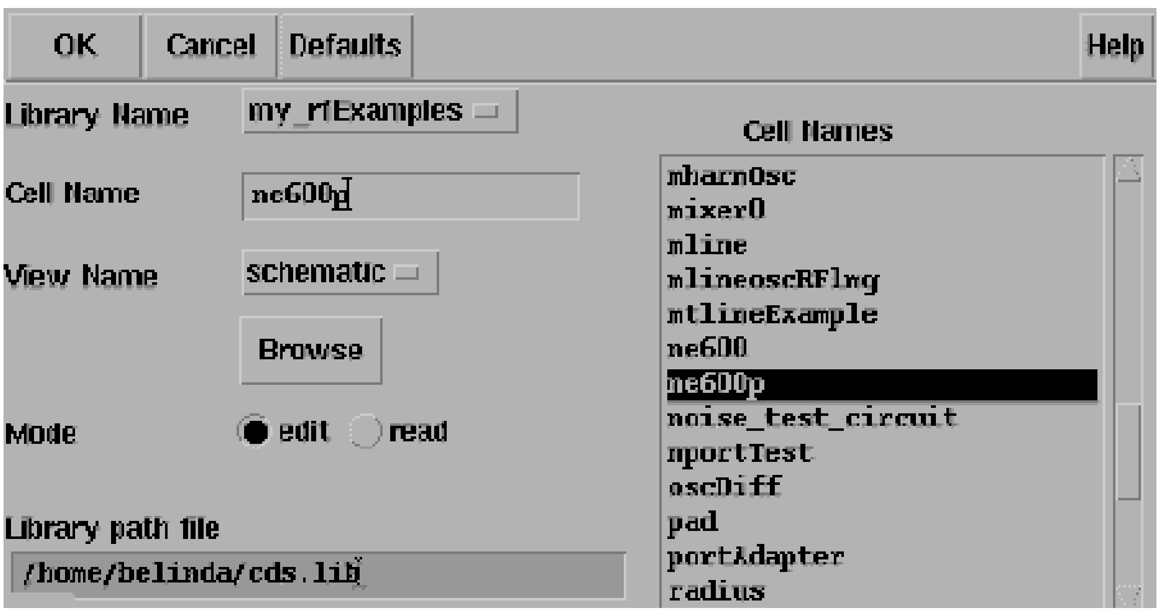
1. In the CIW, choose *File – Open*.

The Open File form appears.

2. In the Open File form, choose *my_rfExamples* in the *Library Name* menu. Choose the editable copy of the rfExamples library you created as described in Chapter 3, “Setting Up for the Examples.”

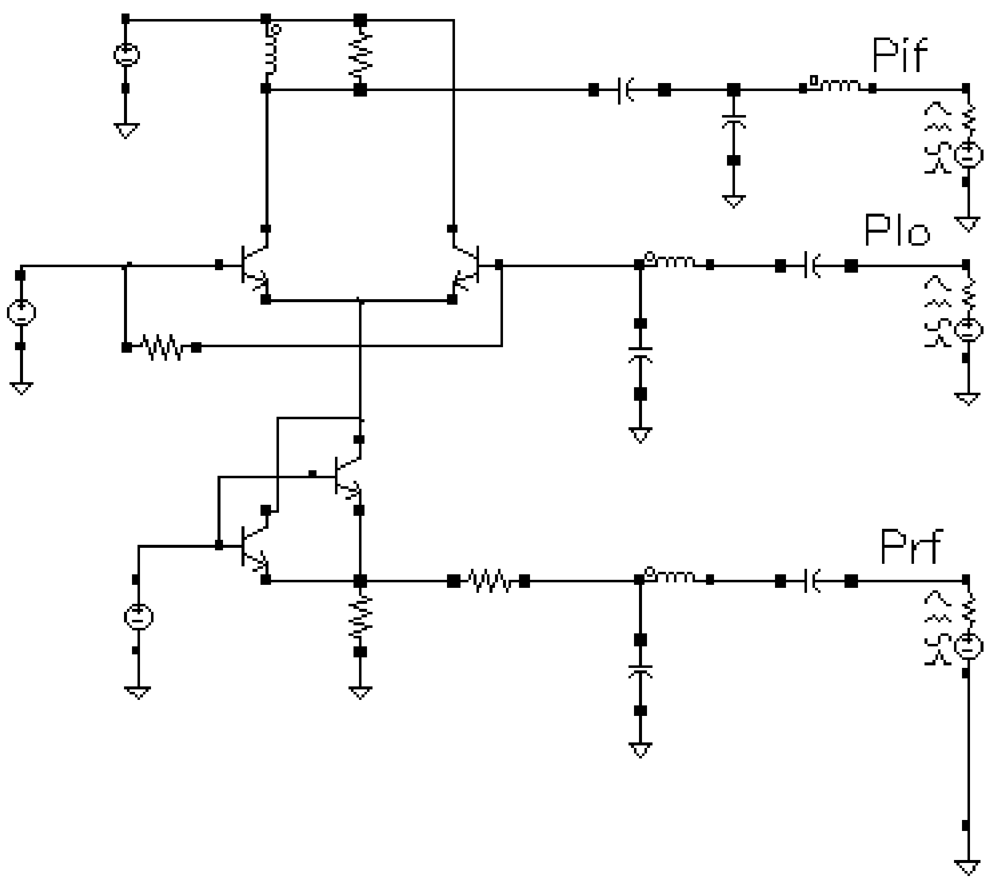
3. Choose *ne600p* in the *Cell Names* list box.

The completed Open File form appears like the one below.



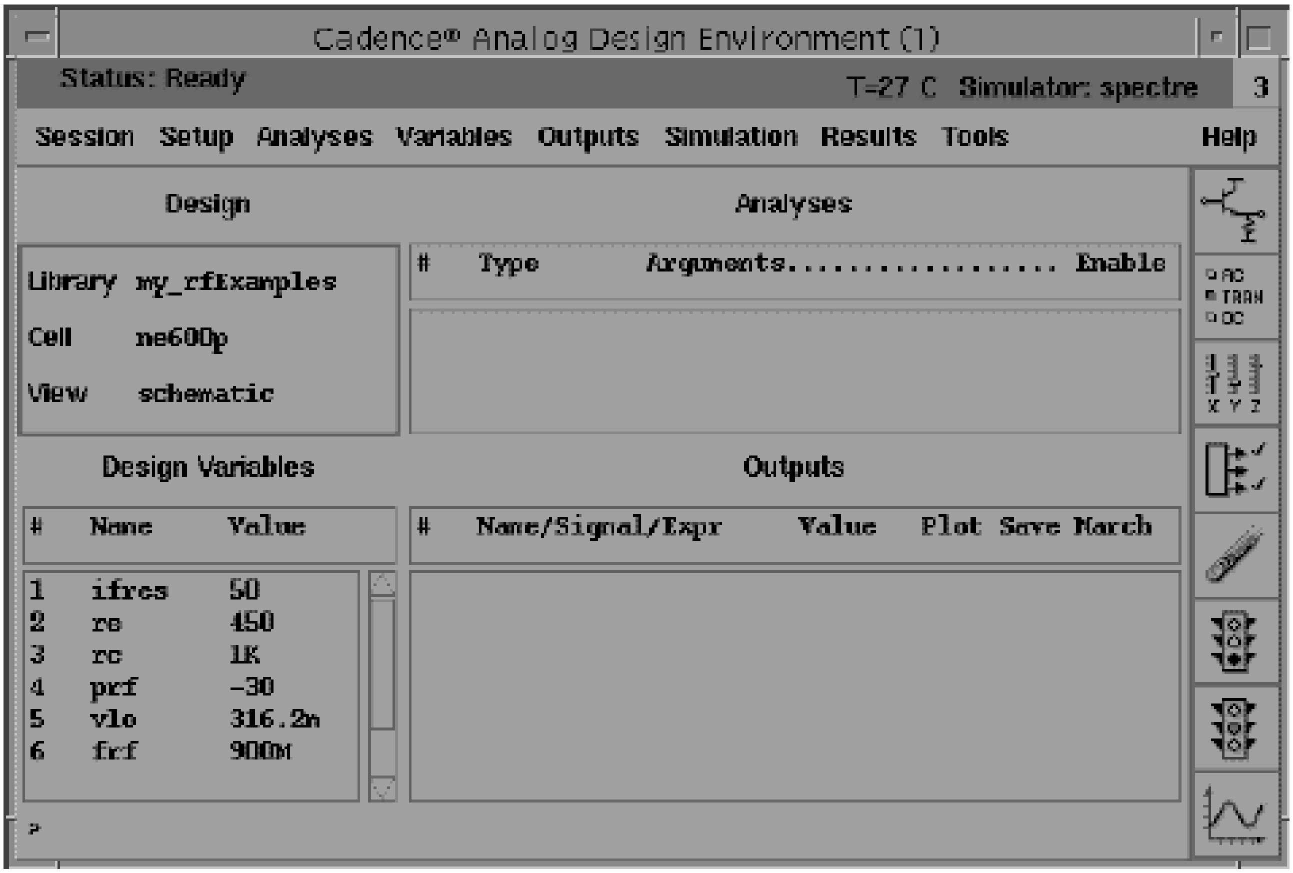
4. Click *OK*.

The Schematic window for the *ne600p* mixer appears.



5. In the Schematic window, choose *Tools– Analog Environment*.

The Simulation window opens. This window is also called the Cadence® Analog Circuit Design Environment.



You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing the *ne600p* in the Choosing Design form.

Choosing Simulator Options

1. Choose *Setup – Simulator/Directory/Host* in the Simulation window.

   The Simulator/Directory/Host form appears.

2. In the Simulator/Directory/Host form, specify the following:

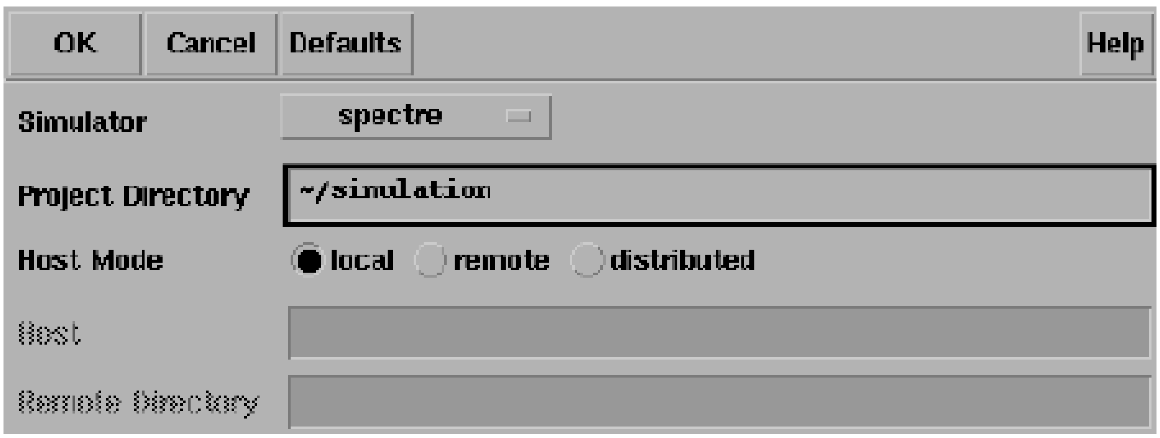
   a. Choose *spectre* for the *Simulator*.

   b. Type the name of the project directory, if necessary.

c. Highlight the *local* or the *remote* button to specify the *Host Mode*.

For remote simulation, type the name of the host machine and the remote directory in the appropriate fields.

The completed form appears like the one below.

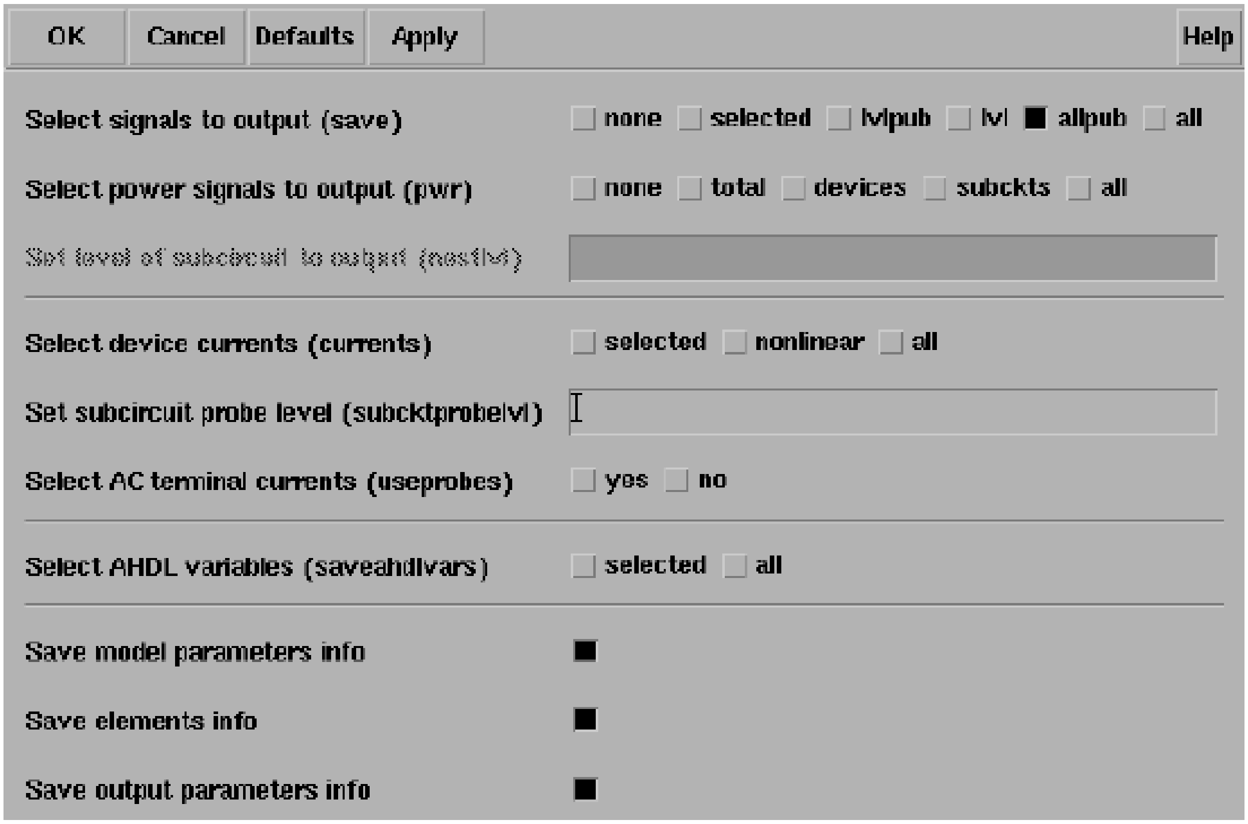


3. In the Simulator/Directory/Host form, click *OK*.

4. In the Simulation window, choose *Outputs – Save All.*

The Save Options form appears.

5. In the *Select signals to output section*, be sure *allpub* is highlighted.



Setting Up Model Libraries

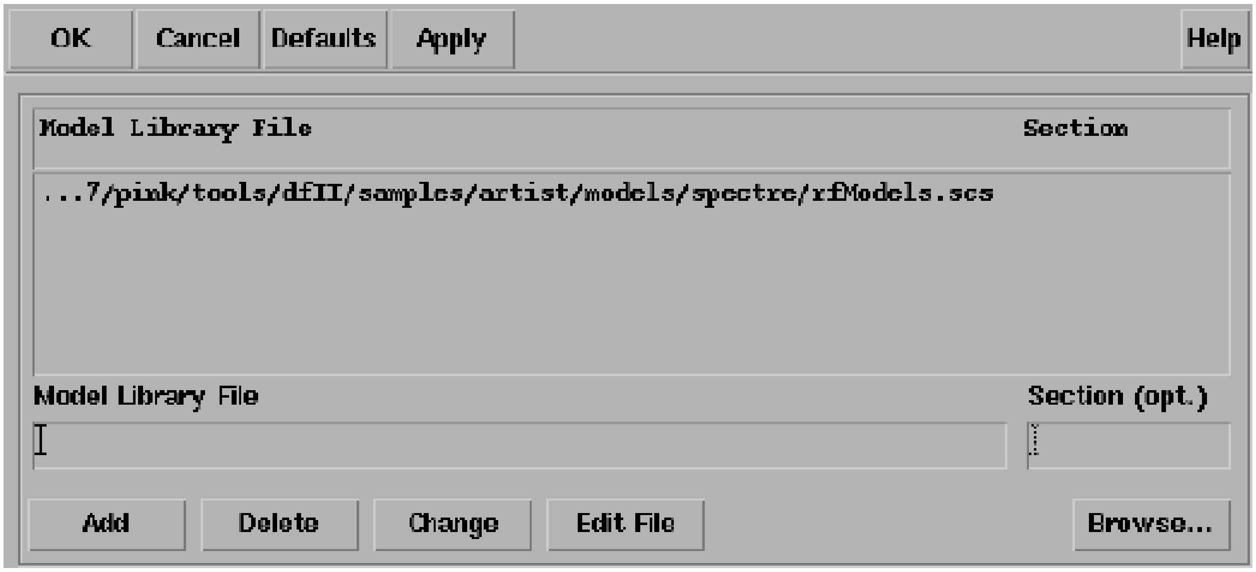
1. In the Simulation window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs.`

3. In the Model Library Setup form, click on *Add*.

The completed form appears like the one below.



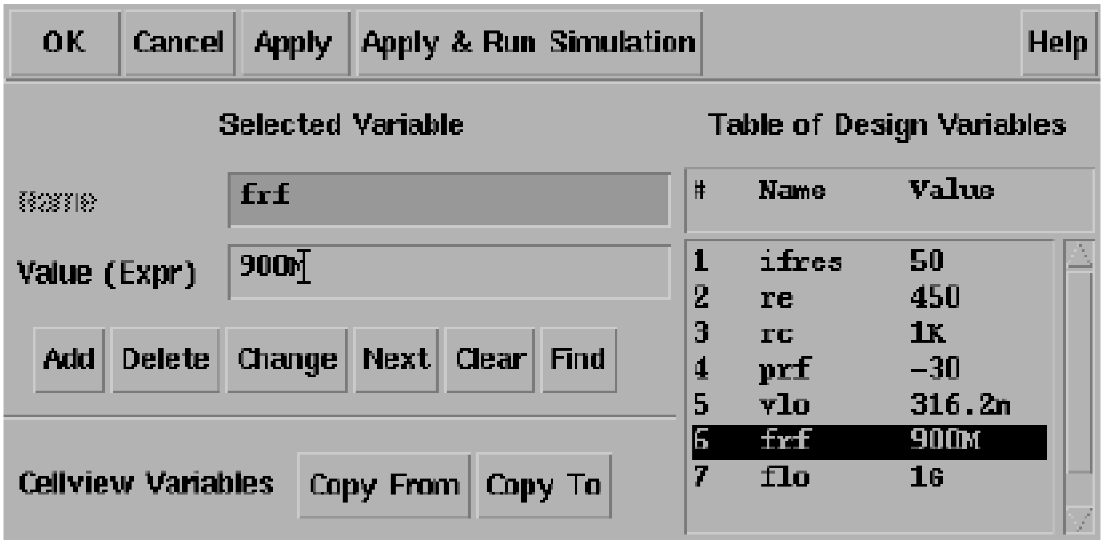
4. In the Model Library Setup form, click *OK*.
5. Disable any analyses you ran previously.

Setting Design Variables

In the Simulation window, use the following procedure to set up the design variables `frf` and `flo` to the values required for each simulation. (See the description of each simulation for the required value.)

1. Highlight the `frf` in the *Design Variable* list box.
2. Choose *Variables – Edit.*

The Editing Design Variables form appears.



3. In the *Value (Expr)* field, type `900M` (or `920M`)for the value of `frf` and click *Change*.
4. Highlight `flo` in the *Table of Design Variables* list box.
5. In the *Value (Expr)* field, type `1G` for the value of `flo` and then click *Change.*
6. In the Editing Design Variables form, click *OK*.

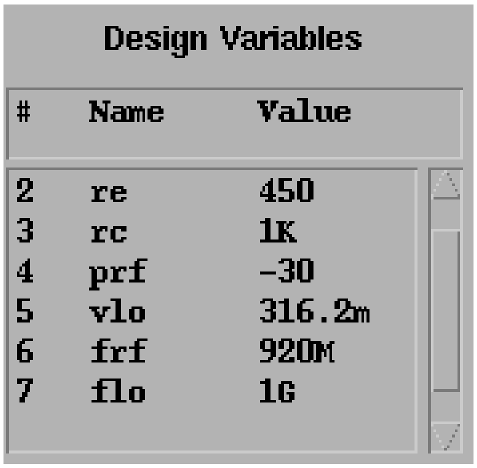
Harmonic Distortion Measurement with PSS

In this first example, a single PSS analysis determines the harmonic distortion of the mixer.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

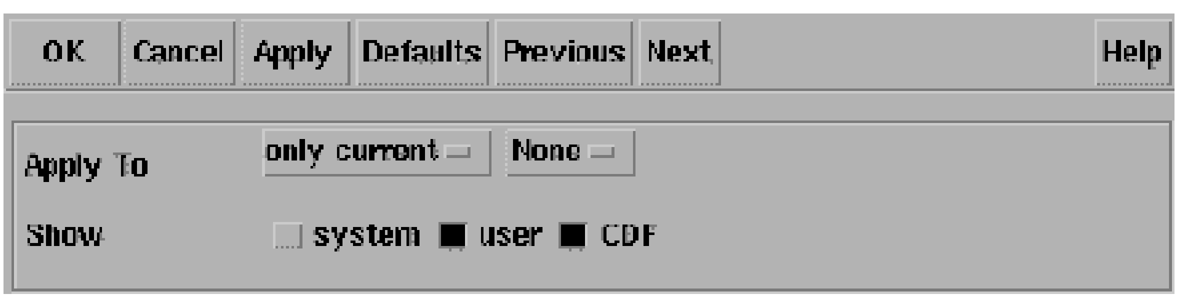


Editing the Schematic

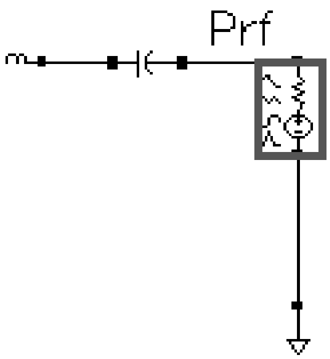
A critical part of this analysis is the correct use of the programmable voltage sources in the `ne600p` circuit. The RF voltage source is based on the *port* sample component. You must change the behavior of this component for each analysis.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the *rf* voltage source and modify the schematic for this simulation.



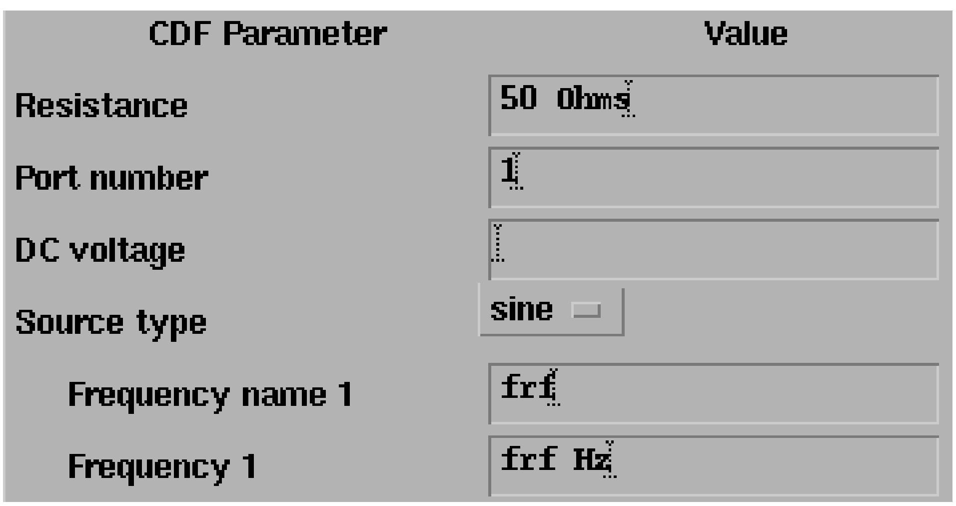
2. In the Schematic window, click on the *rf* voltage source.



The Edit Object Properties form changes to display information for the voltage source.

3. In the Edit Object Properties form, be sure that *sine* is specified as the *Source type*.

   A portion of the completed Edit Object Properties form for the rf port appears like the one below:



CDF Parameter	Value
Resistance	50 Ohms
Port number	1
DC voltage	
Source type	sine
Frequency name 1	frf
Frequency 1	frf Hz

4. In the Edit Object Properties form, click *OK*.
5. In the Schematic window, choose *Design – Check and Save.*

Setting Up the PSS Analysis

1. Choose *Analyses – Choose* in the Simulation window.

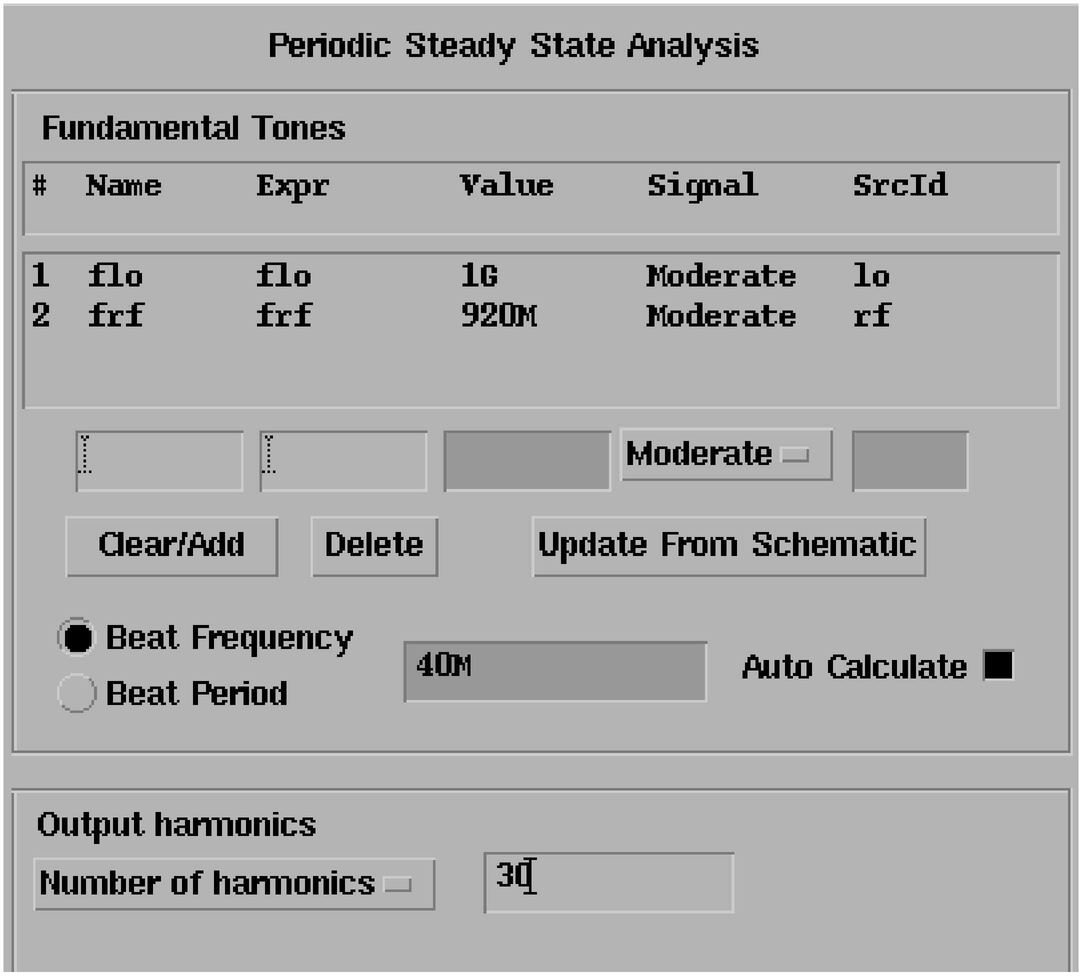
The Choosing Analyses form appears.

2. In the Choosing Analyses form, click on *pss.*
3. In the *Fundamental Tones* list box, *Beat Frequency* is highlighted by default. Be sure *Auto Calculate* is also highlighted.

   The *Beat Frequency* is now displayed. The `40 M` value is the minimum period for which both RF at `920 MHz` and the LO at `1 GHz` are periodic (or for which RF and LO are integer multiples of the fundamental frequency).
4. In the *Output harmonics* cyclic field, choose *Number of harmonics*. Type `30` in the *number of harmonics* field.

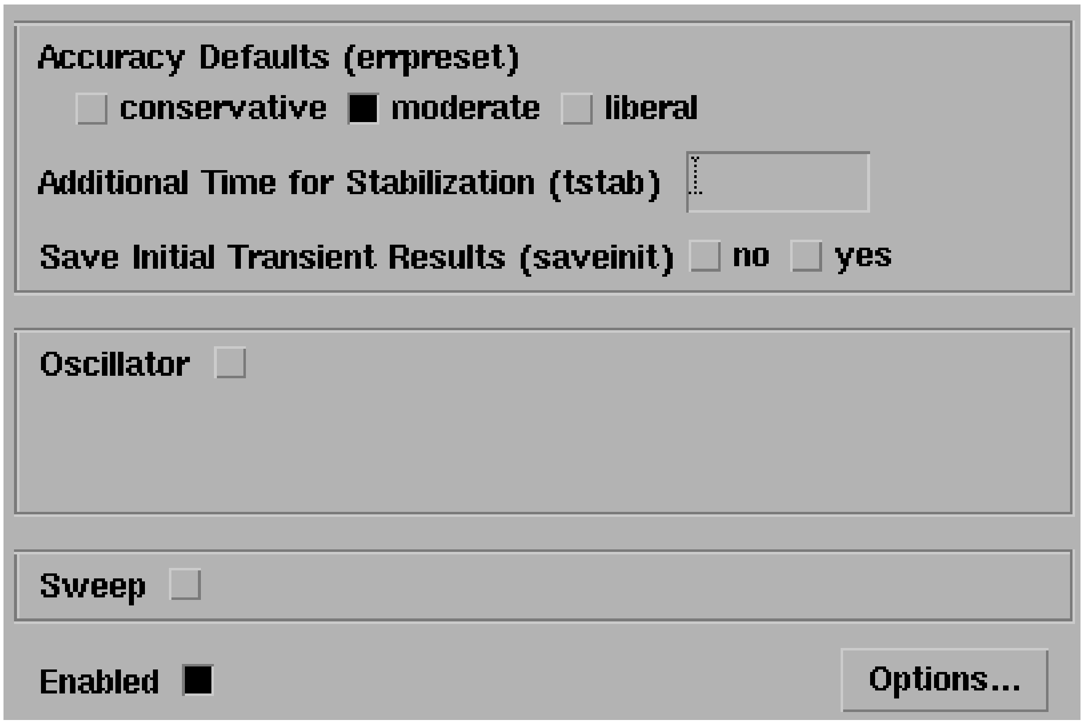
   This entry expands the plotted frequency range to include the areas of interest around both `40 MHz` and `1 GHz`. (`30` x `40 MHz` equals `1.2 GHz`, or `0` to `1.2 GHz`, and includes all important frequencies.)

The top of the PSS Choosing Analyses form is as follows.



5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
6. Verify that *Enabled* is highlighted

The bottom of the PSS Choosing Analyses form is as follows.



7. Click *OK.*

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting and Calculating Harmonic Distortion

➤ Choose *Results – Direct Plot – Main Form* in the Simulation window.

The Direct Plot form appears.

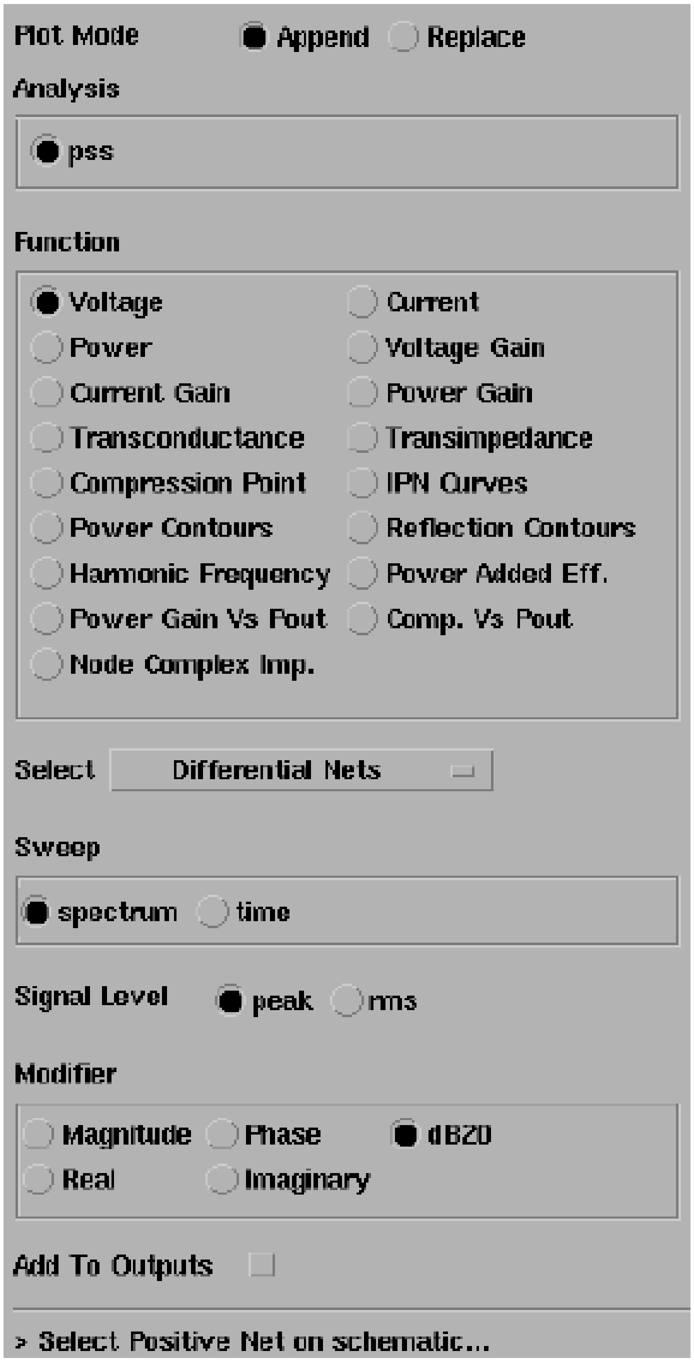
In the Direct Plot form, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *pss* for *Analysis*.

3. Highlight *Voltage* for *Function*.
4. Select *Differential Nets* in the *Select* cyclic field to plot voltage against frequency.

   (Notice that the message at the bottom of the form changes.)
5. Highlight *spectrum* for *Sweep*.
6. Highlight *peak* for *Signal Level*.

7. Highlight *dB20* for *Modifier*.



8. To plot the voltage against frequency, click on the *Pif* and *Prf* wires in the Schematic Window.

a. Following the instructions at the bottom of the Direct Plot Form

`Select Positive Net On Schematic...`

Select the positive net, *Pif*, on the schematic.

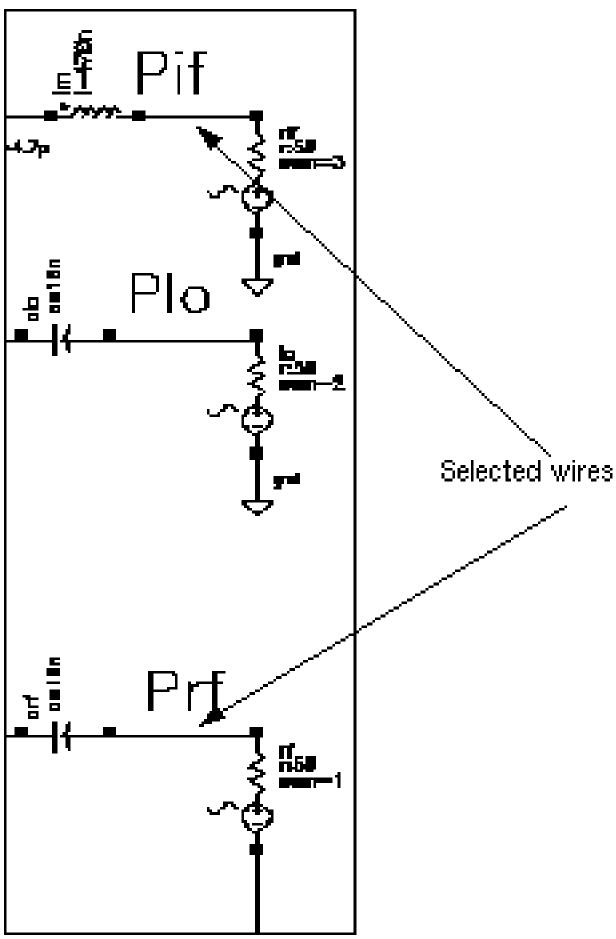
Pif is highlighted in the schematic window and the instructions at the bottom of the Direct Plot form change.

b. Following the instructions at the bottom of the Direct Plot Form

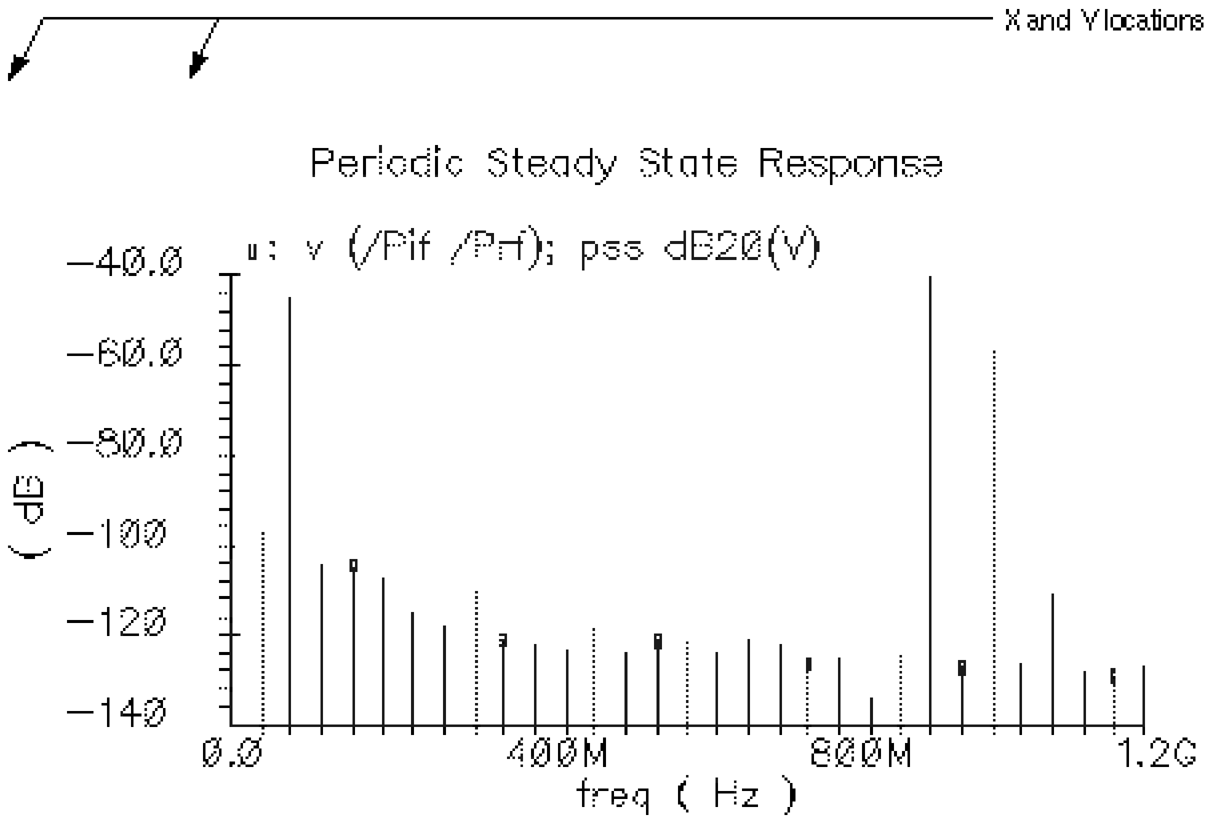
Select Negative Net on schematic...

Select the negative net, *Prf*, on the schematic.

Prf is highlighted in the schematic window.



The Waveform Window display appears like the one below.



9. Measure `Vif` at `80`, `160`, and `240 MHz` by placing the cursor at the top of the appropriate lines in the plot. The X and Y values appear at the top left corner of the Waveform window.

10. Calculate the total harmonic distortion (THD) with the following formula:

$$THD = dB10[(h4)^2 + (h6)^2] - dB20(h2)$$

where

`h2` is `80 MHz` (`Vif` is -45.43 dB)

`h4` is `160 MHz` (`Vif` is -104.8 dB)

`h6` is `240 MHz` (`Vif` is -114.8 dB)

The value of `THD` is `-61.0 dB`.

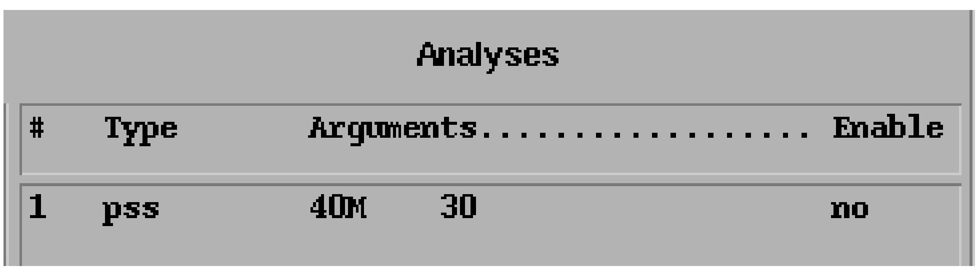
Noise Figure Measurement with PSS and Pnoise

You can combine a PSS analysis with a small-signal Pnoise analysis to determine the noise figure for the ne600p circuit.

Pnoise analysis calculates the total noise at the output of the circuit. The equation for the Noise Figure is given in the SpectreRF Theory document. The SpectreRF Pnoise analysis computes the single sideband noise figure (`-1` in this case). The total noise can vary with the number of harmonics you choose because each harmonic contributes a noise component.

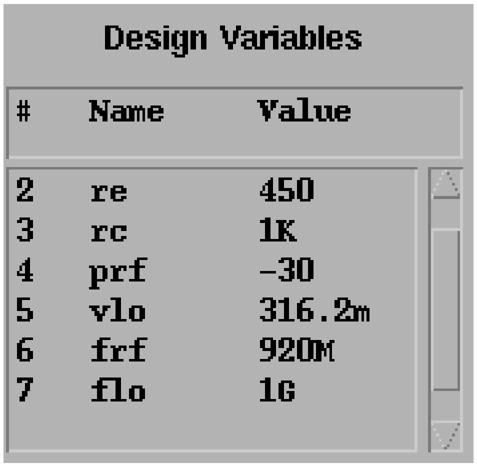
Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)



In this figure, the analysis is disabled.

4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)



Editing the Schematic

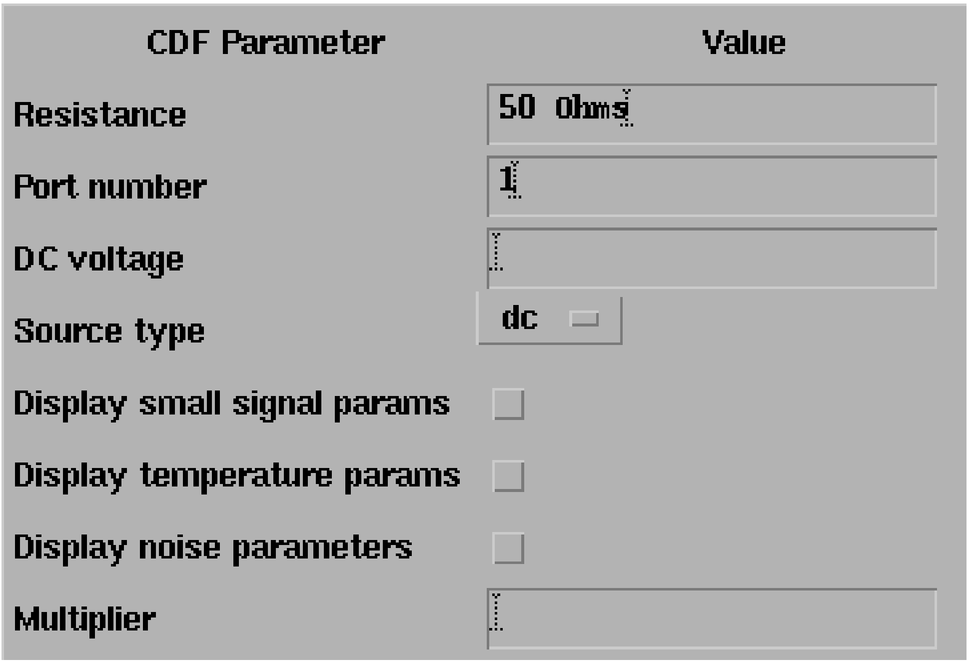
Modify the schematic to be sure the PSS analysis is the response of the ne600p mixer to *only* the LO signal.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic Window, click on the *rf* voltage source.

3. In the Edit Object Properties form, choose *dc* as the *Source type*, if necessary, and click *OK*.



4. In the Schematic window, choose *Design – Check and Save.*

Setting up the PSS and Pnoise Analyses

➤ In the Simulation window, choose *Analyses – Choose.*

The Choosing Analyses form appears.

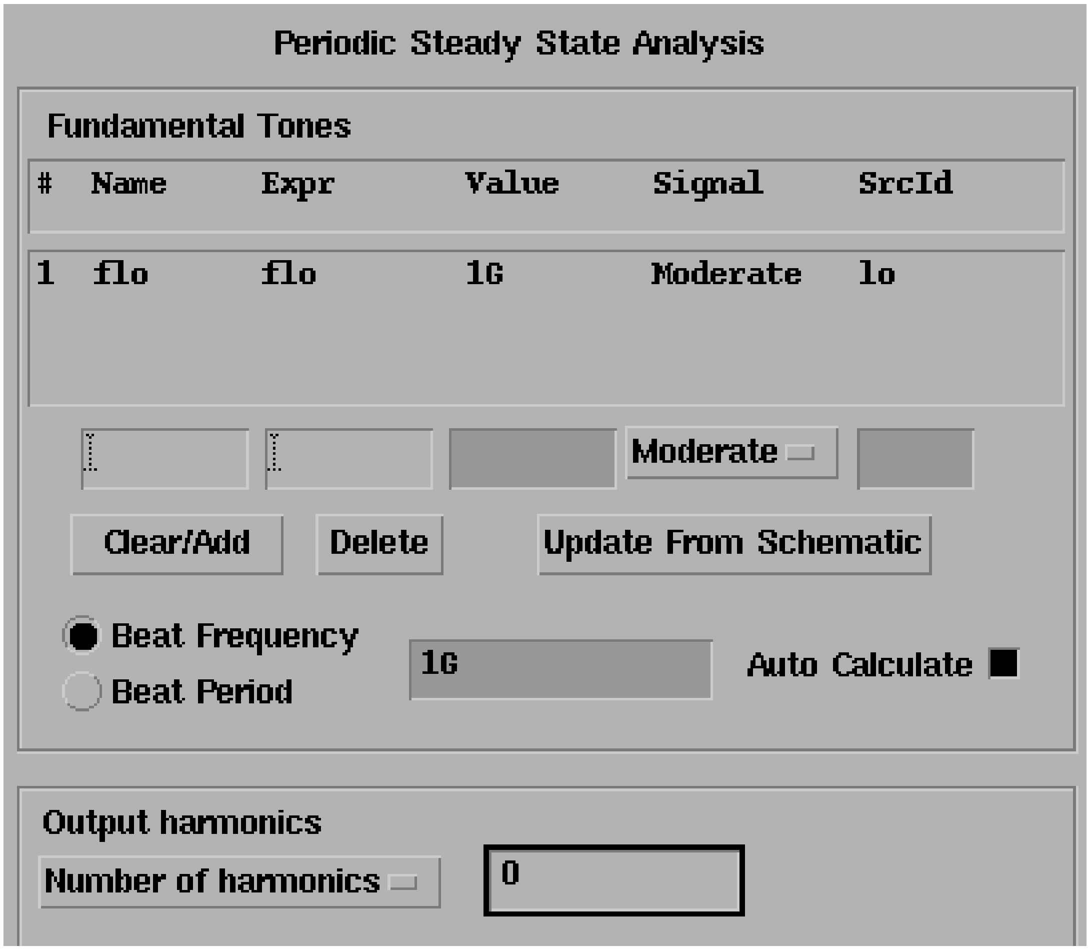
Setting Up the PSS Analysis

1. In the Choosing Analyses form, click on *pss* for the *Analysis*.
2. At the lower right corner of the Fundamental Tones section, highlight *Auto Calculate*.

   The *Beat Frequency* is now displayed as `1G`. The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `0` in the field.

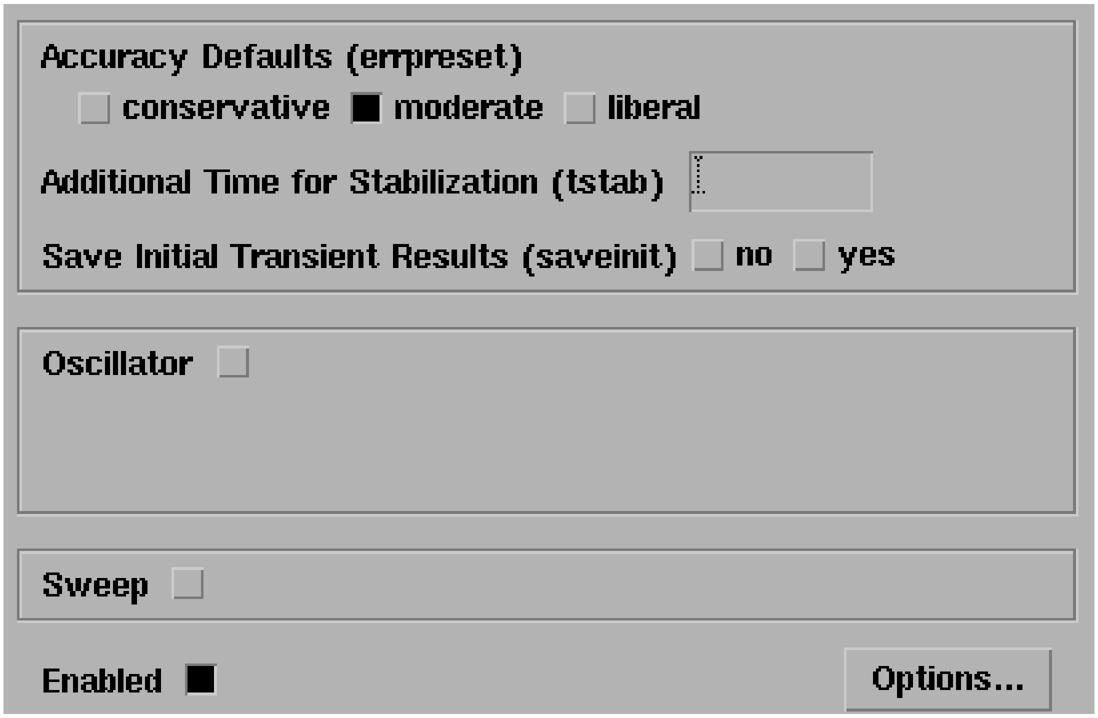
   Pnoise requires that you set the number of harmonics value to `0` to determine the circuit's response to LO only.

The top of the PSS Choosing Analyses form appears as follows.



4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight *Enabled*.

The bottom of the PSS Choosing Analyses form appears as follows.



Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, highlight *pnoise.*

   The Choosing Analyses form changes to let you specify data for a *Pnoise* analysis.

2. In the *Frequency Sweep Range (Hz*) cyclic field, choose *Start-Stop*.

3. Type `1K` in the *Start field* and `2G` in the *Stop* field.

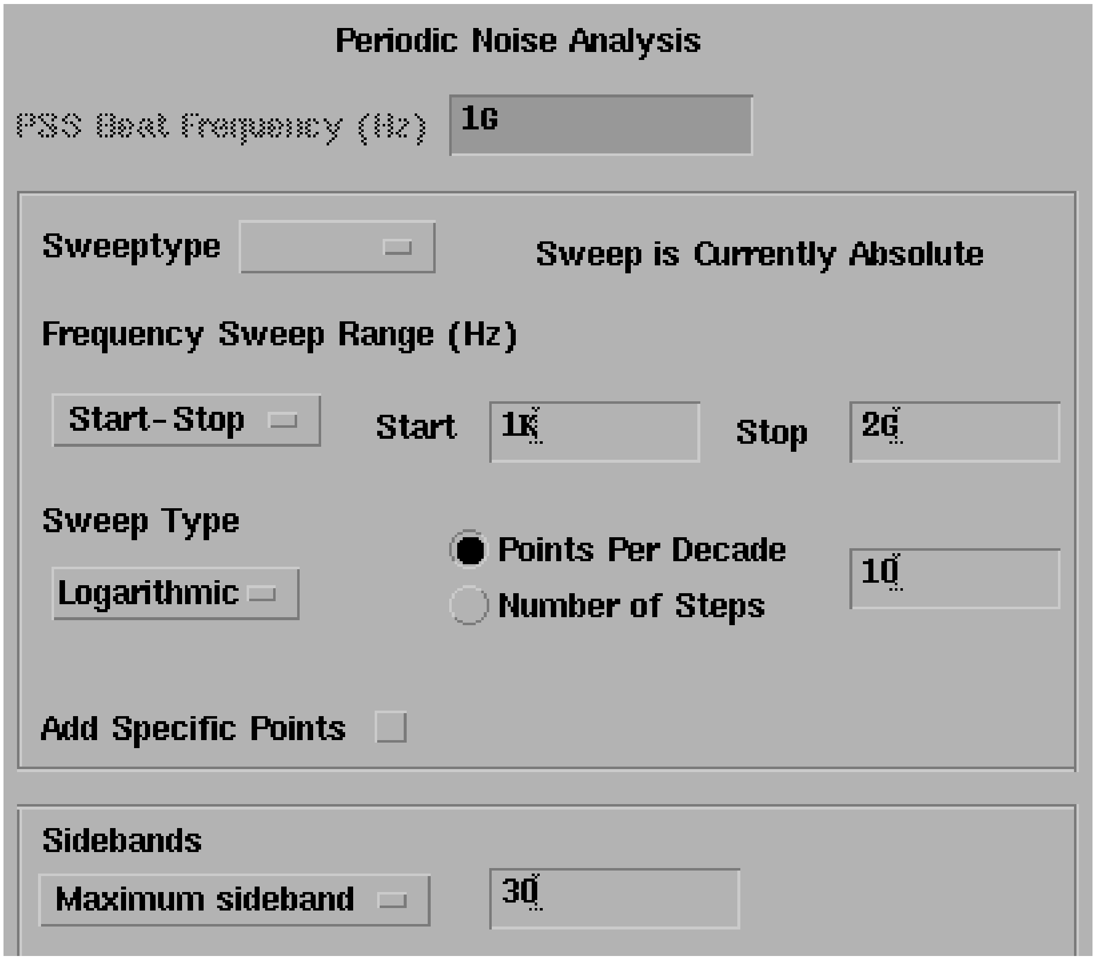
   This frequency range covers the frequencies of interest, but avoids the value `0`. You cannot include the value `0` in a logarithmic sweep.

4. In the *Sweep Type* cyclic field, choose *Logarithmic* for the sweep type, highlight *Points Per Decade* and type `10` in the *Points Per Decade* field.

5. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `30` for the number of sidebands.

   With this setting, you specify that 30 sidebands contribute noise to the output. The value is taken from the first example (*PSS analysis only*) in this chapter.

The top of the Pnoise Choosing Analyses form looks as follows.



6. In the *Output* cyclic field, choose *voltage* for the *Output* value.

7. Highlight the *Positive Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose *Pif*.

`/Pif` appears in the *Positive Output Node* field.

8. Leave the *Negative Output Node* field empty.

This field defaults to `/gnd!` You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.

9. In the *Input Source* cyclic field, choose *voltage*.

10. Click on the *Input Voltage Source Select* button. Then click on the appropriate component in the Schematic window to choose *rf*.

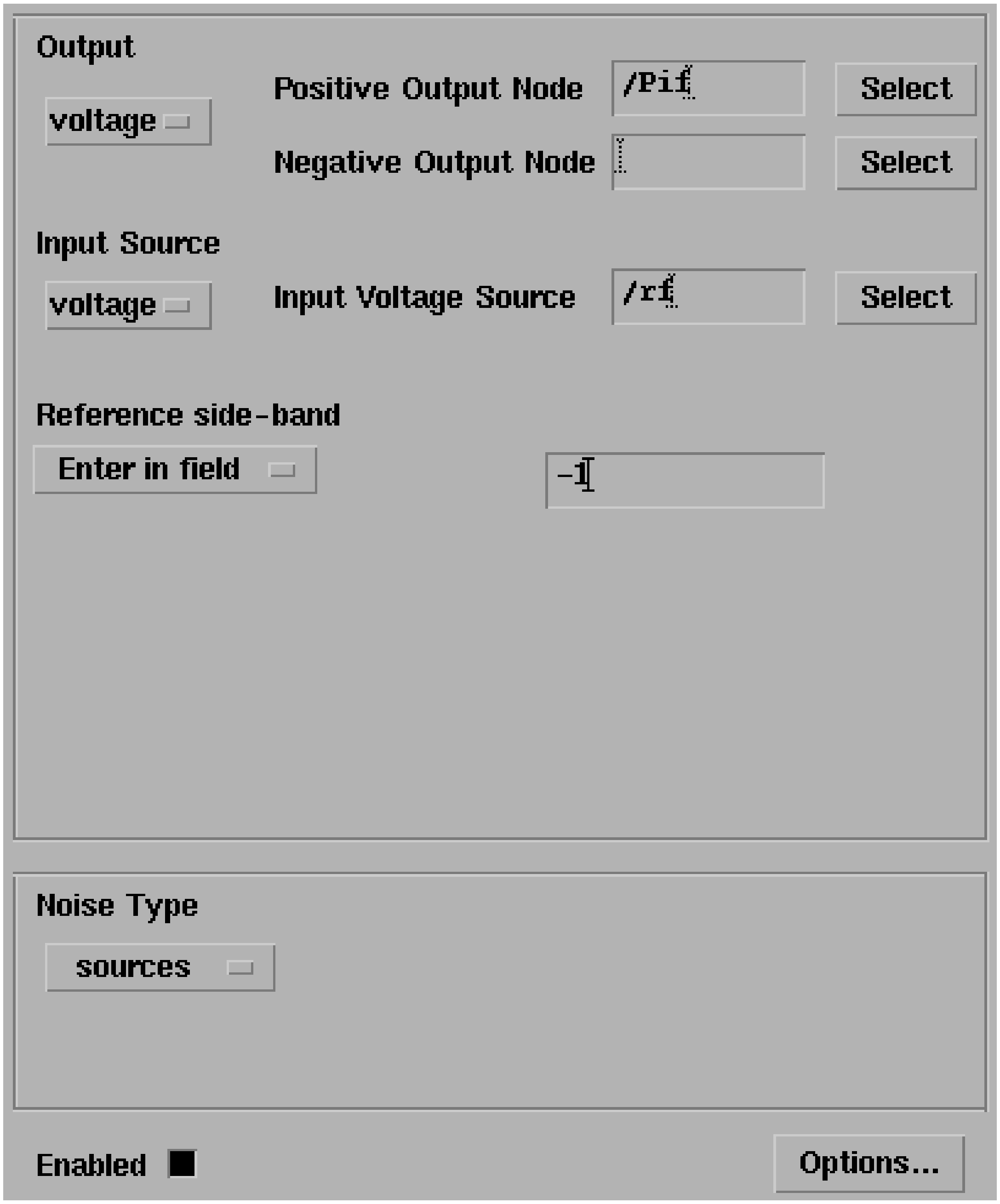
`/rf` appears in the *Input Voltage Source* field.

11. In the *Reference Side-band* cyclic field, choose *Enter in field*.

12. Type `-1` in the *Reference Sideband* field.

In this field, you specify the difference between the input and output frequencies in the whole *frf*. The *Reference Sideband* must be `-1` because this is a down converter. Other possible choices are `0` and `+1`.

The bottom of the Pnoise Choosing Analyses form looks as follows.



13. Highlight *Enabled*.

14. Click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

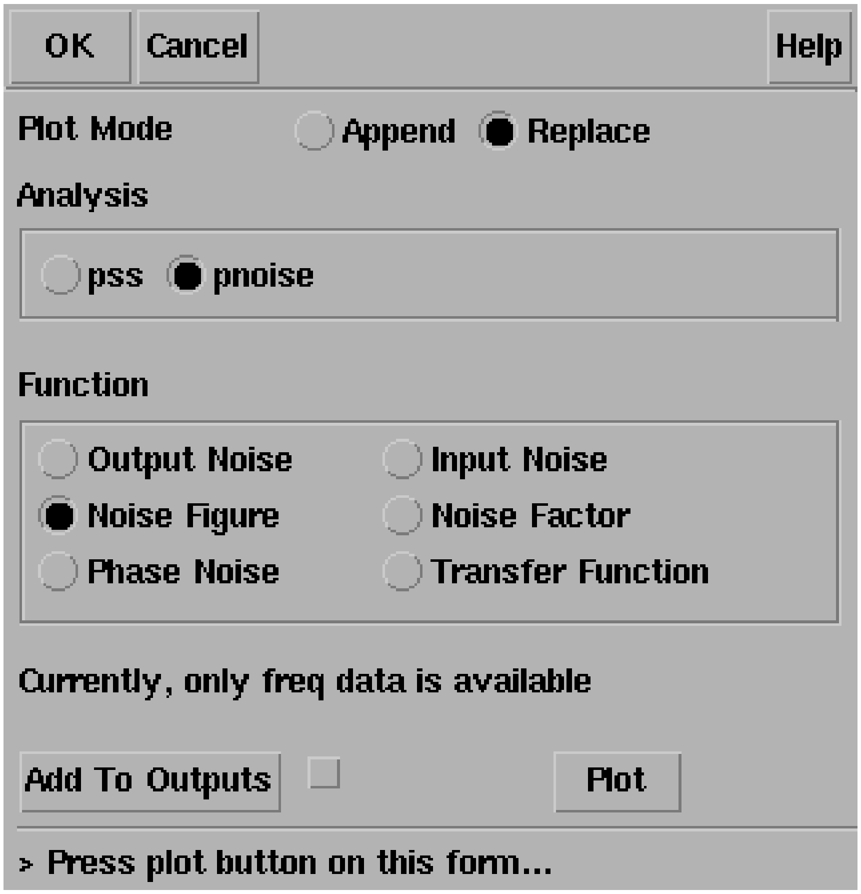
Plotting the Noise Figure

➤ To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise* for *Analysis*.
3. Highlight *Noise Figure* for *Function*.

The completed Direct Plot form appears like the one below.



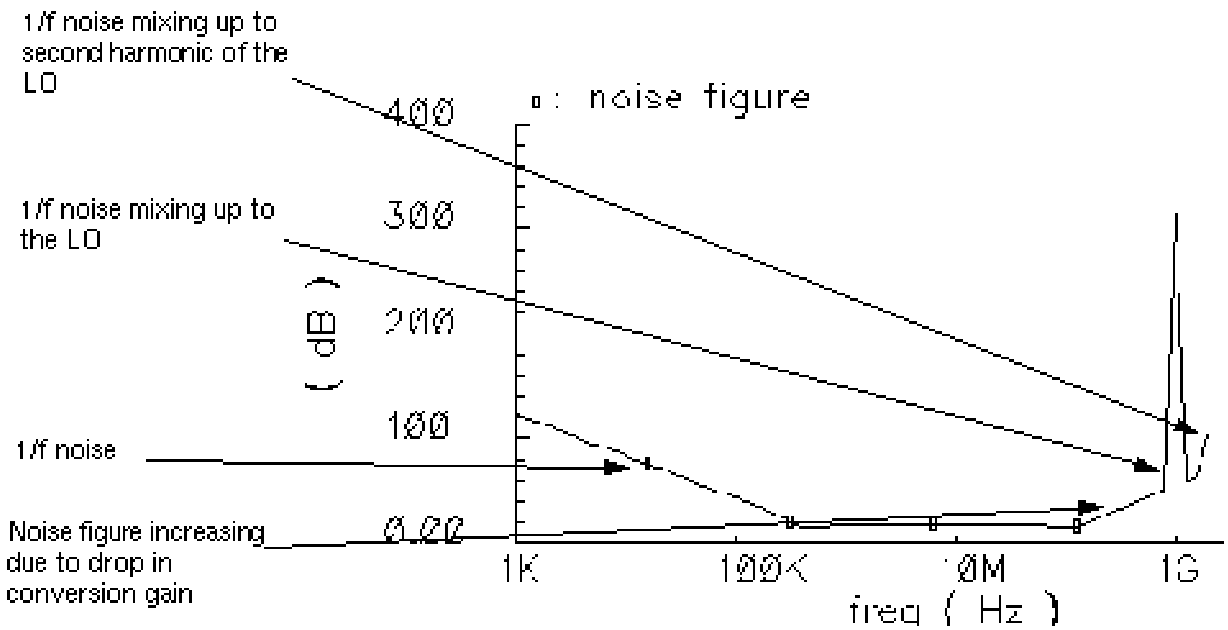
4. Follow the prompt at the bottom of the form

`Press plot button on this form...`

Click on *Plot* in the Direct Plot form.

The plot appears in the Waveform window.

The Waveform window displays the noise figure:



5. To determine the noise figure at different frequencies, move the cursor along the noise figure curve in the Waveform window. In the above plot, the noise figure is `12.22 dB` at `80 MHz`.

Noise Figure Measurement and Periodic S-Parameter Plots with PSS and PSP

The periodic S-Parameter (PSP) analysis computes scattering and noise parameters for n-port circuits that exhibit frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and similar circuits.

In this example you follow a large-signal PSS analysis with a small-signal PSP analysis to analyze noise folding terms induced by the RF input for the ne600p mixer circuit.

This PSP analysis computes

- Periodic S-parameters
- Periodic noise correlation matrices
- Noise figure

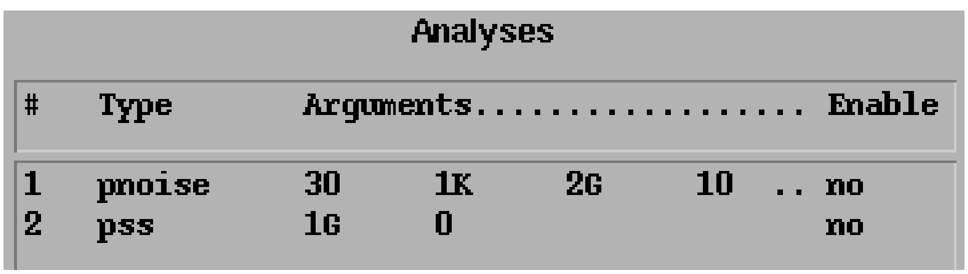
- Equivalent noise parameters

The PSP analysis is a small-signal analysis like the SP analysis, except that, as is true for PAC and PXF analysis, the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing the circuit about a periodically time-varying operating point allows for the computation of S-parameters between circuit ports that convert signals from one frequency band to another.

The PSP analysis can also calculate noise parameters in frequency-converting circuits. PSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. The noise features of the PSP analysis include noise folding effects due to the periodically time-varying nature of the circuit. This is also true for the Pnoise analysis, but not for the SP analysis.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)



Analyses

#	Type	Arguments					Enable
1	pnoise	30	1K	2G	10	..	no
2	pss	1G	0				no

4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

Design Variables

#	Name	Value
2	re	450
3	rc	1K
4	prf	-30
5	vlo	316.2m
6	frf	900M
7	flo	1G

Editing the Schematic

The rf source used in the ne600p mixer circuit has a *Source type* property that you can set to either *sine* or *dc* depending on your application. Suppose that your RF input signal is at 900 MHz, the LO is at 1 GHz, and the IF is at 100 MHz. Before you perform a PSP analysis, you must first perform a PSS analysis.

- In many cases for small-signal analysis, it is sufficient to treat the RF input as a small-signal. For example, most gain measurements are performed under small-signal conditions in which case you set the *Source type* to *dc*.
- When it is important to analyze additional noise folding terms induced by the rf input, you treat the rf source as a large signal in which case you set the *Source type* to *sine*.

See Appendix L, “Using PSP and Pnoise Analyses” for more information.

In this example, you assume the rf source is a large signal so you edit the rf port on the schematic and set the *Source type* parameter to *sine*.

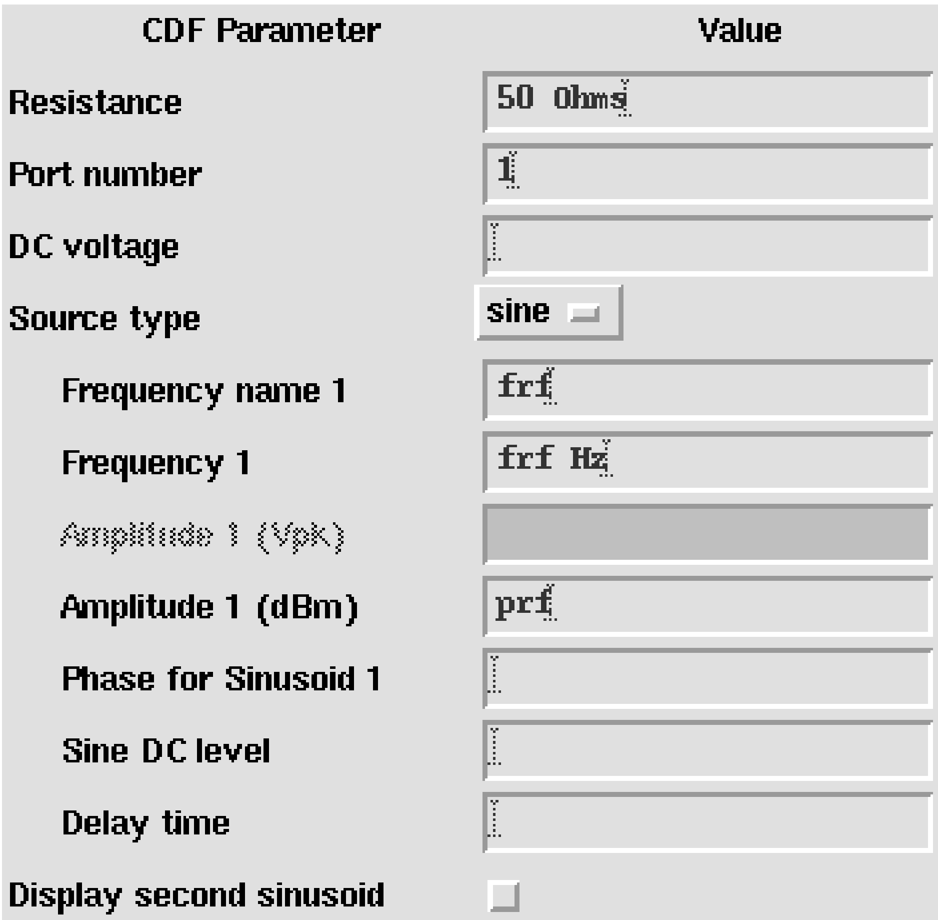
1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic window, click on the *rf* voltage source.

   The Edit Object Properties form changes to display information for the voltage source.

3. In the Edit Object Properties form, choose *sine* as the *Source type*, if necessary, and click *OK*.



4. In the Schematic window, choose *Design – Check and Save.*

Setting up the PSS and PSP Analyses

Use the Choosing Analyses form, to set up two analyses, first PSS then PSP.

➤ Choose *Analyses – Choose* in the Simulation window.

The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. At the top of the Choosing Analyses form, highlight *pss.*

2. In the *Fundamental Tones* list box, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed. (The *Beat Frequency* button is highlighted by default.) The 100 M value for beat frequency is the minimum period for which both RF at 900 MHz and LO at 1 GHz are periodic (or for which RF and LO are integer multiples of the fundamental frequency).

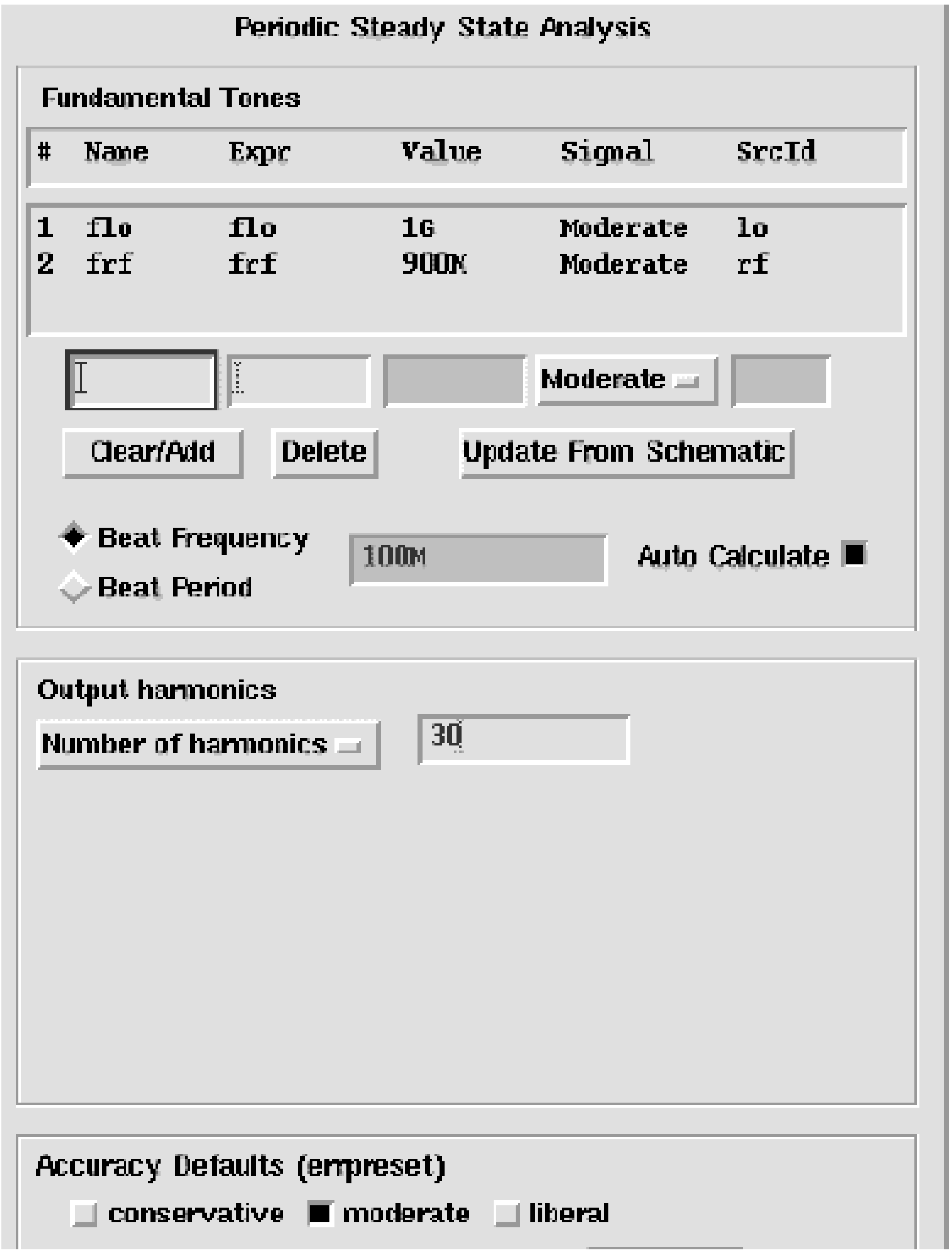
3. In the *Output harmonics* cyclic field, choose *Number of harmonics.* Type `30` in the *Number of harmonics* field.

   This entry expands the plotted frequency range to include the areas of interest around both 100 MHz and 1 GHz. (30 $\times$ 100 MHz equals 3GHz, or 0 to 3GHz, and includes all important frequencies.)

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

5. Be sure the *Enabled* button is highlighted.

The top of the correctly filled out form appears below.



6. Click *Apply* to check the information you entered in the PSS Choosing Analyses form for pss.

Correct any errors.

Setting Up the PSP Analysis

1. In the *Analysis* section at the top of the Choosing Analyses form, highlight *psp.*

   The Choosing Analyses form changes to let you specify data for the *PSP* analysis.

2. In the *Sweeptype* cyclic field, choose *relative*.

   The *Sweeptype* parameter controls the way that frequencies are swept in the PSP analysis. When you specify a *relative* sweep, the sweep is relative to the analysis harmonics (not the PSS fundamental frequency).

   For PSP analysis, the frequencies of the input and response are usually different. This is an important difference between the PSP and SP analyses. Because the PSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics, the `freqaxis` and `sweeptype` parameters behave somewhat differently in PSP analysis than they behave in PAC and PXF analyses.

3. In the *Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop* and type `-20M` in the *Start* field and `30M` in the *Stop* field.

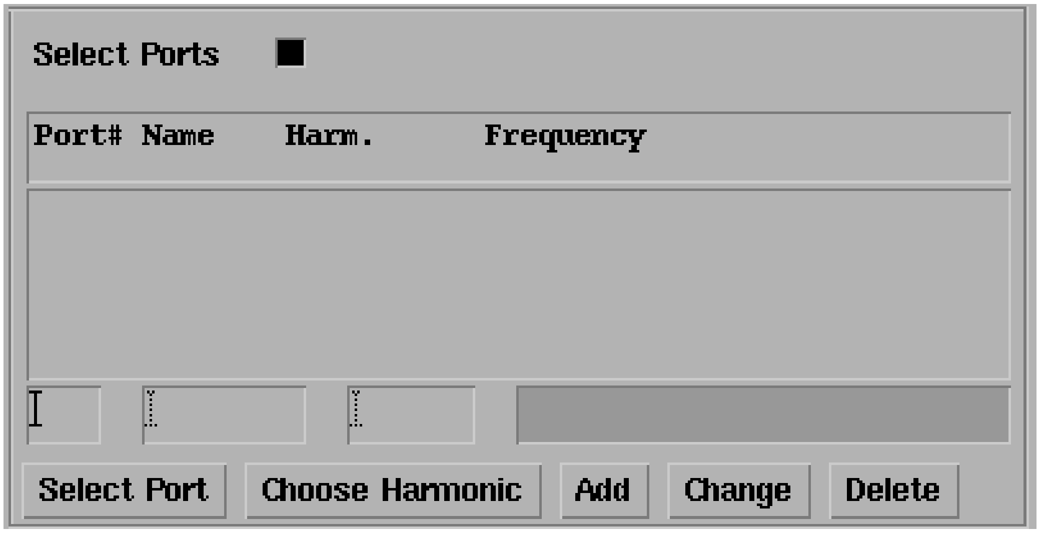
   The PSP analysis is performed from 20 MHz below the RF center frequency to 30 MHz above it. This frequency range accounts for inputs on the rf port in the range of -920 MHz to -870 MHz. Noise parameters (such as noise figure) are computed in a 50 MHz band around the frequency specified by the output harmonic.

4. Leave the *Sweep Type* cyclic field set to *Automatic*.

5. Highlight *Select Ports* to select the input and output ports on the schematic.

   - The first port, the input port, is the *rf* port.

❑ The second port, the output port, is the *rif* port.



6. Select the input port.

a. Type `1` in the first field, the *Port#* field. It is directly above the *Select Port* button.

b. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

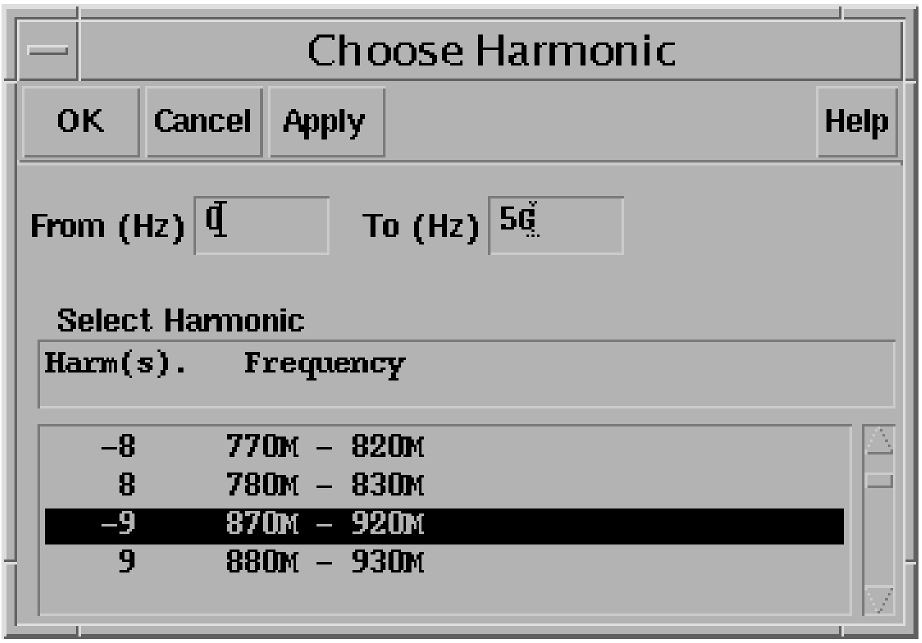
`Select source...`

c. In the Schematic window, click on the appropriate port to choose `/rf`.

`/rf` appears in the *Name* field.

d. Click *Choose Harmonic.*

e. The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rf* port.



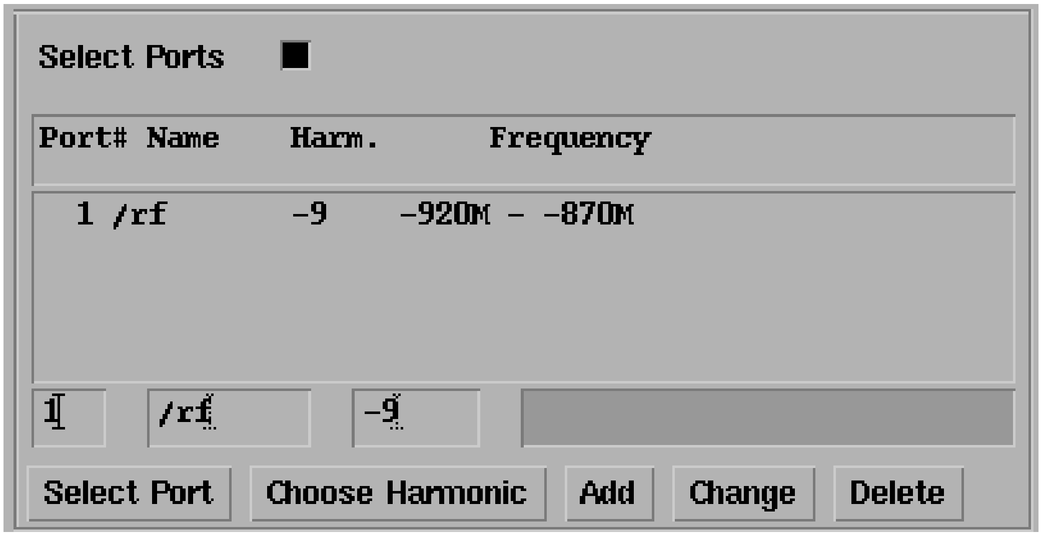
f. In the Choose Harmonic form, scroll through the list and highlight the harmonic with index `-9`.

g. Click *OK*.

The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, `-9` displays in the *Harm.* field .

h. In the *Select Ports* area, click *Add*.

Information for the input port displays in the *Select Ports* list box.



7. Select the output port.

a. Type 2 in the first field, the *Port#* field. It is directly above the *Select Port* button.

b. Click *Select Port* and follow the prompt at the bottom of the Schematic window.

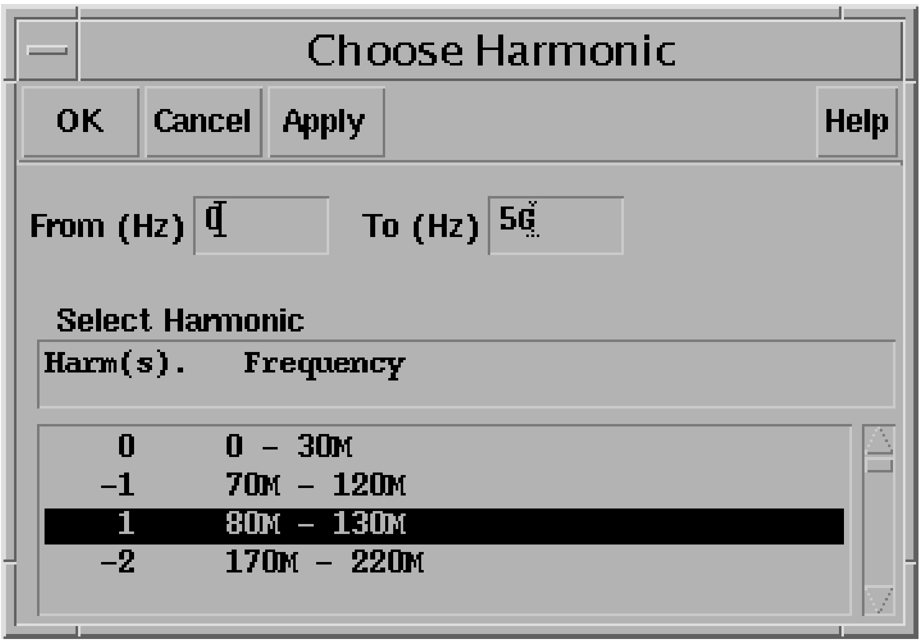
`Select source...`

c. In the Schematic window, click on the appropriate port to choose `/rif`.

`/rif` appears in the *Name* field.

d. Click *Choose Harmonic.*

e. The Choose Harmonic form displays with a list of harmonics (by index and frequency) for the *rif* port.



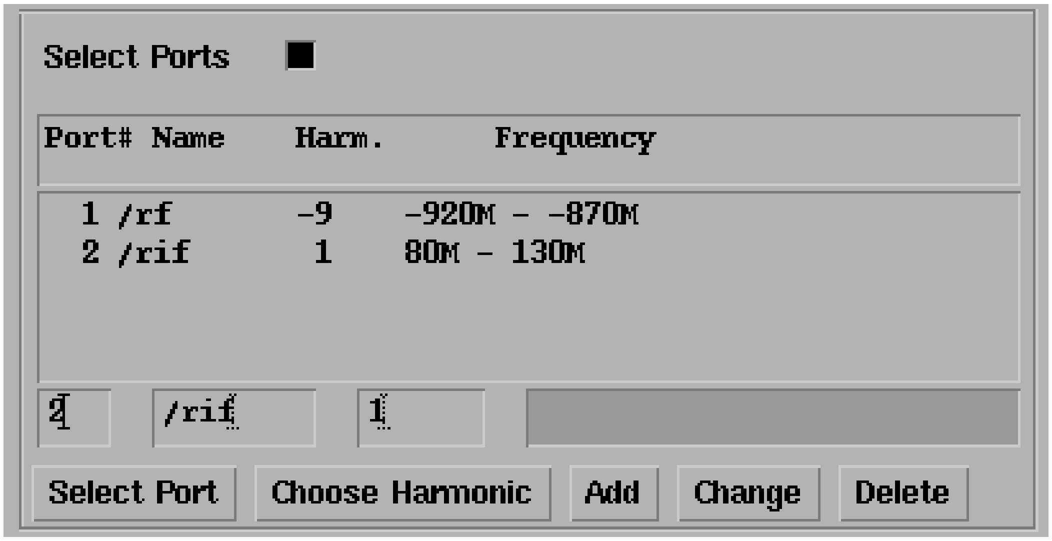
f. In the Choose Harmonic form, scroll through the list and highlight the harmonic with index 1.

g. In the Choose Harmonic form, click *OK*.

h. The Choose Harmonic form closes. In the *Select Ports* area of the Choosing Analyses form, 1 displays in the *Harm*. field .

i. In the *Select Ports* area, click *Add*.

Information for the output port is added to the *Select Ports* list box.



With the 1 GHz LO, small-signal inputs around 900 MHz (harmonics +/-9 of the PSS fundamental) appear at around 100 MHz (or harmonics -/+1 of the fundamental). You select harmonic 1 as the output harmonic, and harmonic -9 as the input harmonic because a single complex-exponential input

$$e^{i\omega_s t}$$

on the lower side of 900 MHz (harmonic -9) appears as the upper sideband of harmonic 1 at around 100 MHz.

Harmonics -9 and 1 are separated by 10 fundamental periods, which corresponds to the LO frequency of 1 GHz.

8. Below *Do Noise* near the bottom of the form, highlight *Yes* to calculate noise parameters as part of the PSP analysis.

9. Type `50` in the *Maximum Sideband* field.

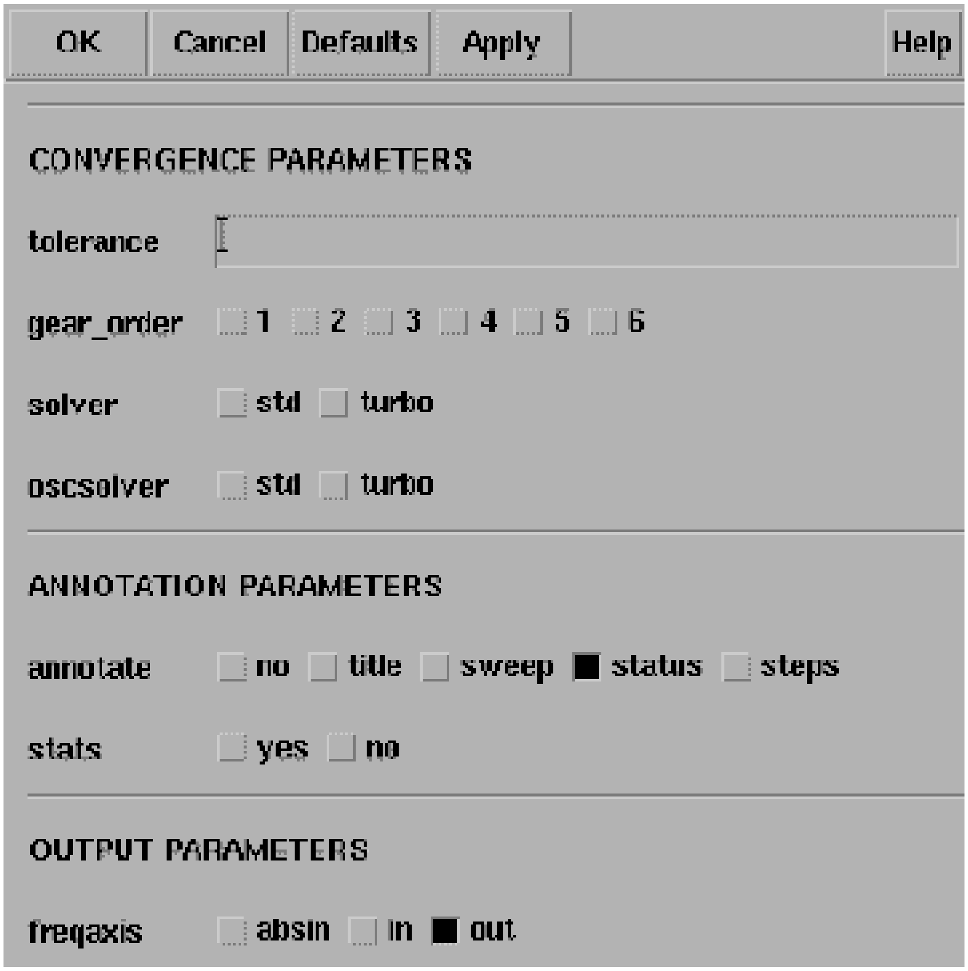
Setting *Maximum Sideband* to `50` accounts for noise folding from up to 5 GHz in frequency.

All noise computations in PSP analysis involve noise folding effects. The *maxsideband* parameter specifies the maximum sideband to include for summing noise contributions either up-converted or down-converted to the output at the frequency of interest.

10. Highlight *Enabled*.

11. Click the *Options* button to display the PSP Options form.

12. Highlight *out* for the *freqaxis* parameter.

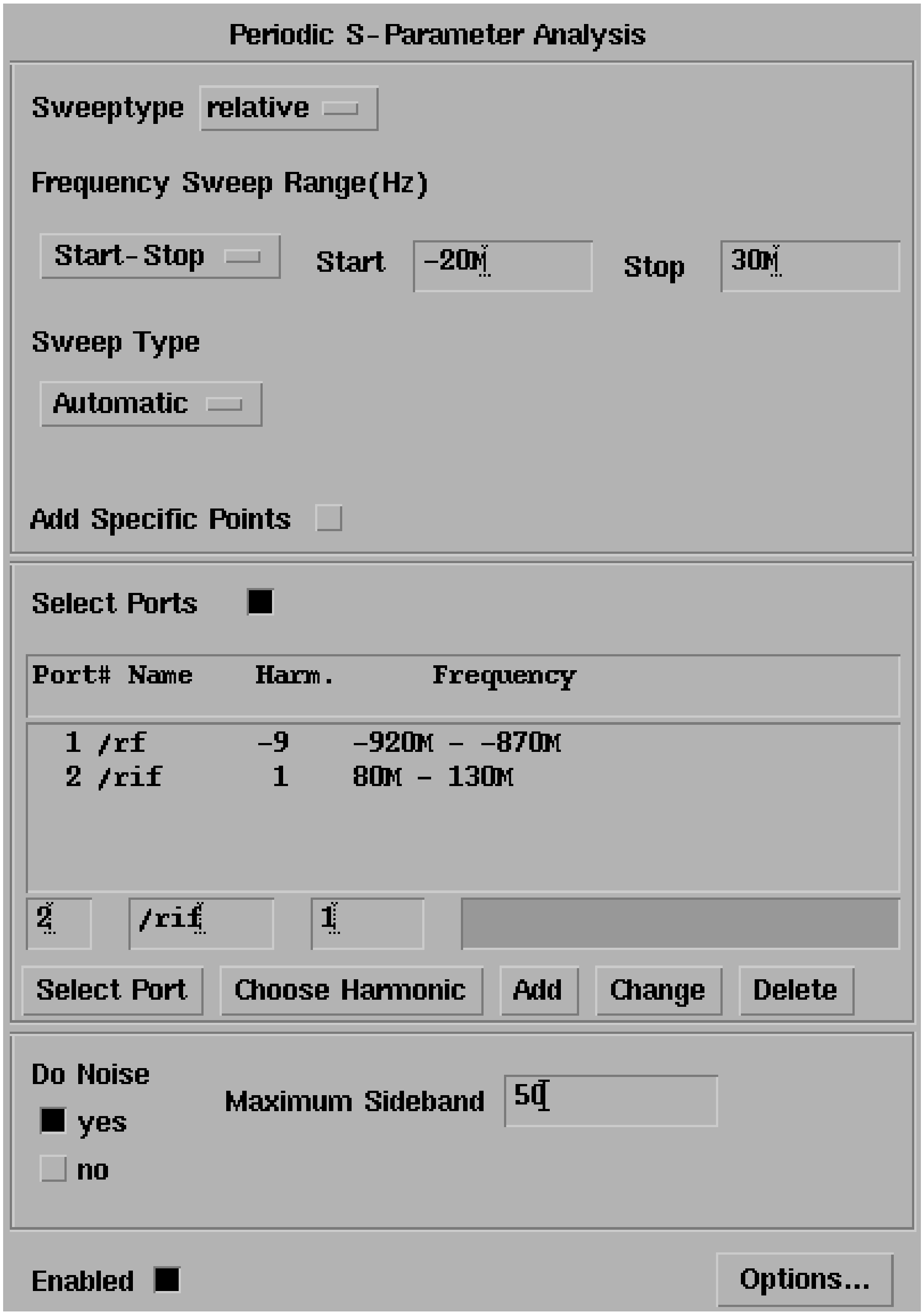


Selecting *out* changes the scale used for the output axis so that it runs from 80 MHz to 130 MHz.

The *freqaxis* parameter specifies whether the results should be output versus the absolute value of the input frequency (*absout*), the input frequency (*in*), or the output frequency (*out*).

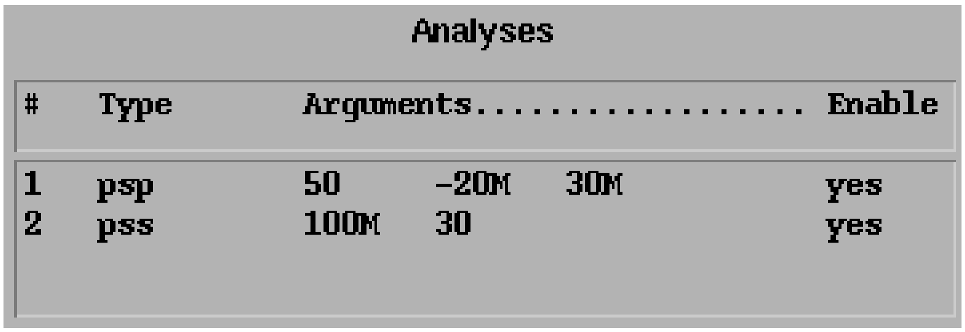
13. Click *OK* in the PSP Options form.

The completed Periodic S-Parameters Analysis section of the Choosing Analyses form appears like the one below.



14. In the Choosing Analyses form, click *OK*.

Information about the pss and psp analyses appears in the Analyses section of the Simulation window.


Analyses

#	Type	Arguments..................			Enable
1	psp	50	-20M	30M	yes
2	pss	100M	30		yes

Running the Simulation

You can now run the simulation. In addition to computing the periodic S-parameters, the periodic noise correlation matrixes are also computed, as are noise figure and the equivalent noise parameters.

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Noise Figure

➤ To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.

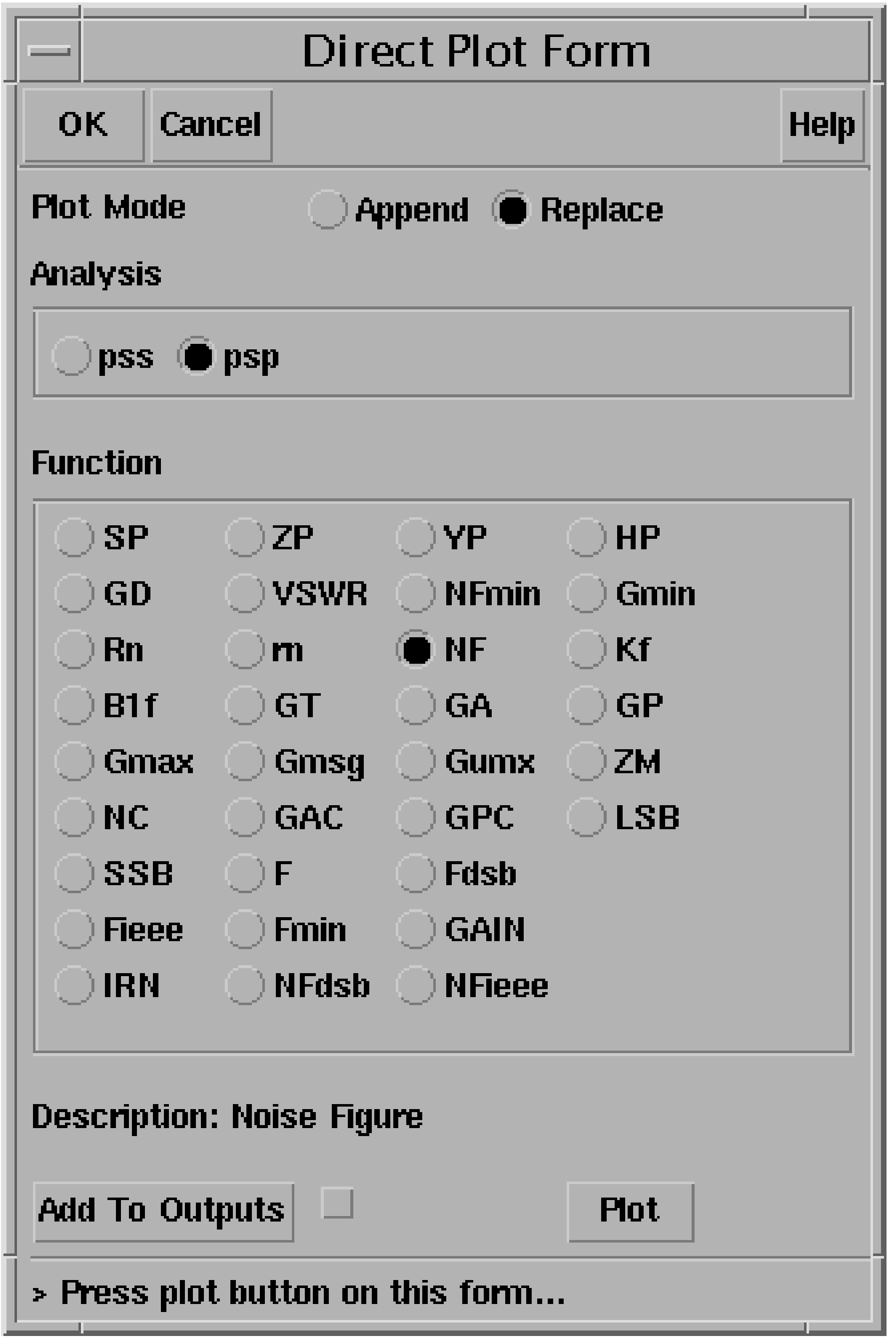
2. Highlight *psp* for *Analysis*.

The form changes to display choices appropriate for PSP analysis results.

3. Highlight *NF* for *Function* to display the Noise Figure.

The form changes to display choices appropriate for noise figure.

The completed Direct Plot form appears like the one below.



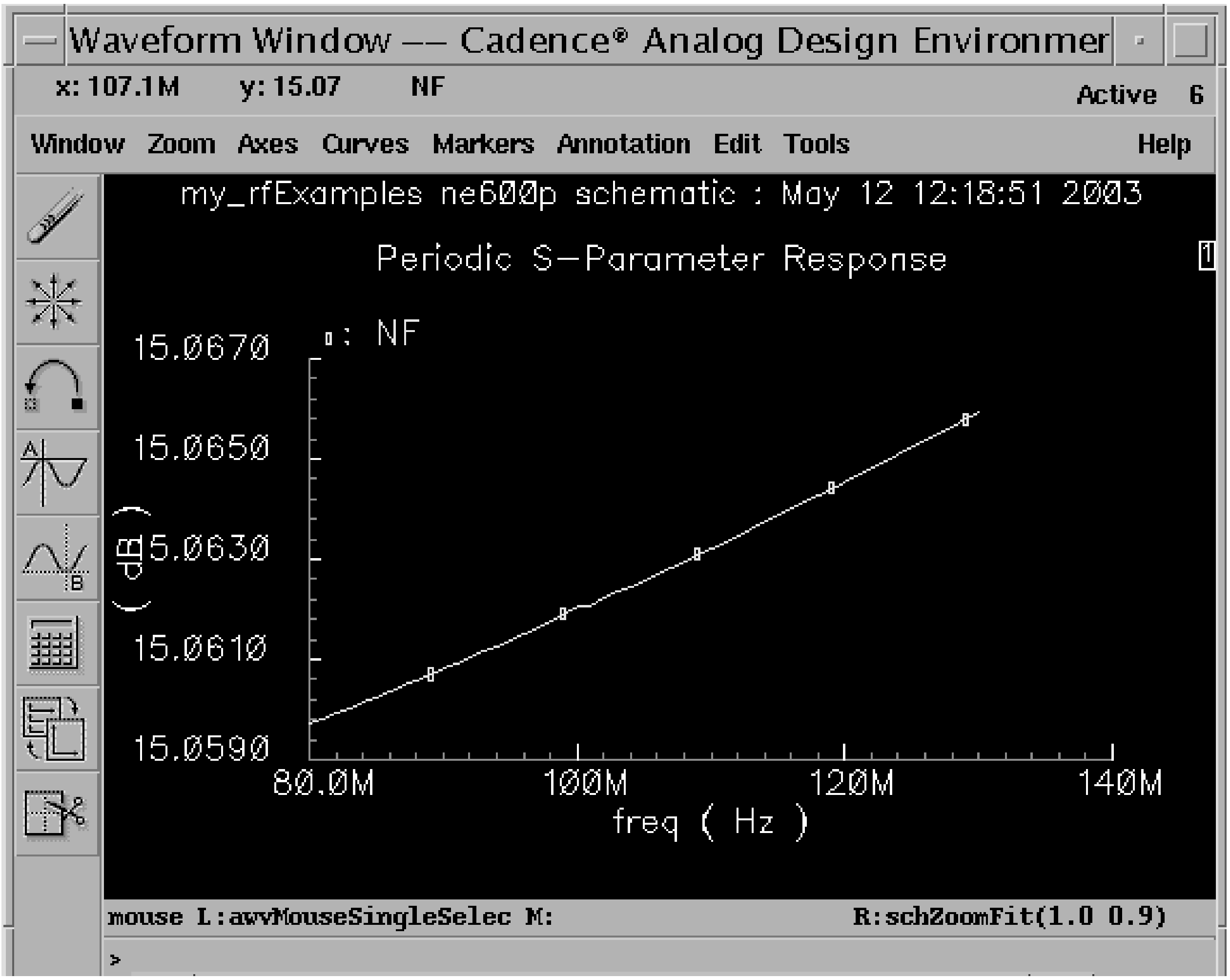
4. Follow the prompt at the bottom of the form

`Press plot button on this form...`

Click *Plot* to display the Noise Figure.

The plot appears in the Waveform window.

The Waveform window display appears like the one below:



5. To determine the noise figure at different frequencies, move the cursor along the noise figure curve and observe the x and y values displayed at the top of the Waveform window.

Plotting Periodic S-Parameters

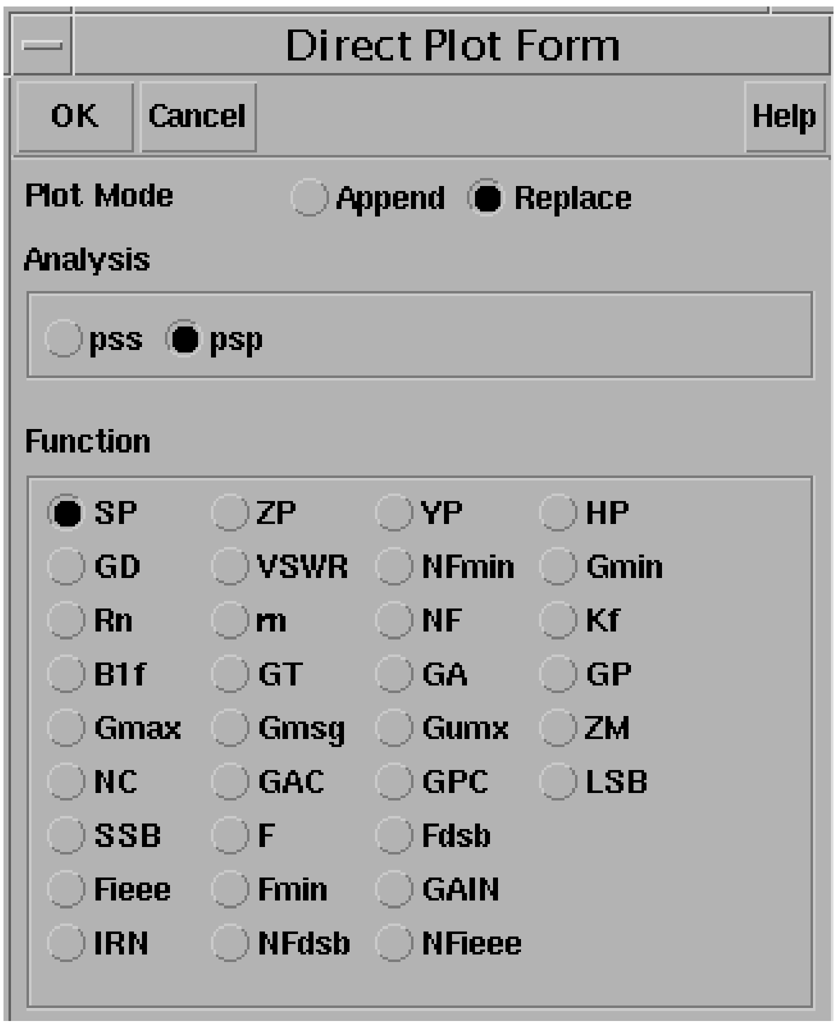
➤ If necessary, open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.

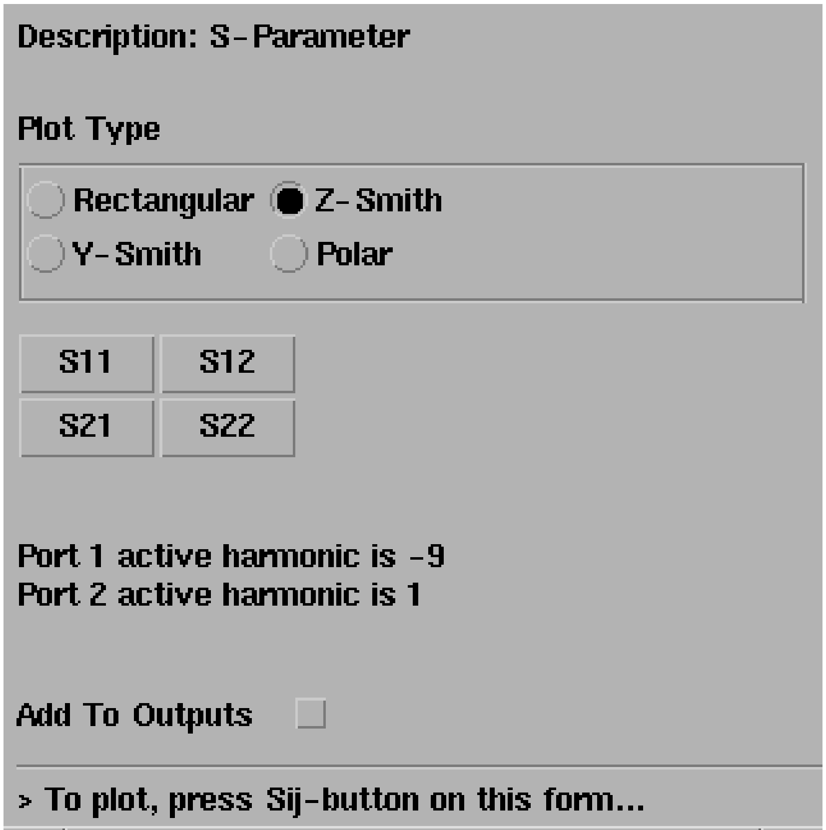
2. Highlight *psp* for *Analysis*.
3. Highlight *SP* for *Function* to display periodic S-parameters.

   The top of the completed Direct Plot form appears like the one below.



4. Highlight *Z-Smith* for *Plot Type* to display periodic S-parameters.

The bottom of the completed Direct Plot form appears like the one below.



5. Follow the prompt at the bottom of the form

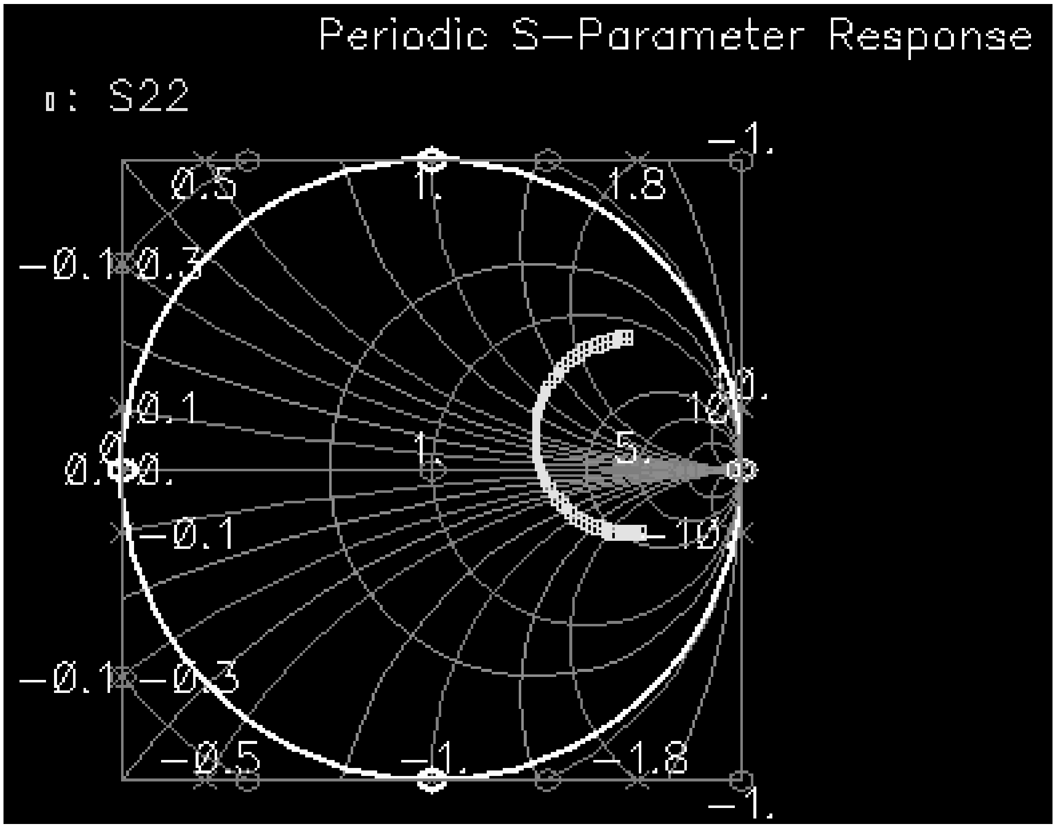
`To plot, press Sij-button on this form...`

Click *S22* to plot output match.

Recall from basic S-parameter theory that for a 2-port circuit S22 represents the match at port 2. In this example S22 represents the match at the if port.

The plot appears in the Waveform window.

The Waveform window display appears like the one below:



At the center of the Smith chart (at the1), the match is perfect. The match is better as the curve gets closer to the center of the Smith chart. Notice that frequency changes with match. The displayed frequency and position on the Smith chart change as you move the cursor along the curve.

The frequency where the output match is the best is approximately `105.4`MHz, and the reflection coefficient at that frequency is `341.2`m at a phase of `9.017` degrees

Conversion Gain Measurement with PSS and PXF

You can combine a PSS analysis with a periodic small-signal transfer function PXF analysis to determine the conversion gain of the down converter. With changes to the plotting form, you can also calculate the power supply rejection and local oscillator feedthrough.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

Editing the Schematic

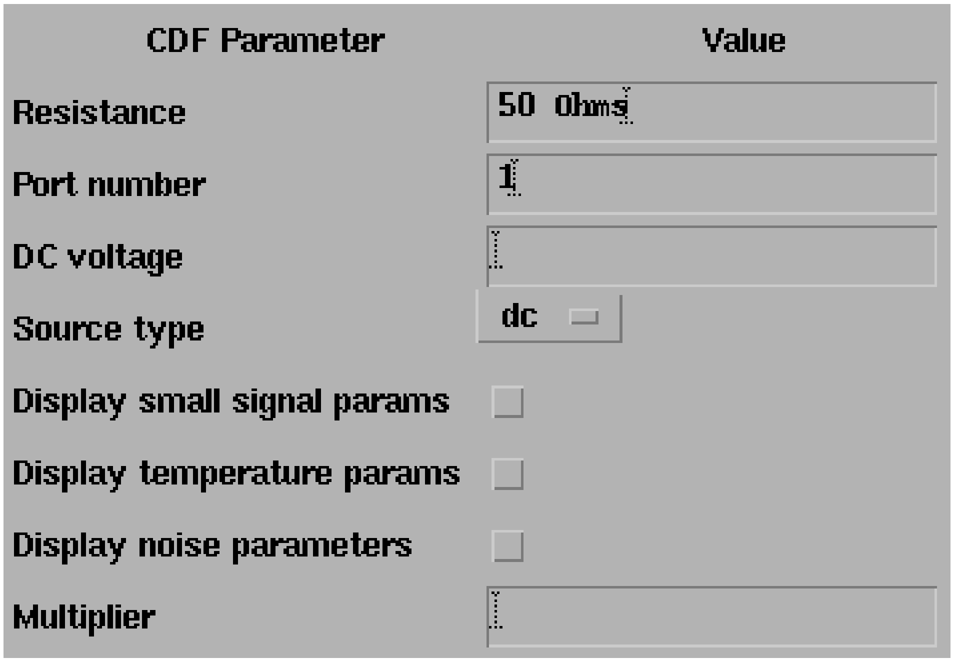
You must modify the schematic to set the RF source to a DC source to be sure the PSS analysis is the response of the `ne600p` mixer to only the LO signal.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

2. In the Schematic window, click on the *rf* voltage source.

3. In the Edit Object Properties form, choose *dc* as the *Source type*, if necessary, and click *OK*.



CDF Parameter	Value
Resistance	50 Ohms
Port number	1
DC voltage	
Source type	dc
Display small signal params	
Display temperature params	
Display noise parameters	
Multiplier	

4. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS and PXF Analyses

1. In the Simulation window, choose *Analyses – Choose*.

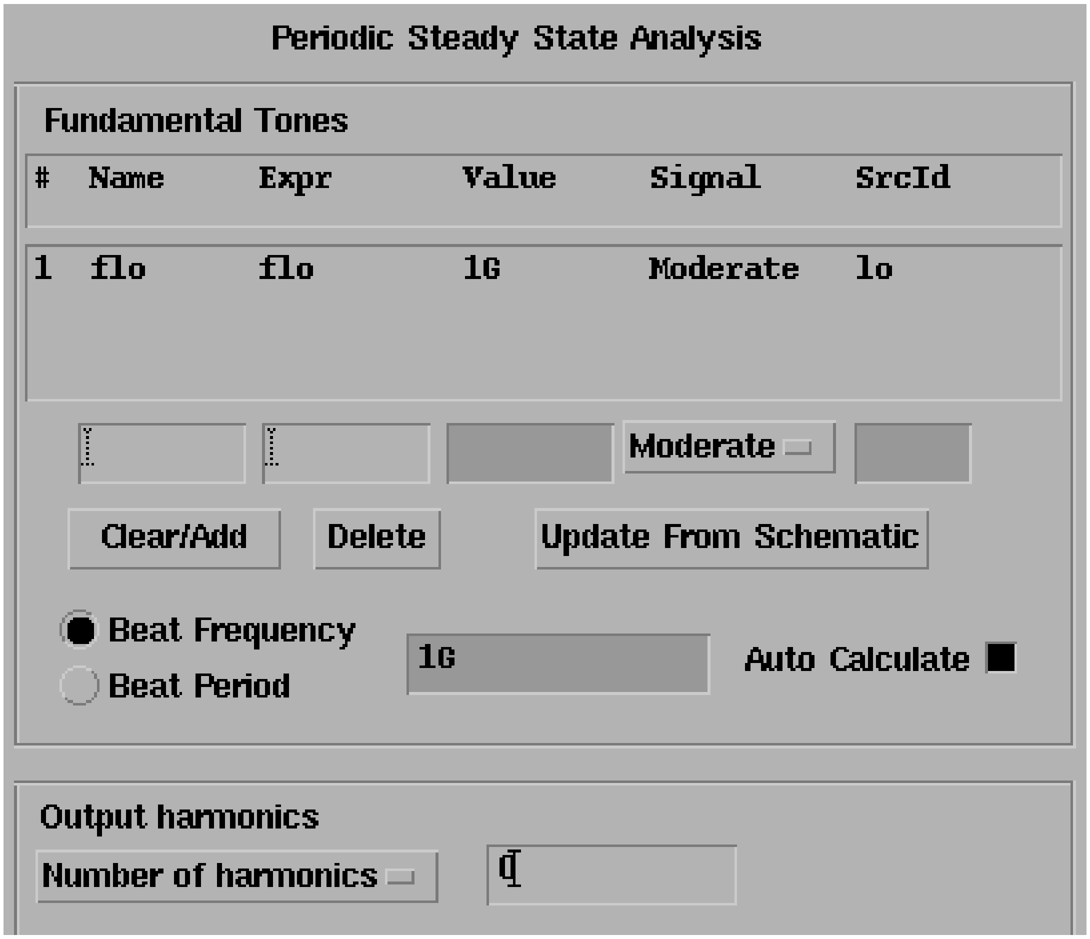
   The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, highlight *pss* for the *Analysis*.
2. In the *Fundamental Tones* section, be sure the *Auto Calculate* button is highlighted.

   The *Beat Frequency* is now displayed as `1G`. LO is the only time-varying signal in the circuit, so it becomes the fundamental frequency. The *Beat Frequency* button is highlighted by default.
3. In the *Output Harmonics* cyclic field, choose *Number of Harmonics* and type `0` in the *Number of Harmonics* field.

The top of the PSS analysis form looks like this.

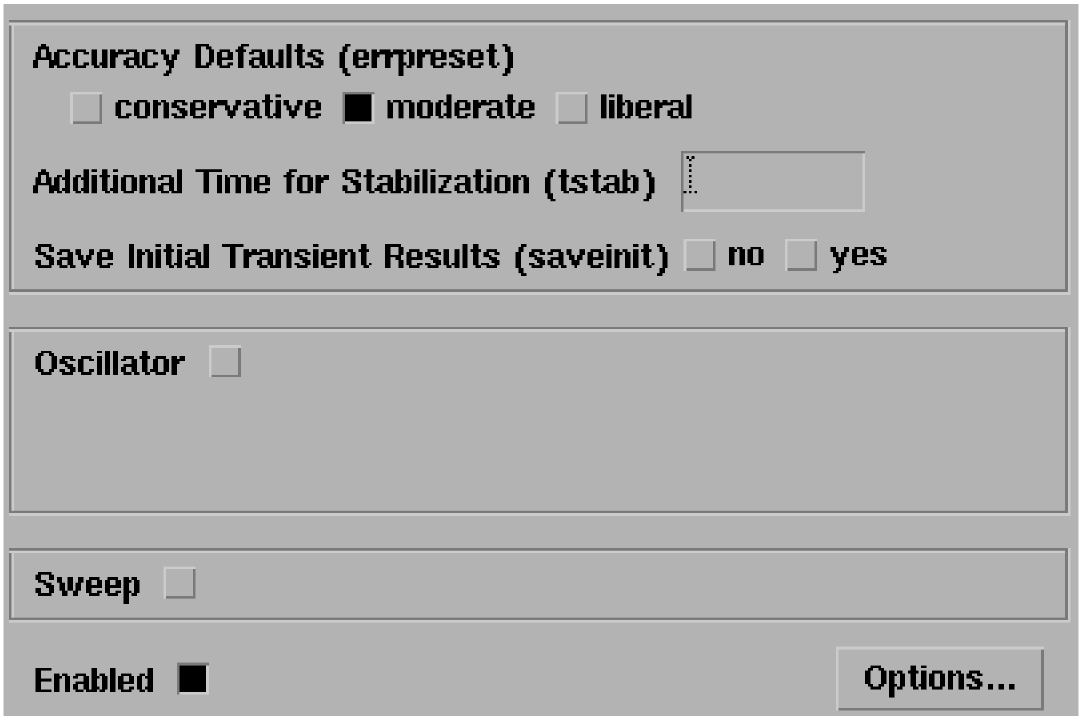


You must turn off harmonic generation because it is required to set up the subsequent PXF analysis.

4. Highlight *conservative* for the *Accuracy Defaults (errpreset)* setting.

5. Highlight *Enabled*.

The bottom of the PSS analysis form looks like this.



Setting Up the PXF Analysis

1. At the top of the Choosing Analyses form, click on *pxf*.

   The Choosing Analyses form changes to let you specify data for the PXF analysis.

2. In the *Frequency Sweep Range (Hz)* cyclic field, choose *Start – Stop*, and type `1M` in the *Start* field and `300M` in the *Stop* field.

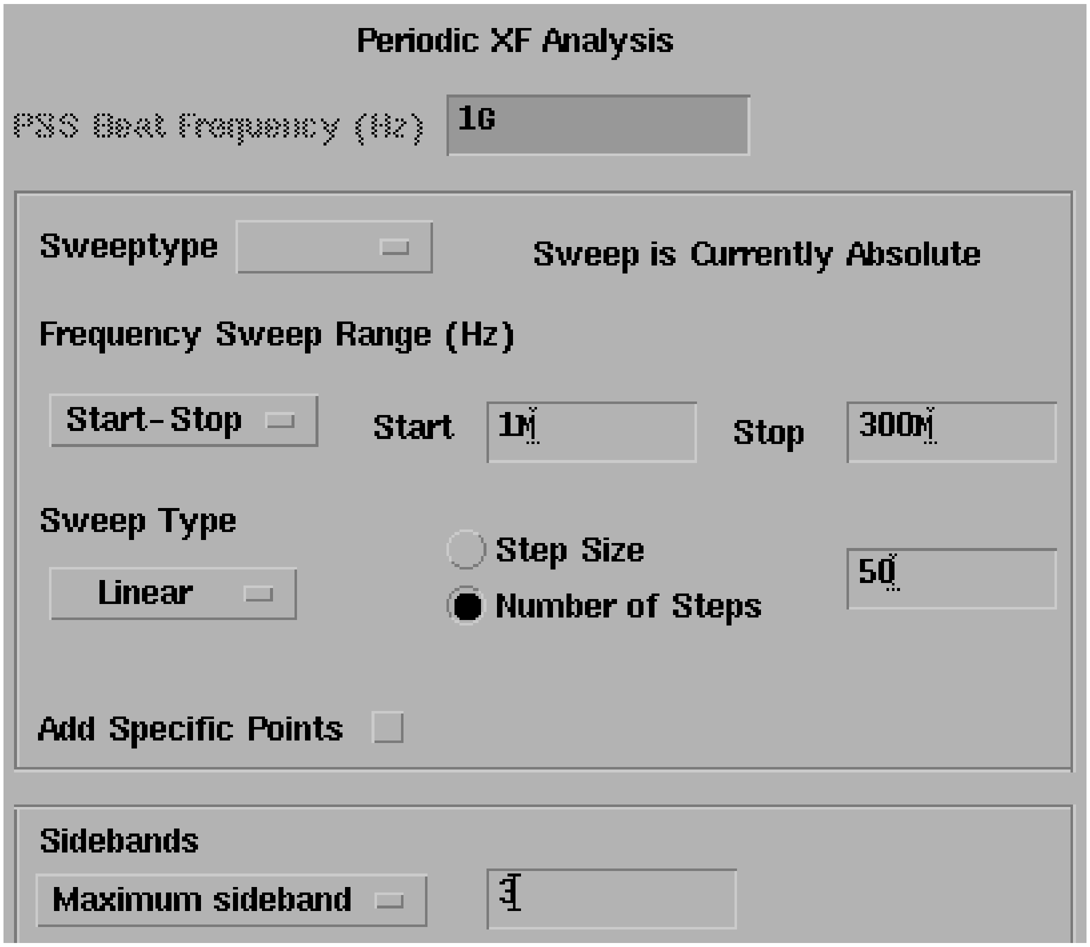
   This range specification leaves space between the plots, but still shows the dramatic gain changes in the first 100 Mhz of each plot.

3. In the *Sweep Type* cyclic field, choose *Linear* and highlight *Number of Steps*. Type `50` for the *Number of Steps*.

   Larger numbers of total points increase the resolution of the plot but also require a longer simulation time.

4. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `3` as the value.

The top of the PXF Choosing Analyses form looks like this.

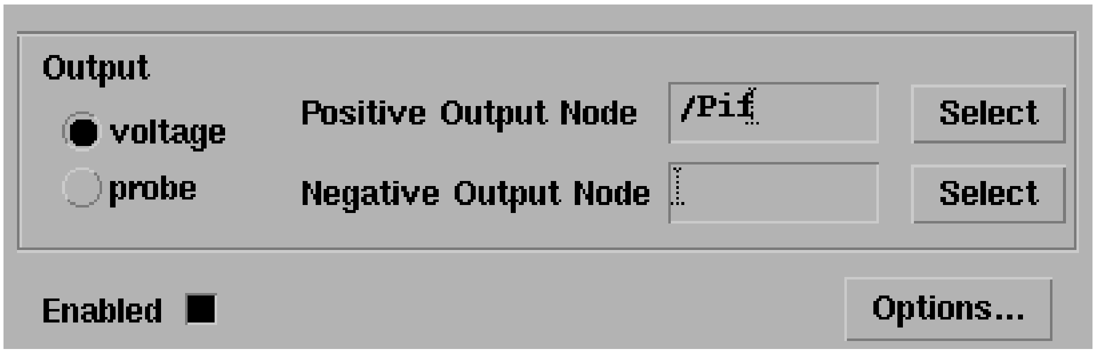


5. In the *Output* section, highlight *voltage*.
6. Click on the *Positive Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose *Pif*.

   `/Pif` appears in the *Positive Output Node* field.
7. Leave the *Negative Output Node* field empty.

   This field defaults to `/gnd!` You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.
8. Highlight *Enabled*.

The bottom of the PXF Choosing Analyses form looks like this.



9. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

The output log file appears and displays information about the simulation while it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Conversion Gain

1. Choose *Results – Direct Plot – Main Form* in the Simulation window.

The Direct Plot form appears.

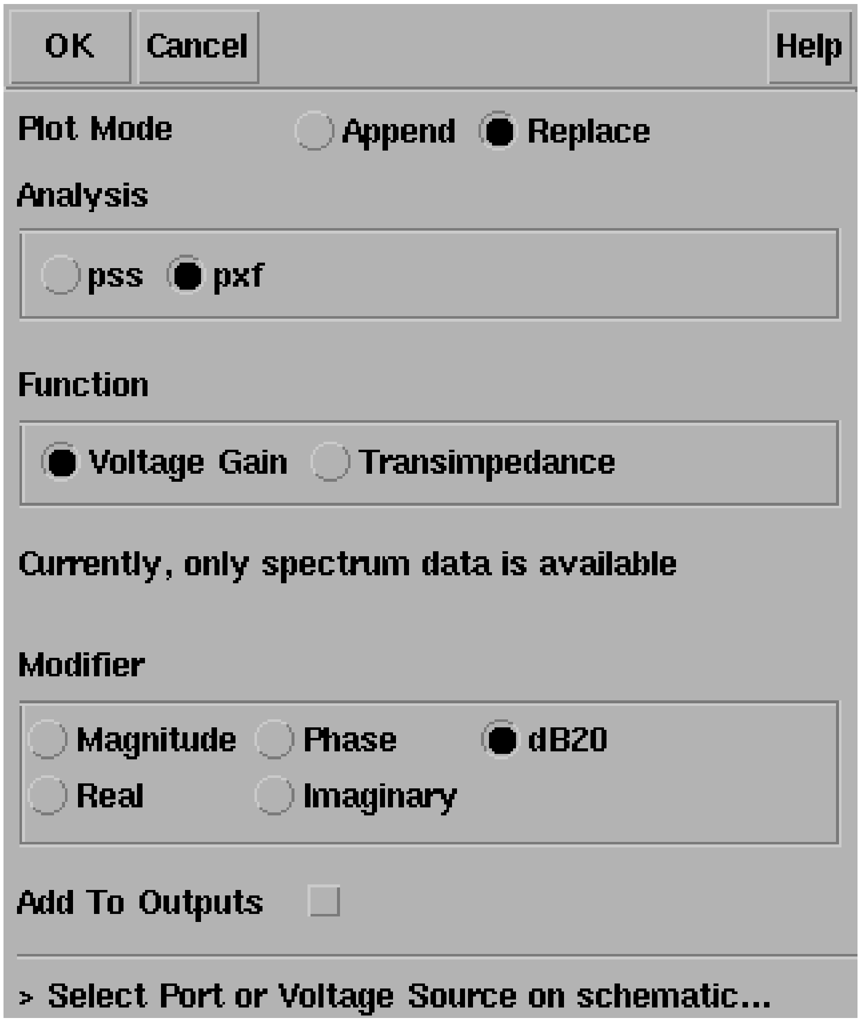
2. Highlight *Replace* for *Plot Mode*.

3. Highlight *pxf* for *Analysis*.

4. Highlight *Voltage Gain* for *Function*.

5. Highlight *dB20* for *Modifier*.

The completed PXF Direct Plot form looks like this.

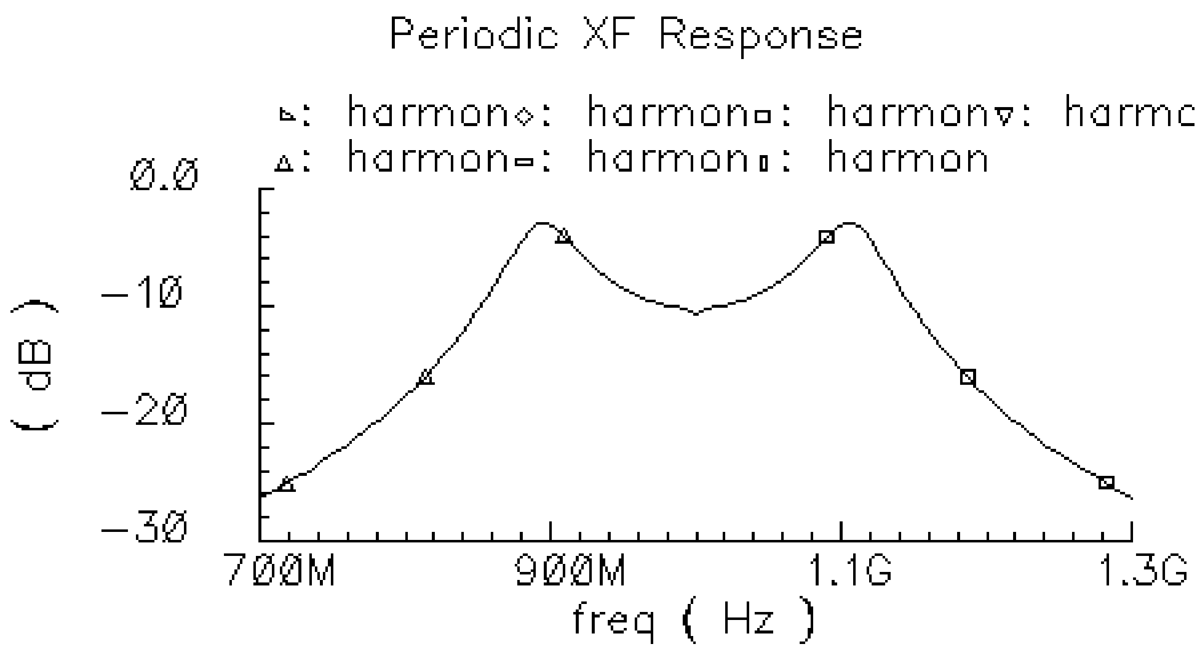


6. Follow the prompt at the bottom of the form

`Select Port or Voltage Source on schematic...`

Click on the *Prf* source on the schematic.

The plot appears in the Waveform window.



7. To determine the gain at different frequencies, move the cursor along the curve and use the readout at the top of the Waveform window.

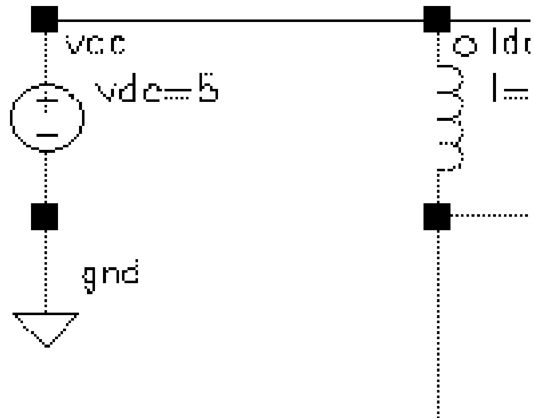
   For example, the gain is `-5.46 dB` at `920 MHz`.

Plotting the Power Supply Rejection

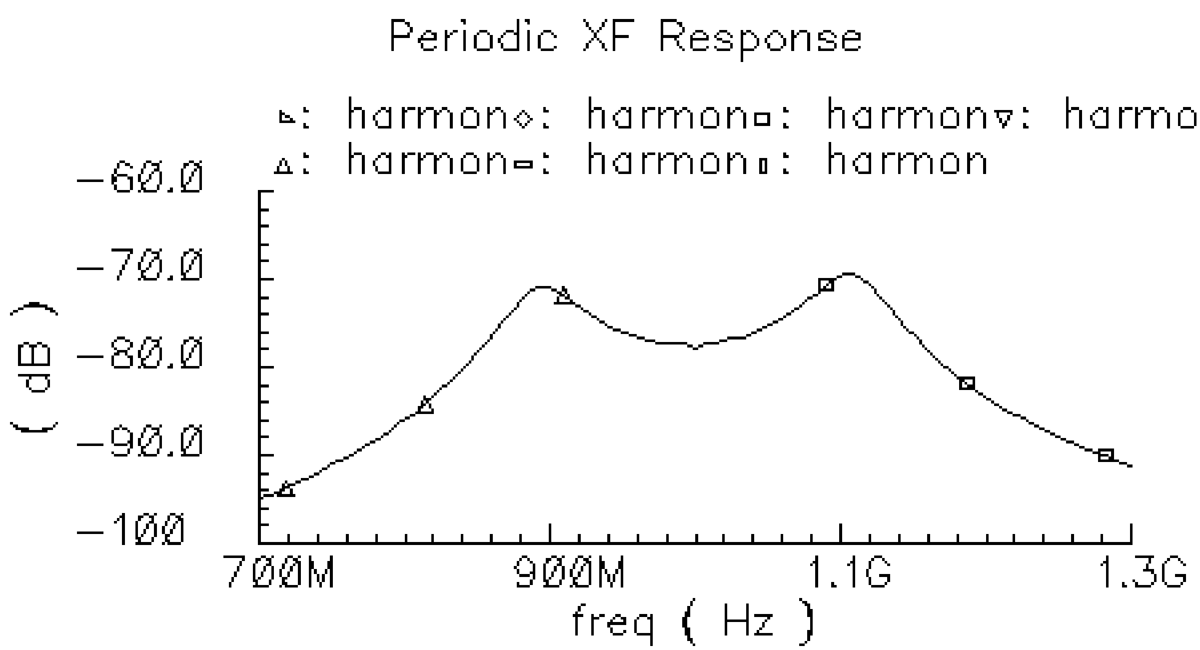
In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pxf* for *Analysis*.

3. In the Schematic window, click on the *vcc* power supply at the top left of the schematic.



The Waveform window appears as follows.



4. To determine the rejection at different frequencies, move the cursor along the curve and use the readout at the top of the Waveform window.

Calculating the 1 dB Compression Point with Swept PSS

In this example, a swept PSS analysis determines the `1dB` compression point for the `ne600p` mixer configured as a down converter.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, click on the *rf* voltage source.
2. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.
3. Choose *sine* for *Source type*.
4. Type `prf` in the *Amplitude 1 (dBm)* field.
5. Click *OK*.
6. In the Schematic window, choose *Design – Check and Save*.

Setting Up the Swept PSS Analysis

1. In the Simulation window, choose *Analyses – Choose*.

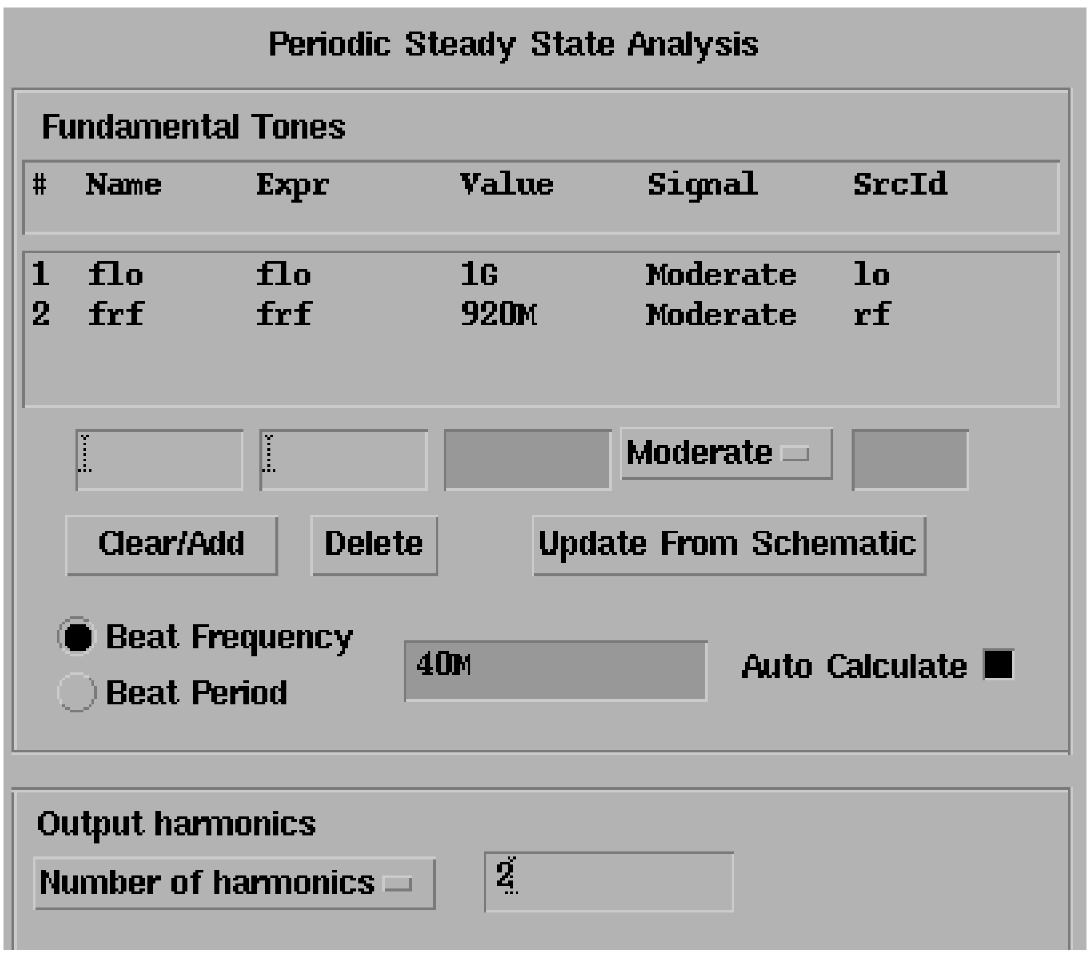
   The Choosing Analyses form appears.
2. In the Choosing Analyses form, choose *pss* for the *Analysis*.
3. In the *Fundamental Tones* list box, be sure the *Auto Calculate* button is highlighted.

The *Beat Frequency* is now displayed as `40M`. Now that the RF signal is active again, the fundamental frequency of the circuit goes back to `40 MHz`. The *Beat Frequency* button is highlighted by default.

4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 2 in the *Number of harmonics* field.

   Only two harmonics are required to determine the 1 dB compression point.

   The top of the Swept PSS Choosing Analyses form looks like the following.



5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
6. Highlight the *Sweep* button.

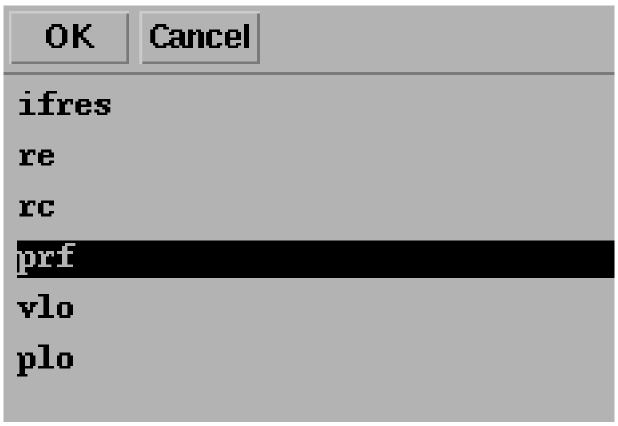
   The form changes to let you specify data for the sweep.

7. In the *Sweep* cyclic field, choose *Variable*.

8. Click on the *Select Design Variable* button.

The Select Design Variable form appears.

9. In the Select Design Variable form, highlight *prf* and click *OK*.



The *Variable Name* `prf` appears in the Choosing Analyses form. Selecting the `prf` variable sweeps RF input.

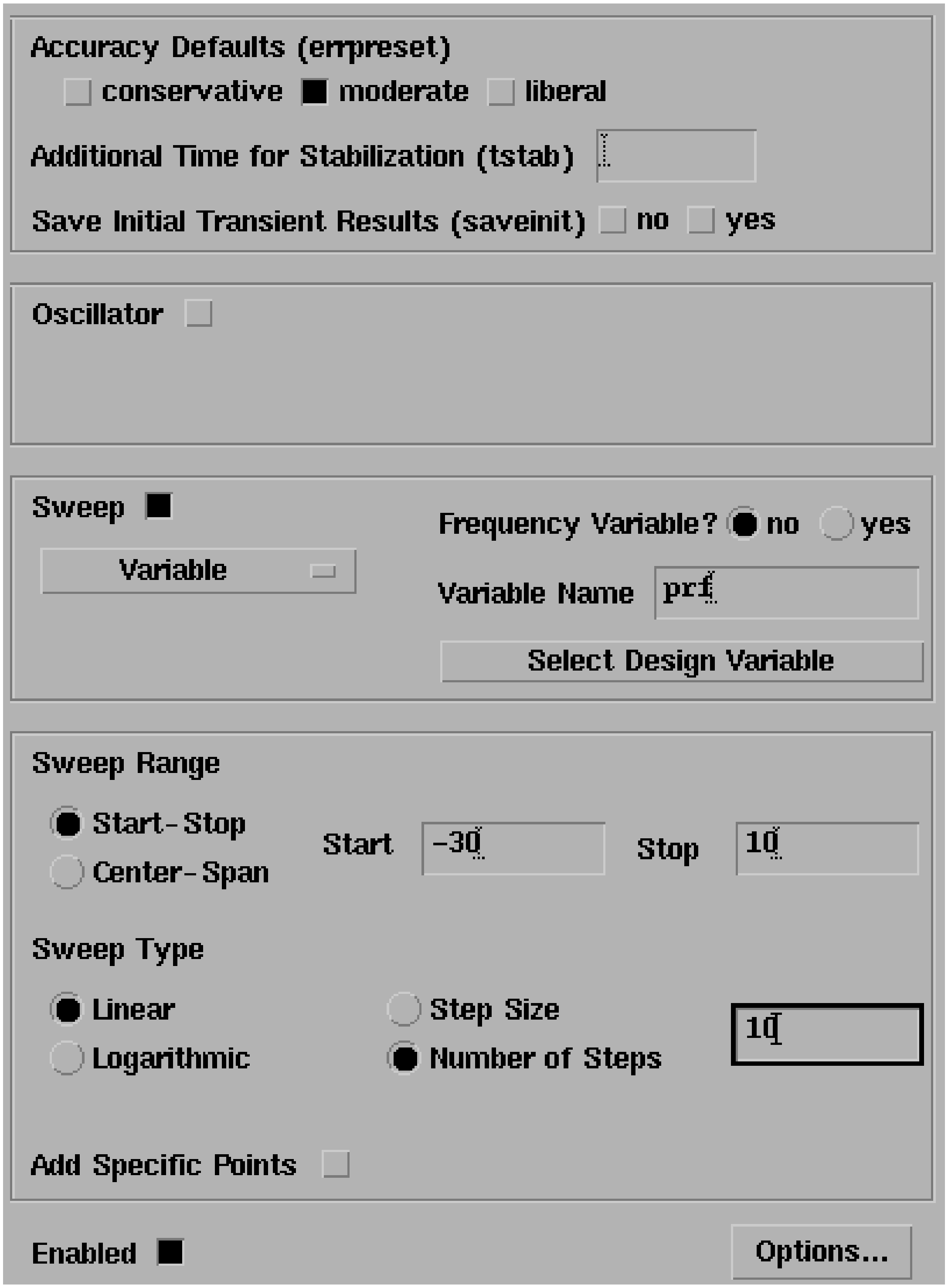
10. Choose *Start-Stop* for the *Sweep Range*, and then type `-30` in the *Start* field and `10` in the *Stop* field.

Both the signal source and the sweep are done in dBm. You learn where to set the sweep limits and how many points to include in the sweep with experience.

11. Choose *Linear* for the *Sweep Type*, and specify `10` for the *Number of Steps*.

12. Highlight *Enabled*.

The top of the Swept PSS Choosing Analyses form looks like the following.



13. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the 1 dB Compression Point

The first sideband of the IF signal is the second harmonic of the 40 MHz fundamental. To plot the 1 dB compression point, perform the following:

1. Choose *Results – Direct Plot – Main Form* in the Simulation window.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *Compression Point* for *Function*.
5. Type `1` for *Gain Compression (dB)*.
6. Type `-25` for *Input Power Extrapolation Point (dBm)*.

   This value specifies the point where the ideal amplification curve intersects the output curve. You must estimate where this point is located. If you do not specify a value, the plot defaults to the minimum variable value.

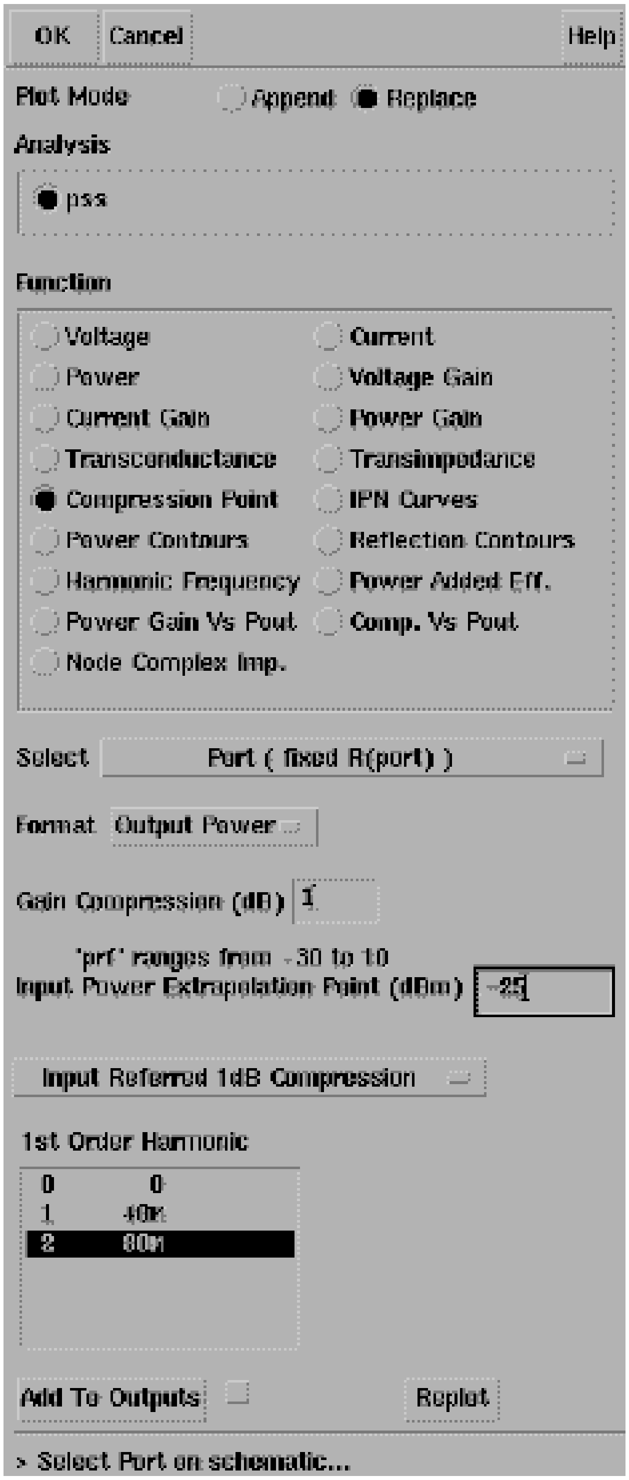
7. In the cyclic field, choose *Input Referred 1dB Compression*.
8. Follow the prompt at the bottom of the form

   ```
   Select 1st Order Harmonic on this form...
   ```

   In the *Harmonic* list box, highlight harmonic *2 (80 MHz)*.

   This is the second harmonic of the 40 MHz fundamental frequency, which is the IF frequency.

The completed Direct Plot form looks like this.

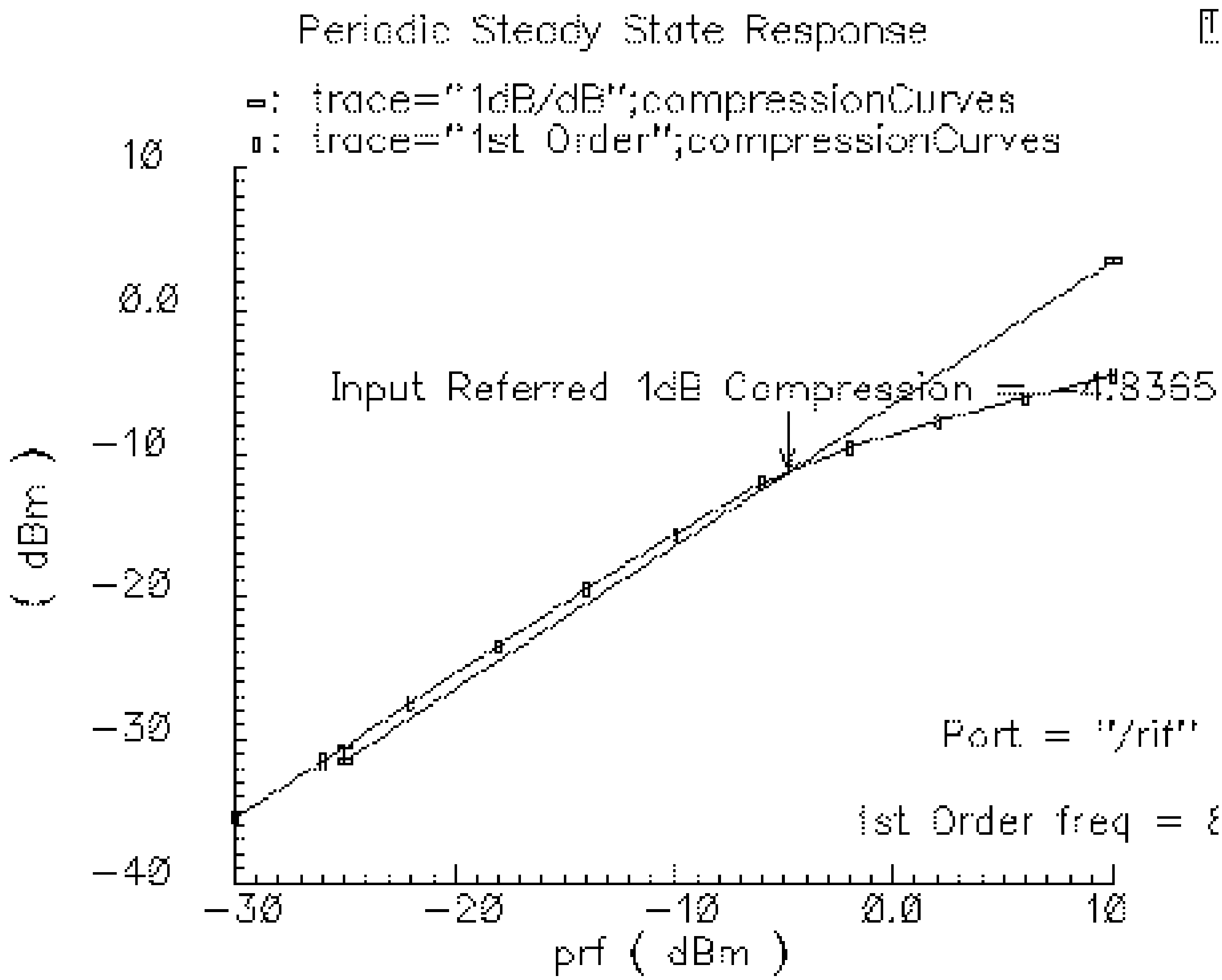


9. Follow the new prompt at the bottom of the form

```
Select Port on schematic...
```

In the Schematic window, click on the *vif* node.

The Waveform window display looks like this.



The SpectreRF simulation plots the signal curve and an ideal linear slope and automatically calculates a value of `-4.8365 dBm` for the 1 dB compression point.

If you have trouble reading the label attached to the 1 dB point, you can use the cursor to drag it to another location.

Third-Order Intercept Measurement with Swept PSS and PAC

In this example, you combine a swept PSS analysis with a Periodic AC (PAC) analysis to produce data for an IP3 plot.

This IP3 Calculation example obtains the same information as the 1 dB compression point example, but this simulation runs more quickly for two reasons

- It processes the second tone only during PAC analysis
- It considers only two of the second tone sidebands.

The swept PSS analysis for the 1 dB compression point example considered all sidebands for all signals.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `920M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, choose *Edit – Properties – Objects*.

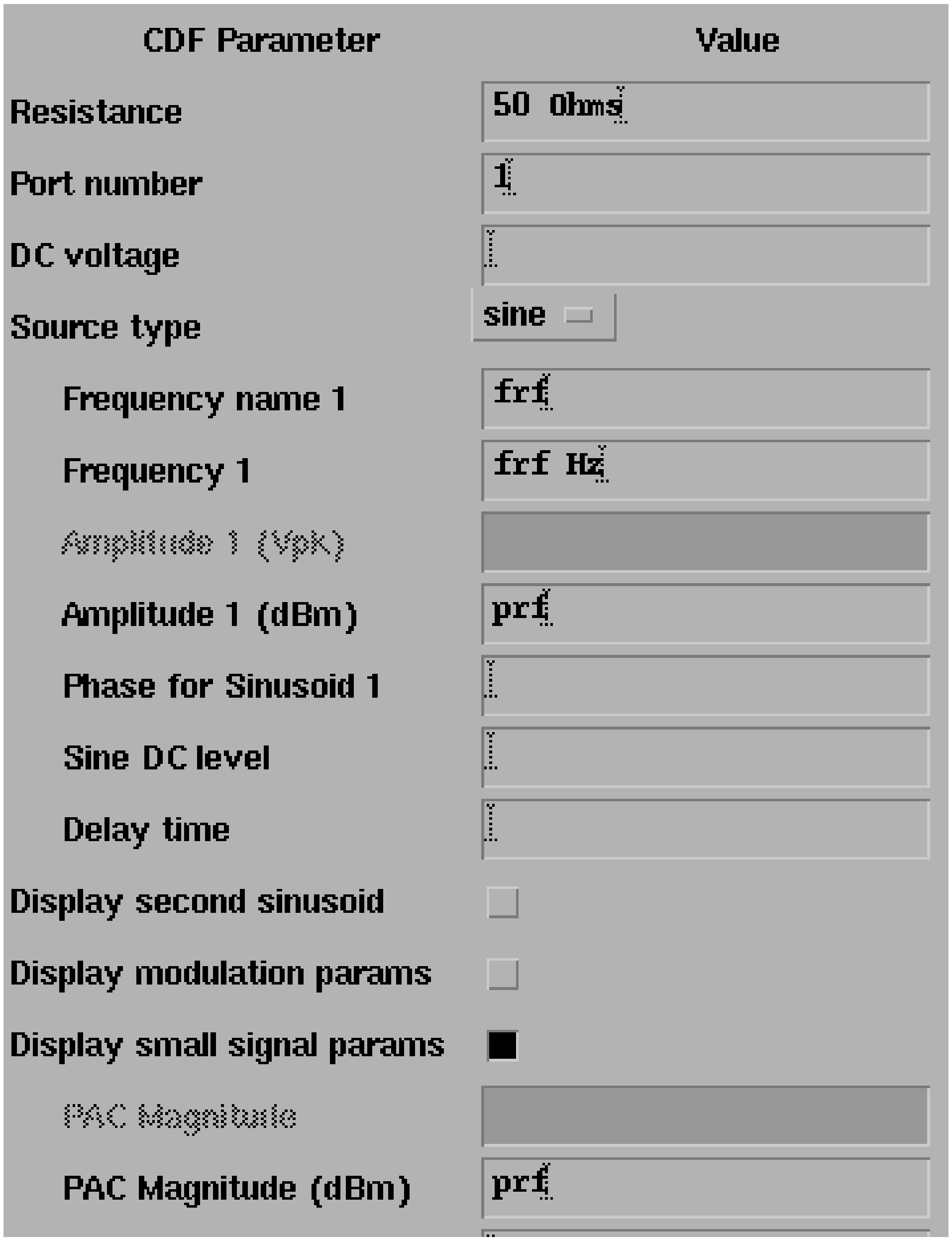
   The Edit Object Properties form appears. You use this form to modify the schematic by changing the list of CDF properties.
2. To modify the schematic for this simulation, click on the rf voltage source in the Schematic window.
3. Highlight *Display second sinusoid* and remove any values that are set from previous analyses.
4. Choose *sine* for *Source type*.
5. Type `frf` for *Frequency 1*.

6. Type `prf` in the *Amplitude 1 (dBm)* field.
7. Highlight *Display small signal params*.

   The form changes to let you specify small signal parameters.
8. Type `prf` for the *PAC Magnitude (dBm)* value.

   This simulation uses dBm values rather than magnitude.

The completed Edit Object Properties form appears like the one below.



9. Click *OK*.
10. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS and PAC Analyses

1. In the Simulation window, choose *Analyses – Choose*.

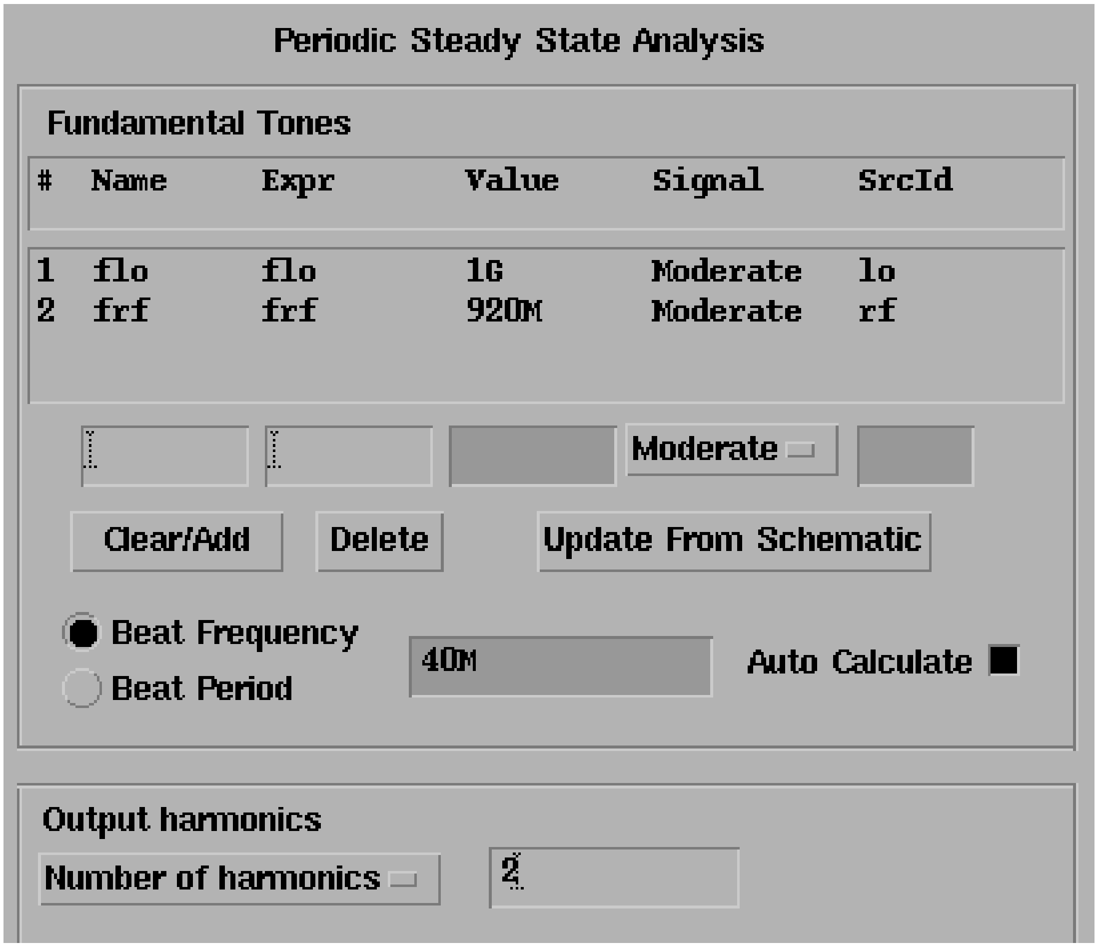
   The Choosing Analyses form appears.

Setting Up the PSS Analysis

1. In the Choosing Analyses form, choose *pss* for the *Analysis*.
2. In the *Fundamental Tones* section, be sure the *Auto Calculate* button is highlighted.

   The value 40M is specified as the *Beat Frequency*. The PAC analysis is responsible for the two tones, and the PSS analysis is now a single signal analysis. Consequently, the fundamental frequency is set to the original 40 MHz. The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type 2 in the field.

The top of the swept PSS Choosing Analyses form looks like this.



4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.
5. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
6. In the *Sweep* cyclic field, choose *Variable*.
7. Click on the *Select Design Variable* button.

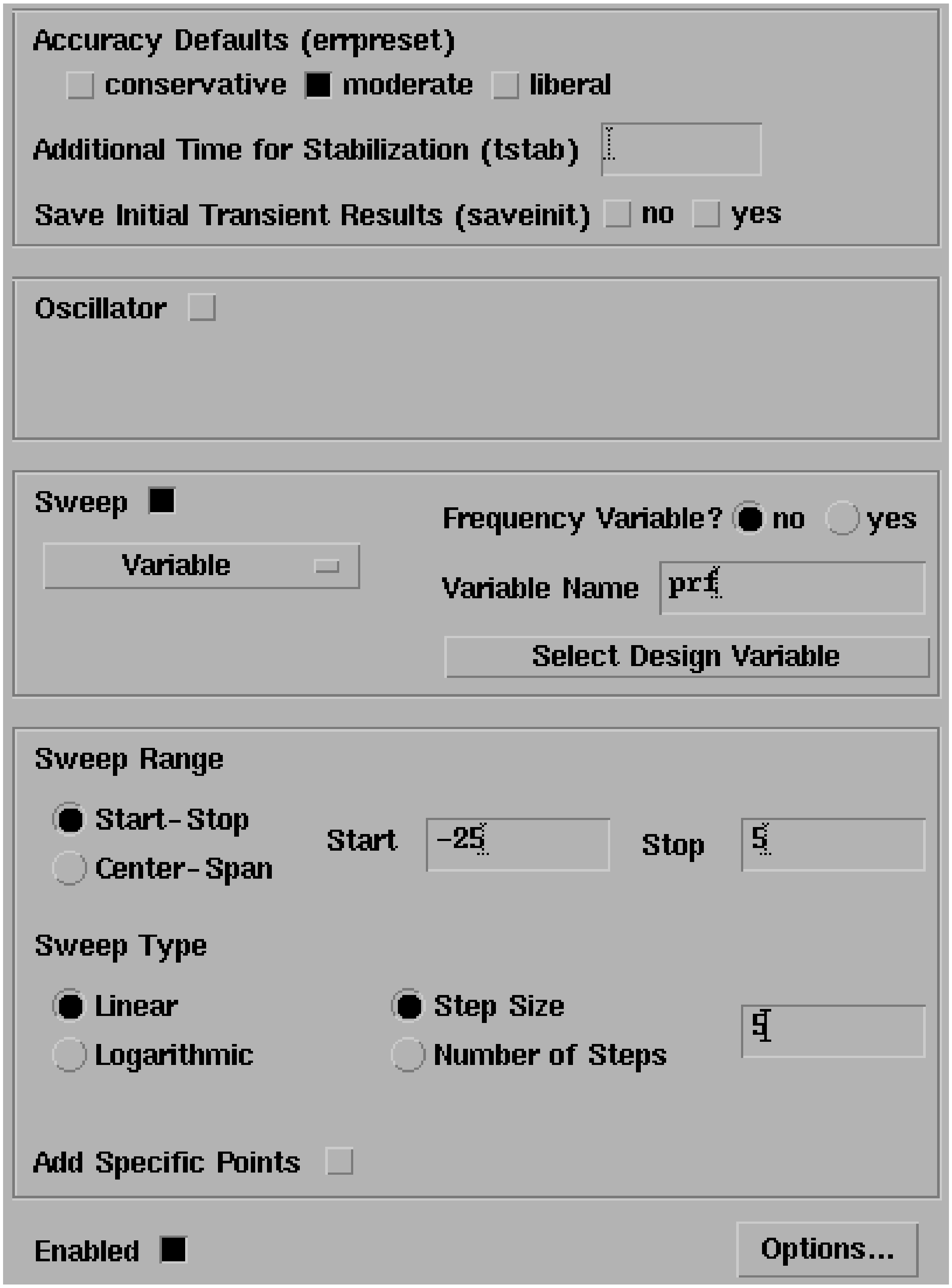
   The Select Design Variable form appears.
8. Highlight `prf` in this form and click *OK*.
9. Choose *Start-Stop* for the *Sweep Range* value, and then type `-25` in the *Start field* and 5 in the *Stop* field.

The sweep is skewed downward for a down converter.

10. Highlight *Linear* for the *Sweep Type*, highlight *Step Size,* and then type 5 in the *Step Size* field.

11. Highlight *Enabled*.

The bottom of the swept PSS Choosing Analyses form looks like this.



Setting Up the PAC Analysis

1. At the top of the Choosing Analyses form, highlight *pac*.

   The Choosing Analyses form changes to let you specify data for a *pac* analysis.

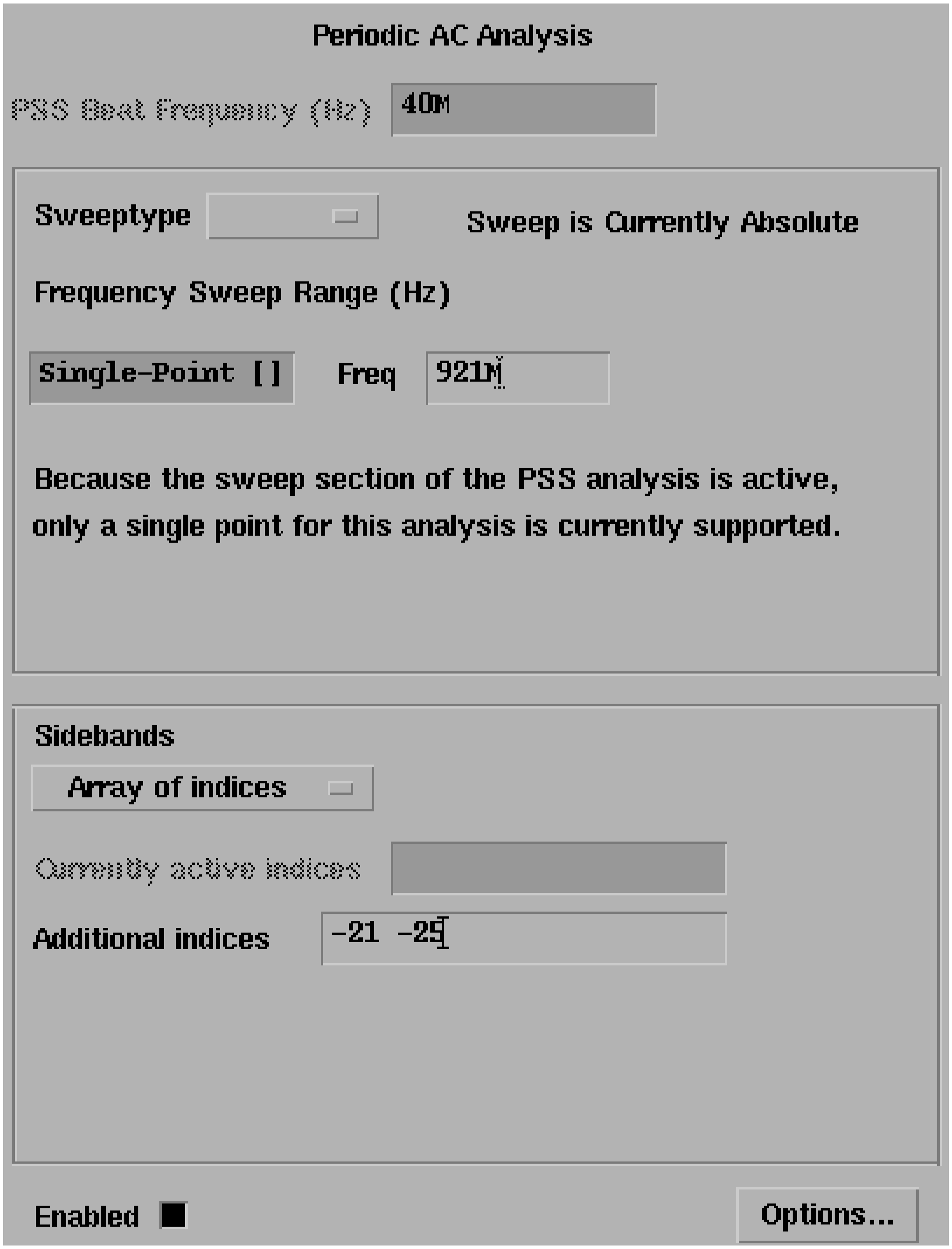
2. Type `921M` for the *Single-Point [ ] Frequency (Hz)* value.

3. In the *Sidebands* cyclic field, choose *Array of indices* and type `-21` and `-25` with a space between them.

   Given a fundamental tone of 40 MHz, the LO at 1 GHz, and two RF tones at 920 MHz and 921 MHz, the sidebands of `-25` and `-21` represent respectively the first-order harmonic of the IF output at 79 MHz (921 -25*40 = 79) and the third-order harmonic at 81 MHz (921 - 21*40 = 81).

4. Highlight the *Enabled* button.

The PAC Choosing Analyses form looks like this.



5. Click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   Note: This example compares two signals only 1MHz apart. An analysis of two signals so close together would have taken much longer with the previous analysis.

   The output log file appears and displays information about the simulation as it runs.

2. Check the output log file to be sure the simulation is completed successfully.

Plotting the IP3 Curve

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pac* for *Analysis*.

   The form changes to display information for the PAC analysis.

4. Highlight *IPN Curves* for *Function*.

   The form changes again.

5. Choose *Variable Sweep ("prf")* for *Circuit Input Power.*
6. Type `-15` for *Input Power Extrapolation Point (dBm)*.

   This value is the intercept point for the ideal amplification extrapolation. If you do not specify a value, the plot defaults to the minimum variable value.

7. In the cyclic field, choose *Input Referred IP3*.
8. Follow the prompt at the bottom of the form

   `Select 3rd Order Harmonic on this form...`

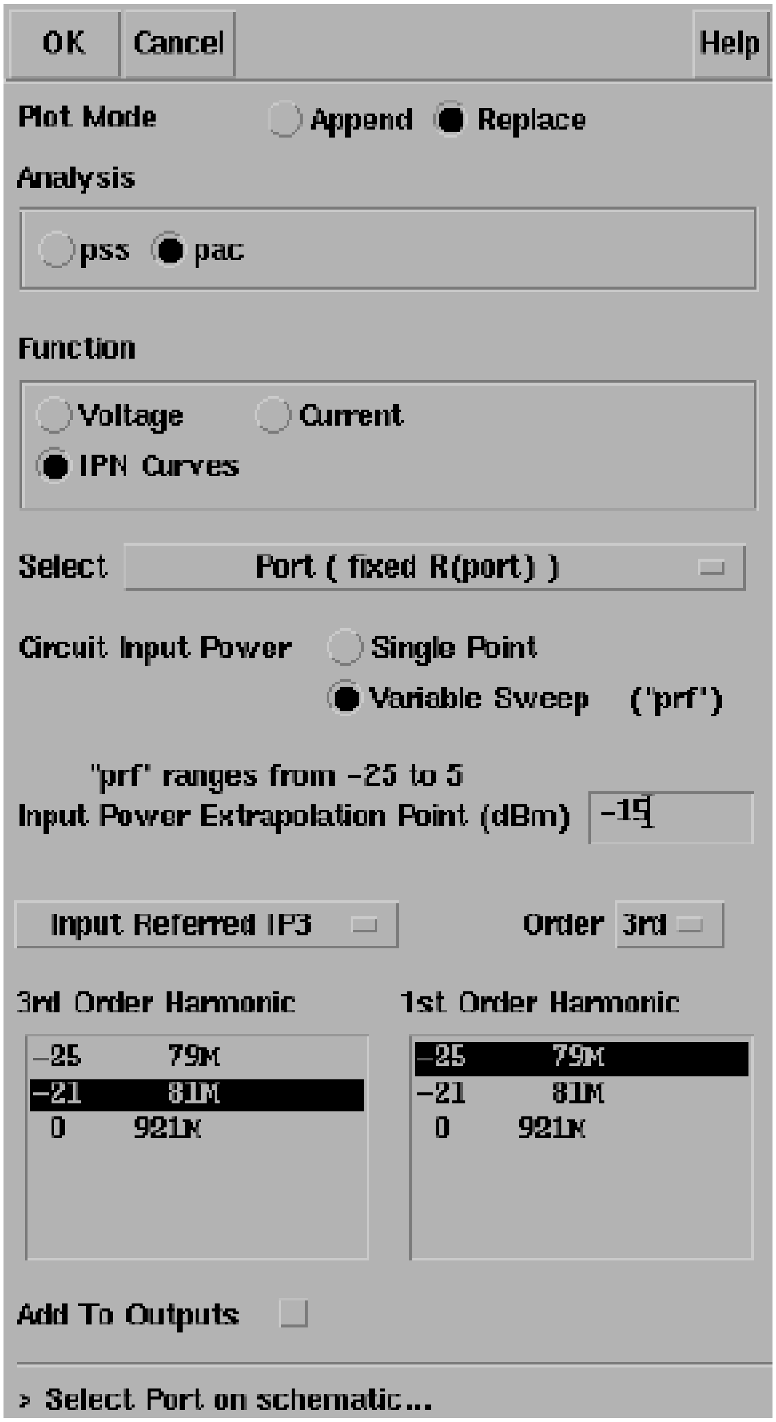
   Highlight *-21 81M* in the *3rd Order Harmonic* list box.

9. Follow the new prompt at the bottom of the form

   `Select 1st Order Harmonic on this form...`

   Highlight *-25 79M* in the *1st Order Harmonic* list box.

The completed form looks like this.

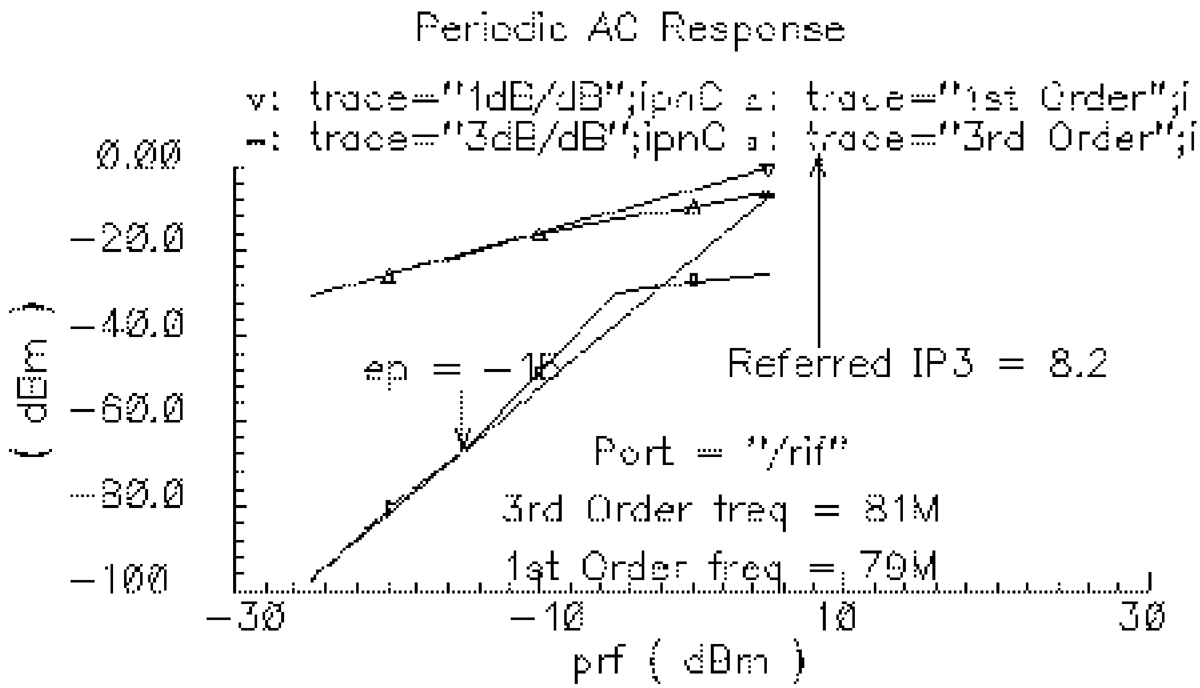


10. Follow the new prompt at the bottom of the form

`Select Port on schematic...`

In the Schematic window, click on the *Pif* signal net.

The Waveform window display appears as shown below. (You can click and drag to move the two labels with arrows, *ep* and *Input Referred IP3*, to improve visibility.)



The general equation to use for computing the third-order intercept point is

$$IP3 = V_{rf} + \frac{(V_{if(-25)} - V_{if(-21)})}{2}$$

where V_{rf} is the value in the *Start* field on the Direct Plot form.

Intermodulation Distortion Measurement with QPSS

The QPSS analysis lets you consider the effects of a few harmonics in intermodulation distortion calculations. This example describes how to run a QPSS analysis with the ne600p mixer circuit.

With the QPSS analysis, you can compute the distortion of moderately sized signals, as opposed to the small signals you investigated with the swept PSS analysis with PAC approach where you assumed there were no significant distortion effects from the small signal harmonics. With QPSS, you specify a large, or clock, signal for the fundamental and one or more moderately sized signals whose distortion effects you want to measure.

Setting Up the Simulation

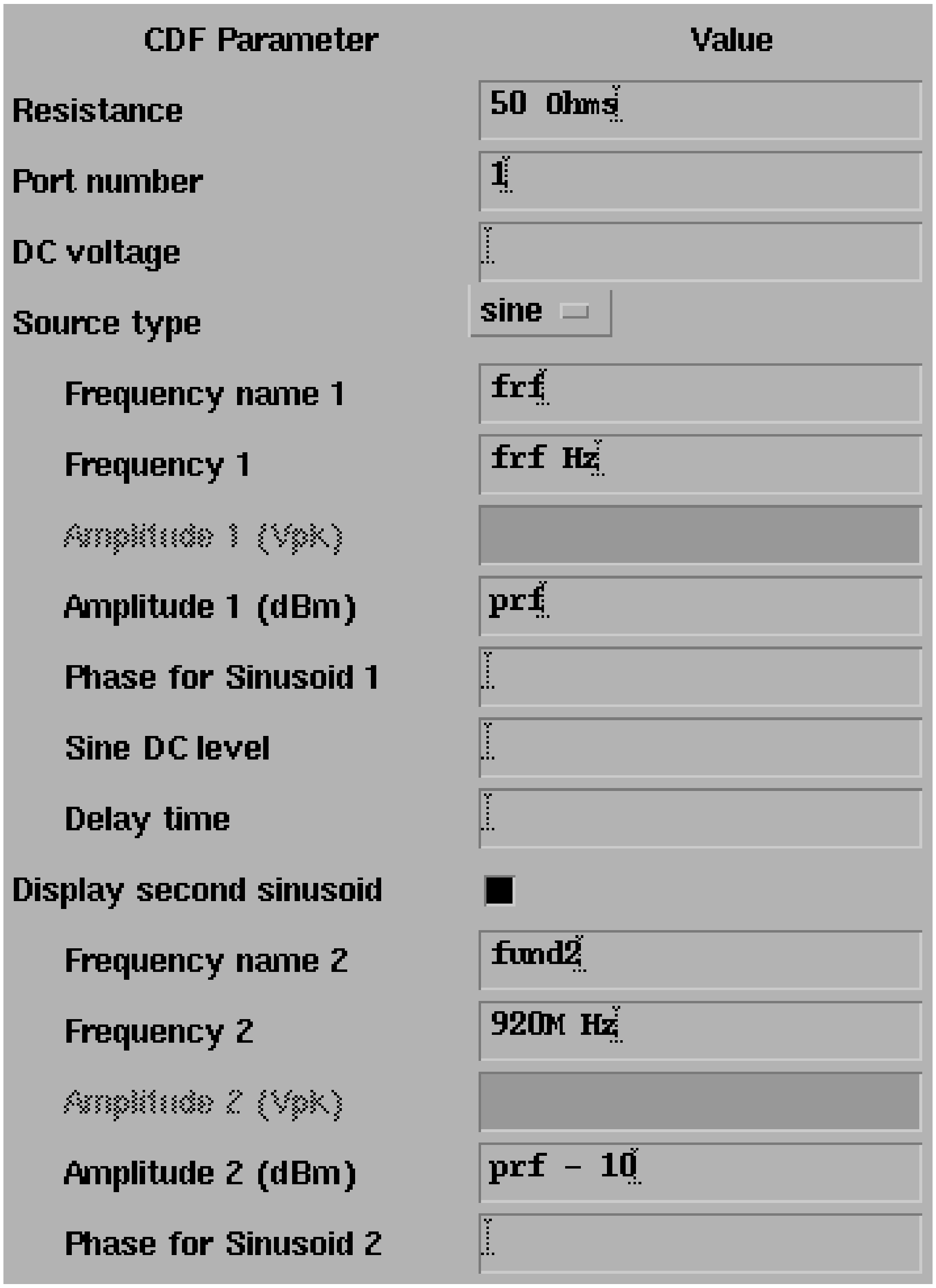
1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, choose the *rf* input port and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form for the *rf* port appears.
2. Choose *Display small signal params* and remove any remaining values from the previous analysis.
3. Choose *sine* for *Source type*.
4. Choose *Display second sinusoid* and specify the following when the forms changes to display new fields:
   a. Type `fund2`, or any name you choose, in the *Frequency name 2* field.
   b. Type `920M` for the *Frequency 2* value.
   c. Type `prf - 10` for the *Amplitude 2 (dBm)* value.
5. In the Edit Object Properties form, click *OK*.
6. In the Schematic window, choose *Design – Check and Save*.

The completed form looks like this.



CDF Parameter	Value
Resistance	50 Ohms
Port number	1
DC voltage	
Source type	sine
Frequency name 1	frf
Frequency 1	frf Hz
Amplitude 1 (Vpk)	
Amplitude 1 (dBm)	prf
Phase for Sinusoid 1	
Sine DC level	
Delay time	
Display second sinusoid	■
Frequency name 2	fund2
Frequency 2	920M Hz
Amplitude 2 (Vpk)	
Amplitude 2 (dBm)	prf - 10
Phase for Sinusoid 2	

Setting Up the QPSS Analysis

1. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

2. At the top of the Choosing Analyses form, highlight *qpss*.

   The form changes to let you specify information for the QPSS analysis. There are three tones in the *Fundamental Tones* list box, two that were present in the original specifications and one that you added.

   To update the fundamental tones, perform the following steps:

3. In the *Fundamental Tones* list box, highlight the tone named `frf`.

   The `frf` tone and its associated values appear in the fields below the list box.

4. If necessary, choose *Moderate* from the *Signal* cyclic field.

5. In the *Harms* data entry field, highlight the current value and then type `2` to specify the range of harmonics for the `frf` tone. An increase in the harmonic range for a moderate tone, such as `frf`, directly increases the simulation run time.

6. In the list box, highlight the tone named `fund2` (or whatever name you gave it).

   The new values for `frf` display in the list box. The `fund2` tone and its associated values appear in the fields below the list box.

7. If necessary, choose *Moderate* from the *Signal* cyclic field.

8. In the *Harms* data entry field, highlight the current value and then type `2` to specify the range of harmonics for the `fund2` tone.

9. In the Fundamental Tones list box, highlight the tone named `flo`.

   The new values for `fund2` display in the list box. The `flo` tone and its associated values appear in the fields below the list box.

10. Choose *Large* from the *Signal* cyclic field. In addition, highlight the current value in the *Harms* field and then type `1` to specify the range of harmonics for the `flo` tone. Click *Clear/Add*.

    The new values for `flo` display in the list box. the harmonic range of 1 gives 3 harmonics (-1, 0,1) for the large tone.

    By choosing *Large* as the *Signal* value, you specify `flo` to be the *Large* or clock signal. Each QPSS analysis must have one tone set to be the *Large* signal. When you choose a signal for the large signal, choose a tone that is larger, less sinusoidal, or more nonlinear than the other tones.

You must choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with an harmonic range (*Harms*) value of 0 is ignored by the simulation. In general,

- The harmonic range (*Harms)* value for the *Large* tone should generally be at least 5 which guarantees11 harmonics. In some cases, for example in a down converting situation such as this one, *Harms* can be as low as 1. For a down-converting situation such as this, 1 is generally enough.

- The *Harms* value for *Moderate* tones should be approximately 2 or 3. For *Moderate* tones, increasing the *Harms* value increases the simulation run time.

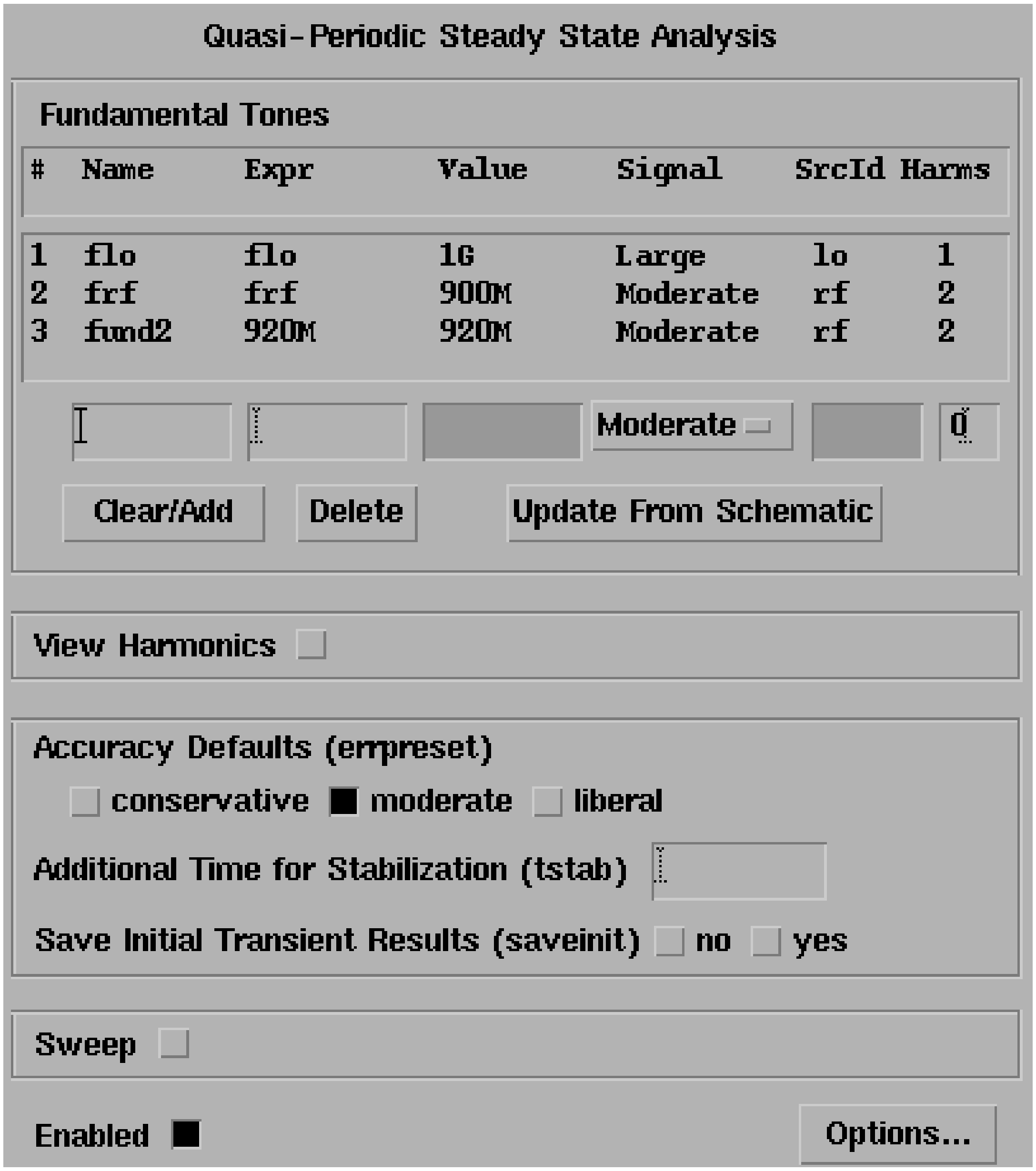
  Setting the *Harms* value to 2, yields up to the 3rd order intermodulation terms; setting the *Harms* value to 3, yields up to the 5th intermodulaiton terms. For higher order intermodulation terms, increase the *Harms* value accordingly. However, you should avoid setting the *Harms* value unnecessarily high since it increases the simulation time.

When, as in this example, you specify *Harms* values of 1 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [1,2]` which gives you 3 harmonics (-1, 0, 1) for the *Large* tone and 5 harmonics (-2, -1, 0, 1, 2) for the *Moderate* tone. As a result, you get noise from 3 x 5 = 15 frequency sidebands.

11. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

12. Be sure the *Enabled* button is highlighted.

The completed form appears like the one below.



13. Click *OK*.

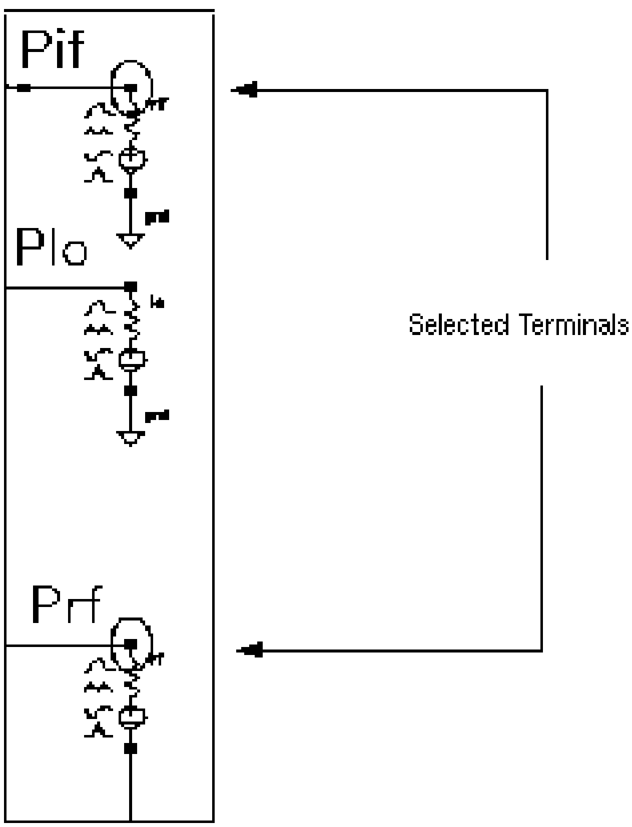
Selecting Simulation Outputs

1. In the Simulation window, choose *Outputs – To Be Saved – Select on Schematic*.

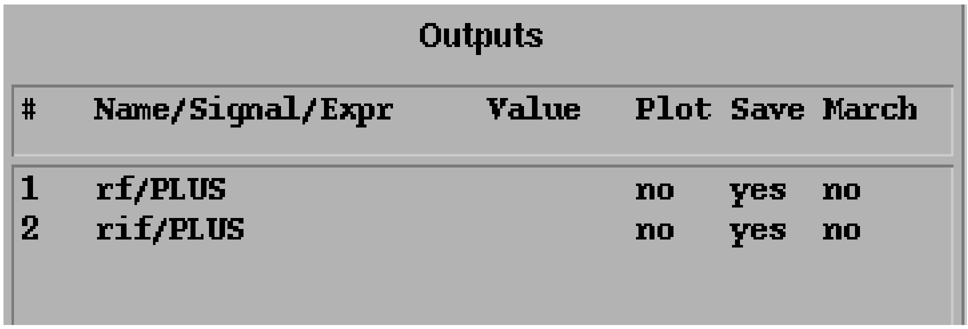
If you want to plot current or power at the end of the simulation, you must explicitly save the currents necessary for the calculations. The most economical way to do this, in terms of simulation time, is to choose specific currents on the schematic.

2. In the Schematic window, click on the appropriate terminals to choose *rf* and *rif*.

   The terminals are circled in the Schematic window after you choose them.



The selected terminals also display in the Simulation window.



Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message that says the simulation completed successfully.

   Note: QPSS sends less information to the output log than most other SpectreRF analyses. While QPSS is running for this analysis, there might be periods of several minutes during which there are no messages. This does not mean that the simulation is not progressing.

Plotting the Voltage and Power

➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

The Direct Plot form and the Waveform window appear.

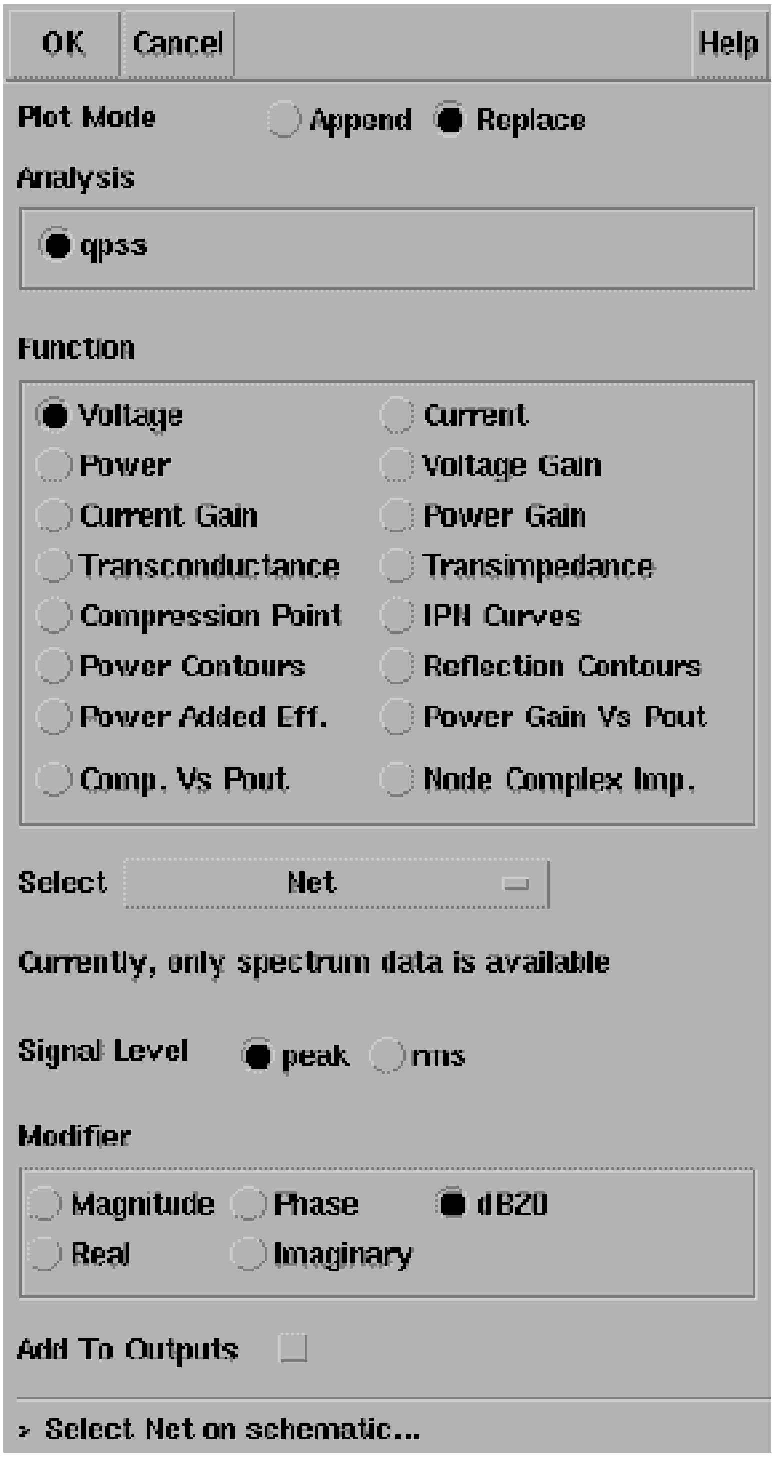
Plotting Voltage

To plot voltage do the following in the Direct Plot form:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *qpss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Note that the *Select* cyclic field displays *Net*.
5. Highlight *peak* for *Signal Level*.

6. Highlight *dB20* for *Modifier*.

   The completed form looks like this.

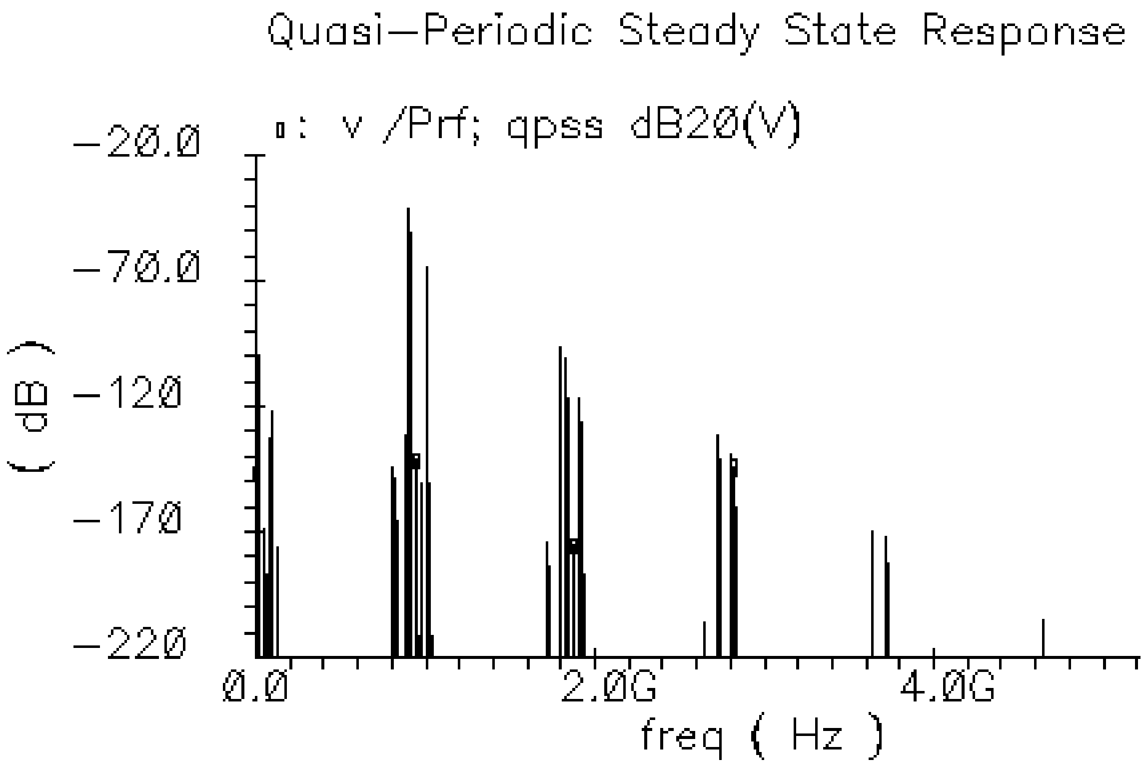


7. Follow the prompt at the bottom of the form

`Select Net on schematic...`

Click on the net connecting to the *rf* terminal.

The voltage plot for the *rf* terminal displays in the Waveform window.



Because this is a down-converting mixer, the cluster of frequencies from 20 M to 120 M near the Y axis of the plot are of interest. To find a frequency and its associated voltage, place the cursor at the tip of a line. The X and Y values that represent the frequency and the voltage, respectively, appear at the top left corner of the Waveform window. If you place the cursor at the points shown above, you display the following voltage and frequency values:

Difference Translation	Frequency	Voltage
920M - 900M	20M	-100.2V
1G - 920M	80M	-132.5V
1G - 900M	100M	-122.6V

Plotting Power

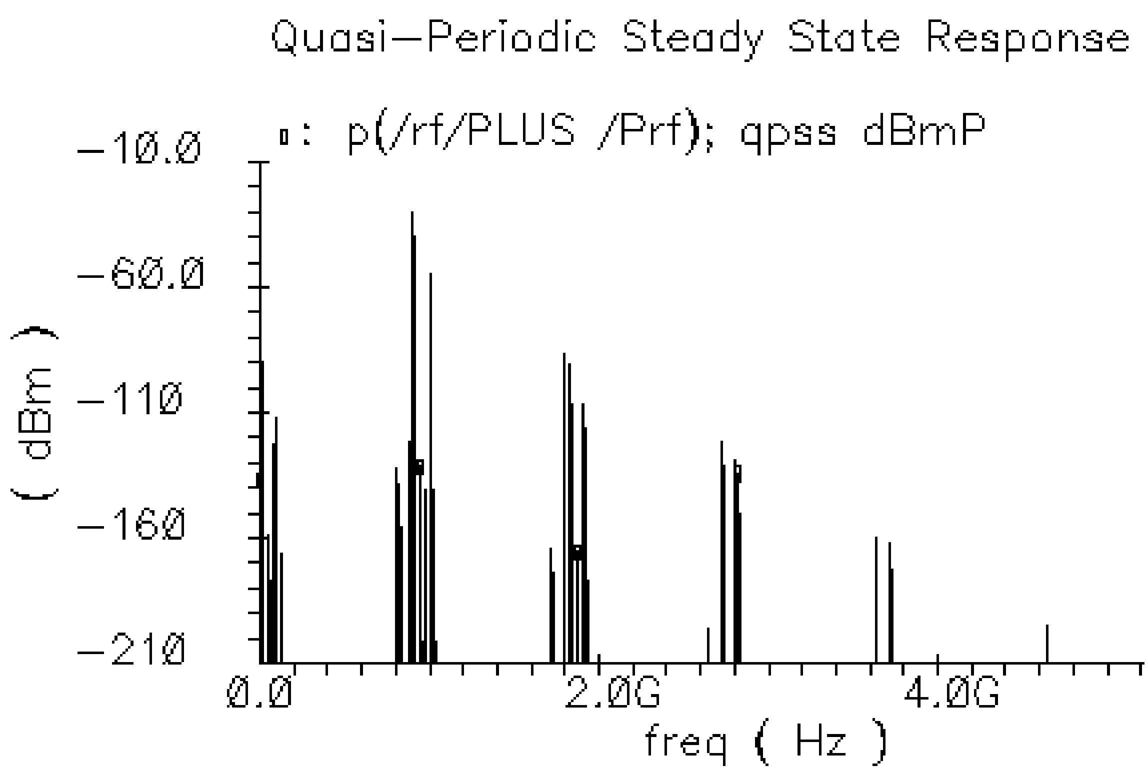
To plot the power, do the following in the Direct Plot form:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *Power* for *Function*.
3. Note the *Select* cyclic field displays *Terminal*.
4. Highlight *dBm* for *Modifier*.
5. Following the message at the bottom of the form

   `Select Instance Terminal on schematic...`

   In the Schematic window, click on the *rf* terminal.

   The plot for the power appears in the Waveform window.



6. If you want to see the voltage and power for the *rif* terminal, repeat the steps to create the plots for the *rif* terminal, but choose the *rif* terminal in the schematic.

   Note: If you want to run simulations in other chapters, reset the design variable *frf* to `920 M`.

Noise Figure with QPSS and QPnoise

You can combine a QPSS analysis with a small-signal QPnoise analysis to determine the degradation of the noise figure for the ne600p mixer circuit caused by the two interfering signals present at the input of the mixer. The noise figure for the ne600p mixer without the interferer signals is determined in "Noise Figure Measurement with PSS and Pnoise" on page 250. The QPSS and QPnoise analyses described in this example include the interferer signals which allow more noise foldings to occur into the output IF band.

The Quasi-Periodic Noise, or QPnoise analysis, is similar to the conventional noise analysis, except that it includes frequency conversion and intermodulation effects. Hence is it useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically or quasi-periodically driven circuits.

Setting Up the Simulation

1. If necessary, open the ne600p mixer circuit.
2. If necessary, specify the full path to the model file in the Model File Set-up form.
3. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)
4. If necessary, set the design variables `frf` to `900M` and `flo` to `1G`. (Check the Simulation window to verify the current design variable values.)

Editing the Schematic

1. In the Schematic window, click on the *rf* input port and then choose *Edit – Properties – Objects.*

   The Edit Object Properties form for the *rf* input port appears. You use this form to change the list of CDF properties and modify the schematic for this simulation.

In the Edit Object Properties form, do the following:

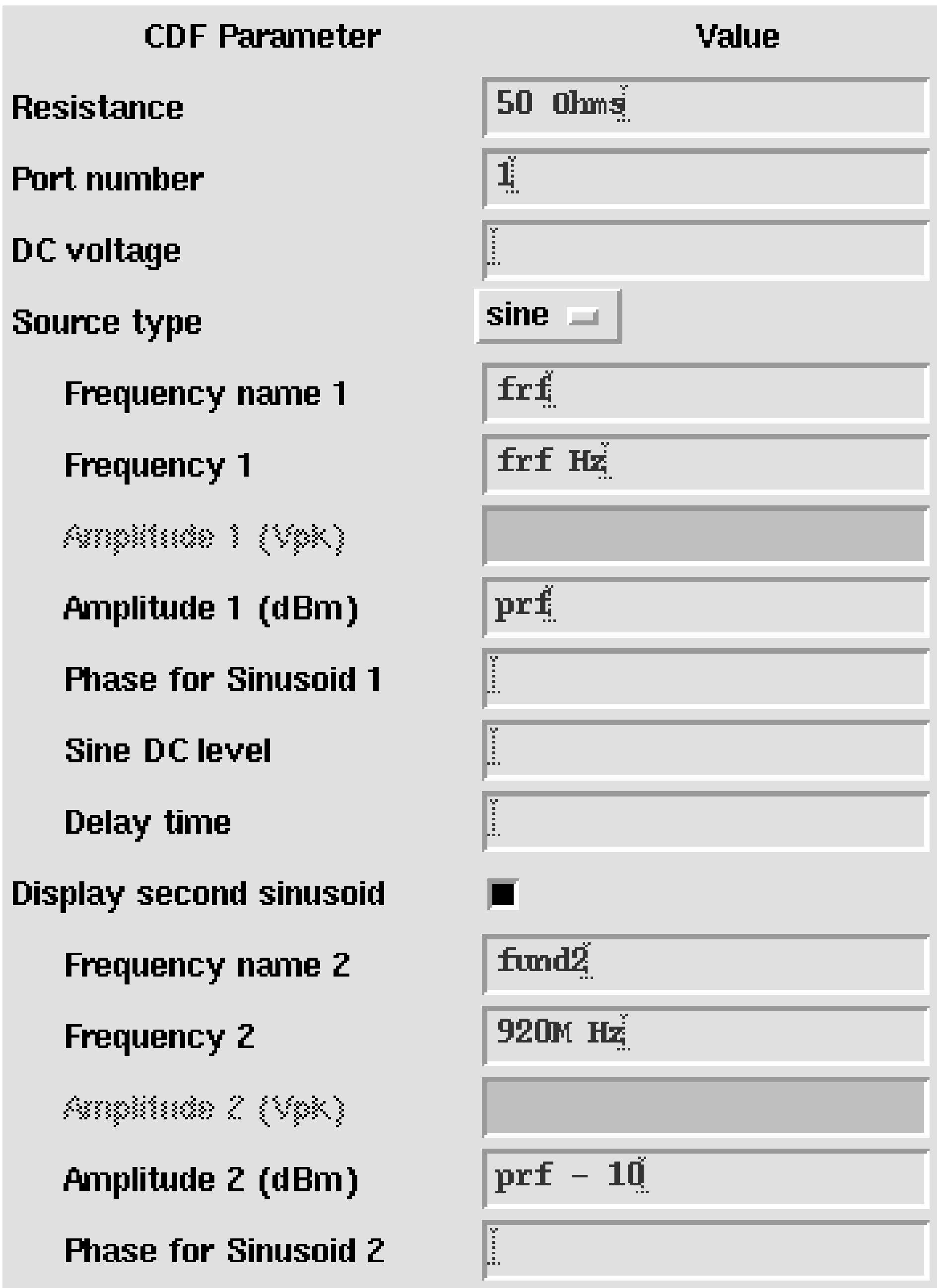
1. Choose *Display small signal params* and remove any remaining values from the previous analysis.
2. Since the `rf` signal is moderate signal, choose *sine* for *Source type*.
3. Choose *Display second sinusoid* and specify the following when the forms changes to display new fields:

a. Type `fund2`, or any name you choose, in the *Frequency name 2* field.

b. Type `920M` for the *Frequency 2* value.

c. Type `prf - 10` for the *Amplitude 2 (dBm)* value.

The completed form looks like this.



CDF Parameter	Value
Resistance	50 Ohms
Port number	1
DC voltage	
Source type	sine
Frequency name 1	frf
Frequency 1	frf Hz
Amplitude 1 (Vpk)	
Amplitude 1 (dBm)	prf
Phase for Sinusoid 1	
Sine DC level	
Delay time	
Display second sinusoid	■
Frequency name 2	fund2
Frequency 2	920M Hz
Amplitude 2 (Vpk)	
Amplitude 2 (dBm)	prf - 10
Phase for Sinusoid 2	

4. In the Edit Object Properties form, click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Setting Up the QPSS and QPnoise Analyses

In the Choosing Analyses form, set up two analyses, first *qpss* then *qpnoise*.

➤ In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting Up the QPSS Analysis

1. At the top of the Choosing Analyses form, highlight *qpss*.

   The form changes to let you specify information for the QPSS analysis.

2. In the *Fundamental Tones* list box, highlight the tone named `frf`.

   The `frf` tone and its associated values appear in the fields below the list box.

3. If necessary, choose *Moderate* from the *Signal* cyclic field.
4. In the *Harms* data entry field, highlight the current value and then type 2 to specify the range of harmonics for the `frf` tone. An increase in the harmonic range for a moderate tone directly increases the simulation run time.
5. In the list box, highlight the tone named `fund2`.

   The new values for `frf` display in the list box. The `fund2` tone and its associated values appear in the fields below the list box.

6. If necessary, choose *Moderate* from the *Signal* cyclic field.
7. In the *Harms* data entry field, highlight the current value and then type 2 to specify the range of harmonics for the `fund2` tone.
8. In the list box, highlight the tone named `flo`.

   The new values for `fund2` display in the list box. The `flo` tone and its associated values appear in the fields below the list box.

9. Choose *Large* from the *Signal* cyclic field. In addition, highlight the current value in the *Harms* field and type 5 to specify the range of harmonics for the `flo` tone. Click *Clear/Add*.

The new values for `flo` display in the list box. The harmonic range of 5 gives 11 harmonics (-5, ... ,0, ... +5) for the large tone. In this case where a QPnoise analysis is following the QPSS analysis, the *Harms* value must be sufficiently large to guarantee an accurate noise analysis. Generally, 5 is a sufficiently large *Harms* value. Entering a large *Harms* value for the large tone does not affect the simulation run time.

By choosing *Large* as the *Signal* value, you specify `flo` to be the large or clock signal. Each QPSS analysis must have one tone set to be the large signal. When you choose a large signal, select a tone that is larger, less sinusoidal, or more nonlinear than the other tones.

You must choose at least one harmonic for each signal that you want to include in a QPSS analysis. A signal with a harmonic range value of 0 is ignored by the simulation. In general,

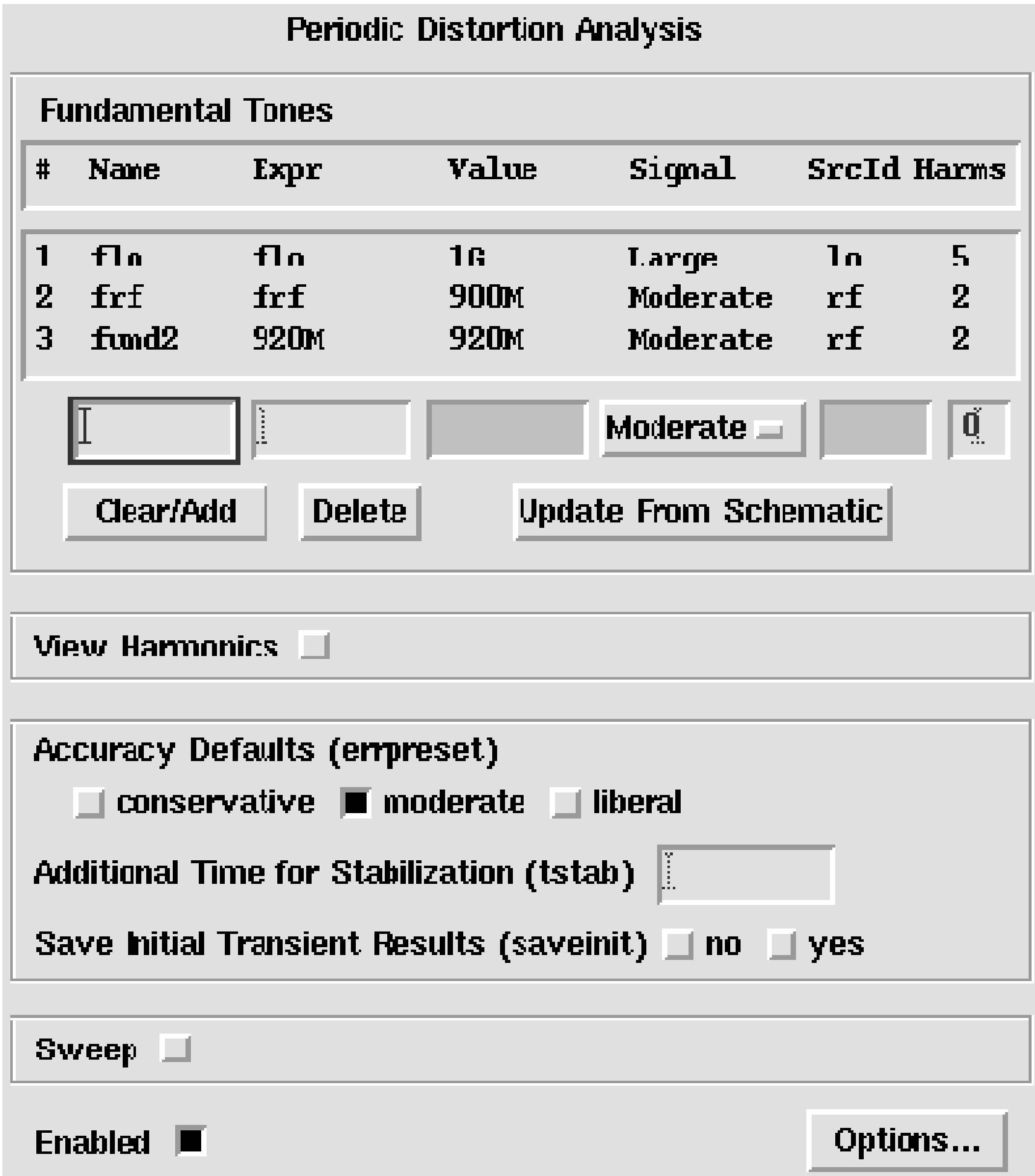
- ❑ The harmonic range (*Harms)* value for the *Large* tone should be at least 5, guaranteeing 11 harmonics.
- ❑ The *Harms* value for *Moderate* tones should be approximately 2 or 3. For *Moderate* tones, increasing the *Harms* value increases the simulation run time.

When, as in this example, you specify *Harms* values of 5 for the *Large* signal and 2 for the *Moderate* signals, you get `maxharms = [5,2]` which gives you 11 harmonics for the Large tone and 5 harmonics (-2, -1, 0, 1, 2) for the Moderate tone. As a result, you get noise from 11 x 5 = 55 frequency sidebands. SpectreRF considers all these combinations when performing QPnoise calculations.

10. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

11. Be sure the *Enabled* button is highlighted.

The completed form appears like the one below.



Setting Up the QPnoise Analysis

1. In the Analysis section at the top of the Choosing Analyses form, highlight *qpnoise.*

   The Choosing Analyses form changes to let you specify data for a Quasi-Periodic Noise Analysis.

2. In the *Frequency Sweep Range (Hz)* cyclic field, choose *Start-Stop.*
3. Type `50M` in the *Start* field and `150M` in the *Stop* field.

   This frequency range covers the frequencies of interest.
4. In the *Sweep Type* cyclic field, choose *Linear* for the sweep type and highlight *Number of Steps*. Type `20` in the *Number of Steps* field.
5. In the *Sidebands* cyclic field, choose *Maximum clock order* and type `5`.

   Maximum clock order is related to large signal harmonics range. A *Maximum clock order* of 5 insures that sufficient large-signal harmonics are processed. You can increase this value.
6. In the *Output* cyclic field, choose *probe* for the *Output* value.
7. Click on the *Output Probe Instance Select* button. Then click on the appropriate component in the Schematic window to choose *Pif*.

   `/rif` appears in the *Output Probe Instance* field.
8. In the *Input Source* cyclic field, choose *port*.
9. Click on the *Input Port Source Select* button. Then click on the appropriate component in the Schematic window to choose *rf*.

   `/rf` appears in the *Input Port Source* field.
10. Type `1 0 0` in the *Reference Sideband* field. One entry for each tone separated by spaces.

    In the *Reference Sideband* field, enter a vector whose indices have a one-to-one correspondence with the fundamental tones. For this example, the correspondence is illustrated in the following table.

Ref Sideband Vector	1	0	0
Fundamental Tones	LO	FUND2	RF
Frequencies	1 GHz	920 MHz	900 MHz

    The reference sideband vector `1 0 0` mixes with the LO tone only to get the output.
11. Highlight *freq* for the *refsidebandoption.*

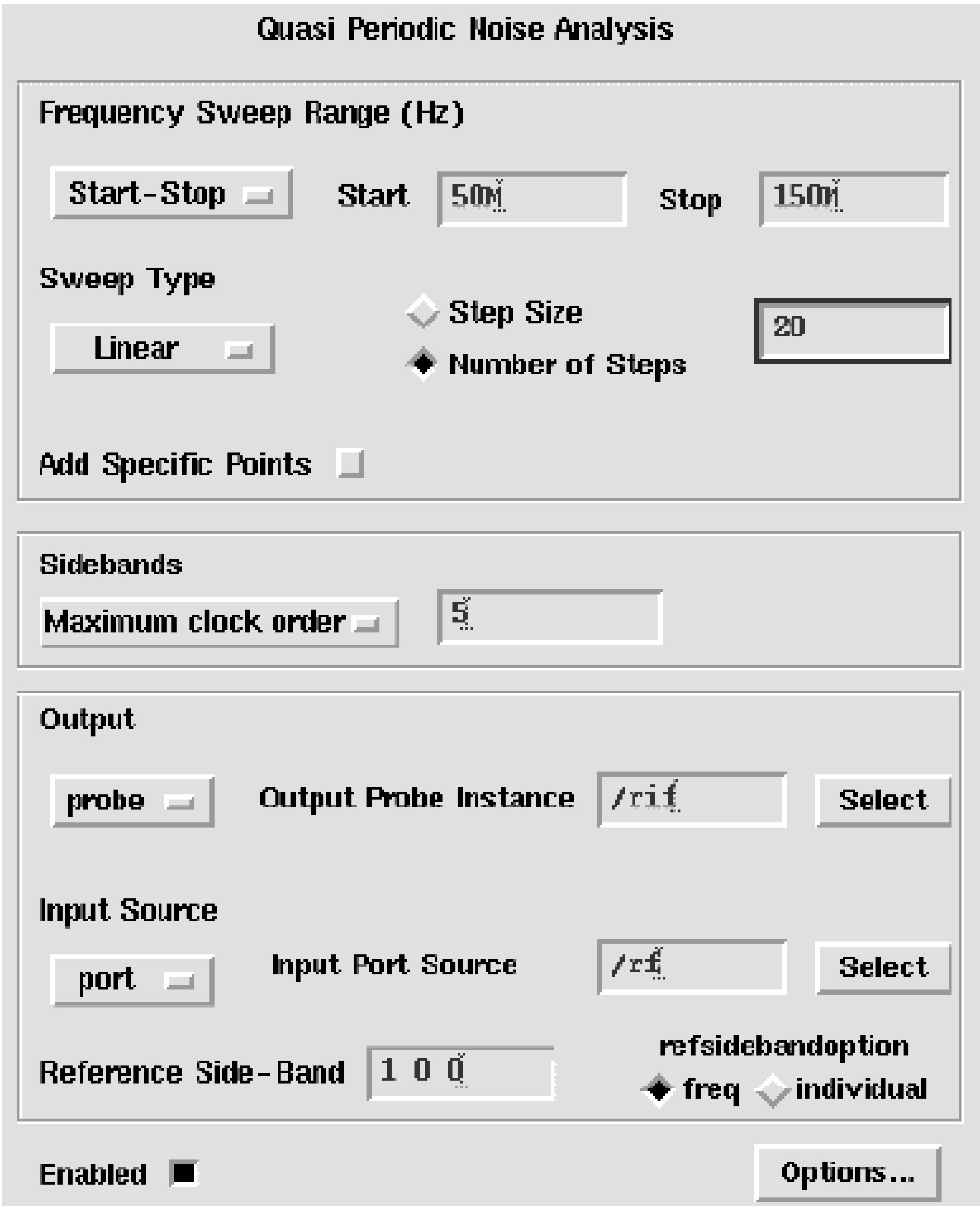
    This selects the frequency as the reference and the reference sideband frequency shift is computed relative to all combinations of fundamental tones.

When the frequency shift between input and output is calculated, the fundamental tones are multiplied by the *Reference Sideband* vector entries and summed. Since different combinations might lead to the same frequency shift, *refsidebandoption* specifies how to perform this calculation.

- ❑ As in this example, you select *freq* to calculate frequency shift based on all possible combinations.
- ❑ Select *individual* when you want to calculate frequency shift based on the one specified combination only.

12. Be sure the *Enabled* button is highlighted.

The completed *Quasi Periodic Noise Analysis* section of the Choosing Analyses form appears like the one below.



13. In the Choosing Analyses form, click *OK*.

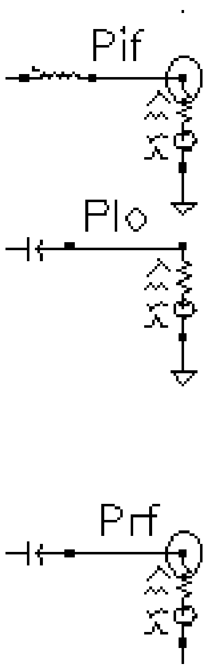
Selecting Simulation Outputs

1. In the Simulation window, choose *Outputs – To Be Saved – Select on Schematic*.

   If you want to plot current or power at the end of the simulation, you must explicitly save the currents necessary for the calculations. The most economical way to do this, in terms of simulation time, is to select the specific currents to save on the schematic.

2. In the Schematic window, click on the appropriate terminals to choose *rf* and *rif*.

   The terminals are circled in the Schematic window after you choose them.



The selected terminals also appear in the Simulation window.

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	rif/PLUS		no	yes	no
2	rf/PLUS		no	yes	no

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message that says the simulation completed successfully.

   Note: QPSS analysis sends less information to the output log than most other SpectreRF simulator analyses. While QPSS is running for this analysis, there might be periods of several minutes during which there are no messages. This does not mean that the simulation is not progressing.

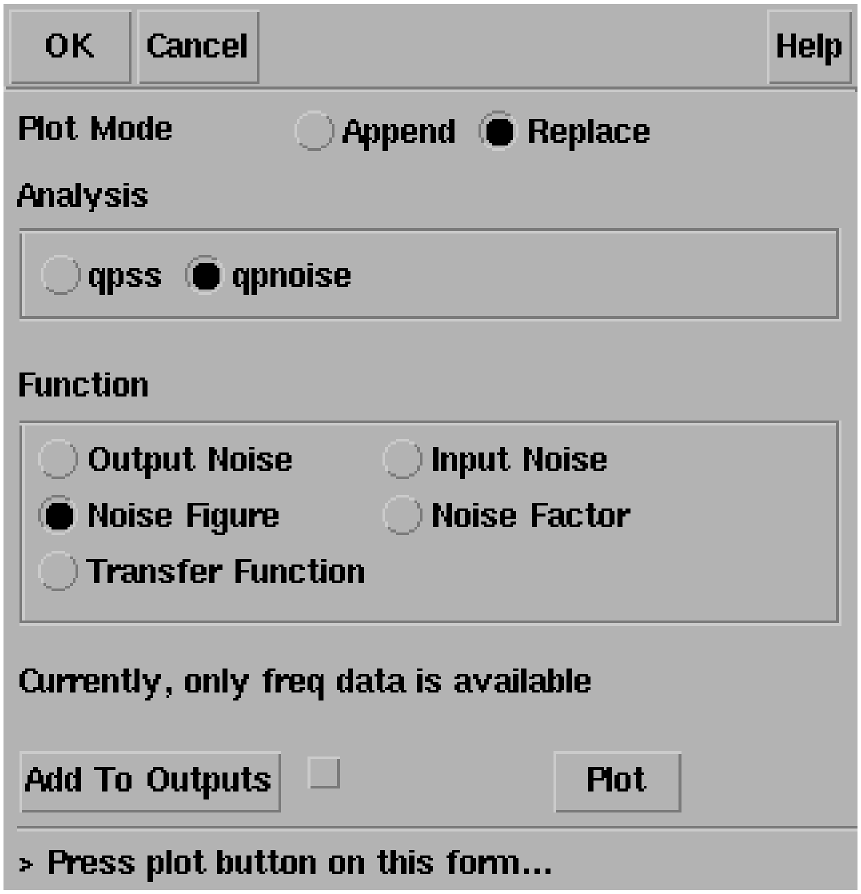
Plotting the Noise Figure

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form and the Waveform window appear.

2. Highlight *Replace* for *Plot Mode*.
3. Highlight *qpnoise* for *Analysis*.
4. Highlight *Noise Figure* for *Function*.

The completed Direct Plot form for Qpnoise appears like the one below.



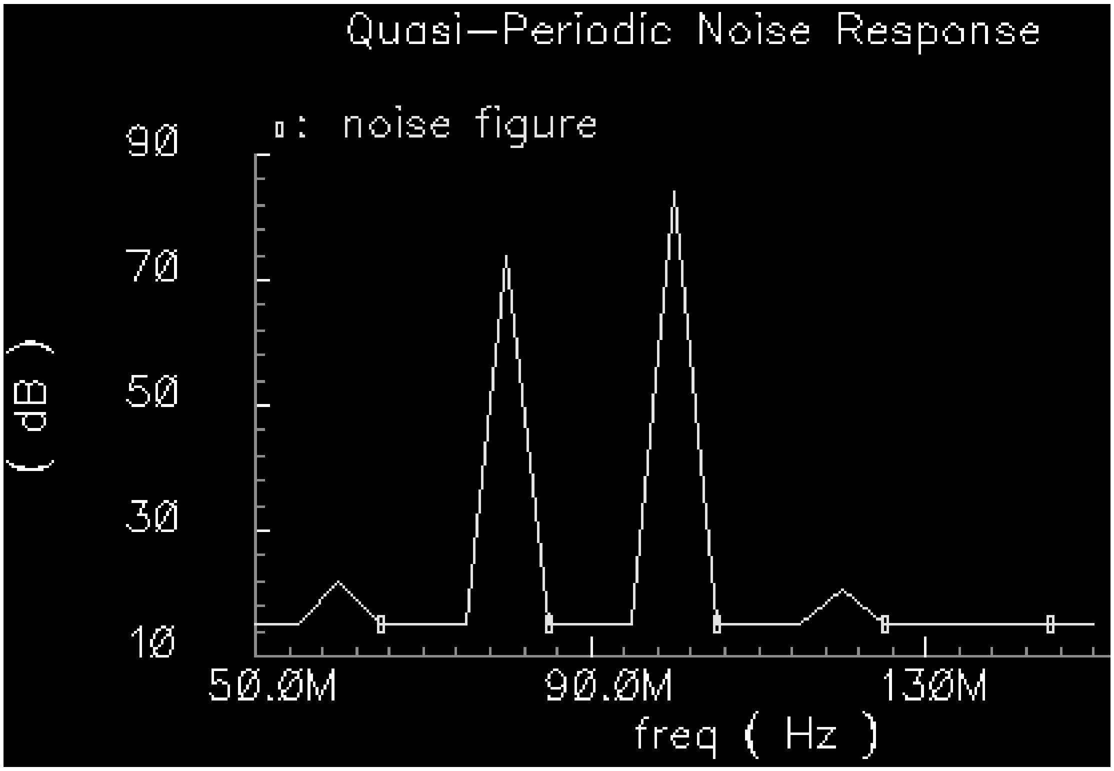
5. Follow the prompt at the bottom of the form

`Press plot button on this form...`

Click on *Plot* in the Direct Plot form.

The plot appears in the Waveform window.

The Waveform window appears like the one below:



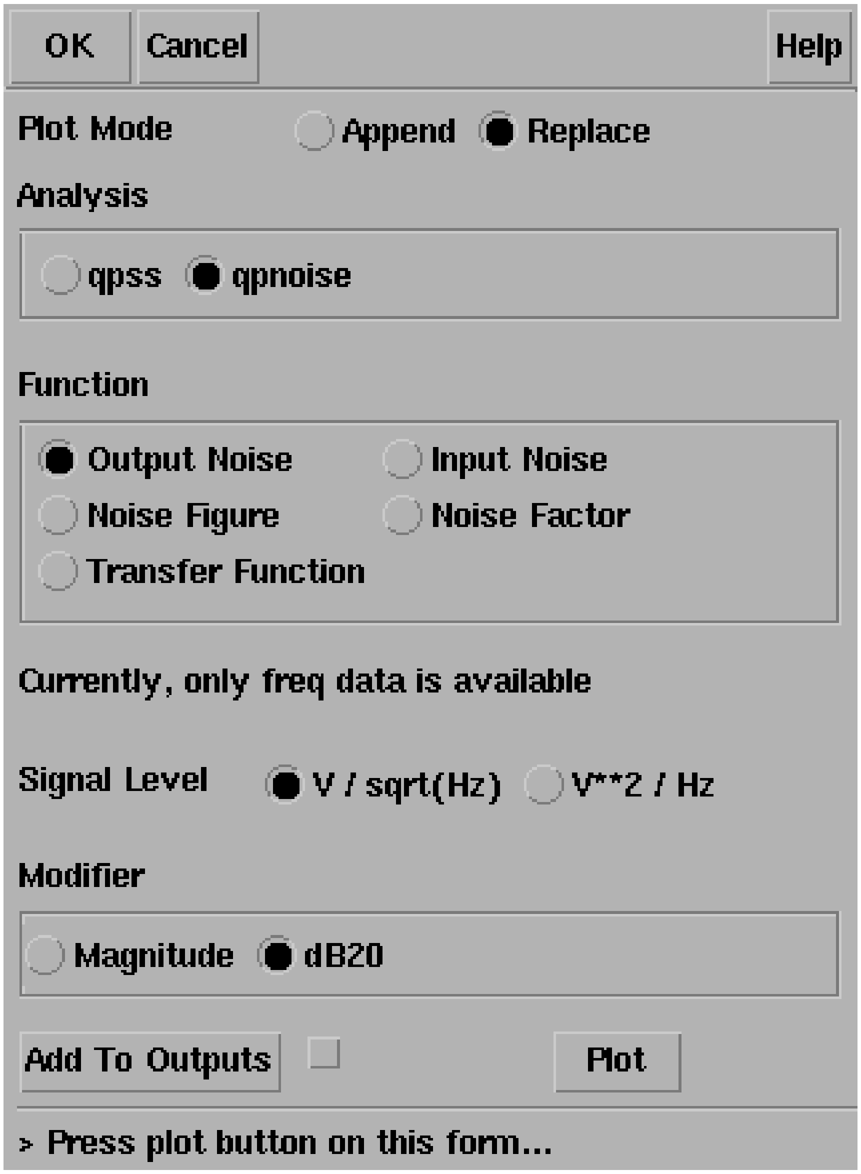
6. To determine the noise figure at different frequencies, move the cursor along the noise figure curve in the Waveform window. In the above plot, the noise figure is about `73.24dB` at `80 MHz`.

Plotting the Output Noise

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *qpnoise* for *Analysis*.
3. Highlight *Output Noise* for *Function*.
4. Highlight V/sqrt(Hz) for Signal Level.
5. Highlight *dB20* for *Modifier*.

The completed Direct Plot form appears like the one below.



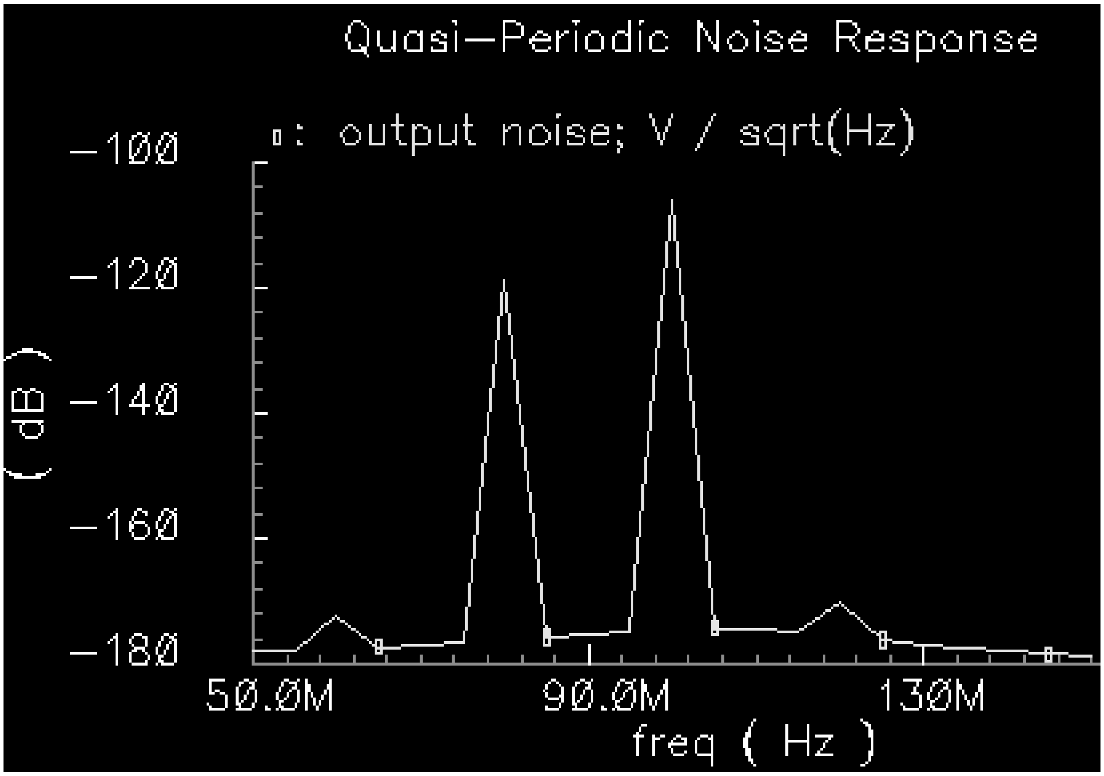
6. Follow the prompt at the bottom of the form

   `Press plot button on this form...`

   Click on *Plot* in the Direct Plot form.

   The plot appears in the Waveform window.

The Waveform window appears like the one below:



7. To determine the output noise at different frequencies, move the cursor along the output noise curve in the Waveform window. In the above plot, the output noise is about `-119.2 dB` at `80 MHz`.

Note: Before you run most other simulation examples in this chapter, be sure to reset the *Design Variable* `frf` to `920M`.

5

Simulating Oscillators

Autonomous PSS Analysis

Periodic steady state (PSS) analysis lets you simulate both driven and autonomous circuits. Driven circuits; such as amplifiers, filters, or mixers, require a time-varying stimulus to create a time-varying response. Autonomous circuits, such as oscillators, however, have time-varying responses and generate non-constant waveforms even though the circuits themselves are time-invariant.

With driven circuits, you specify an analysis period that is an integer multiple of the period of the drive signals. You cannot specify the analysis period for autonomous circuits because you do not know the precise oscillation period in advance. Instead, you specify an estimate of the oscillation period. The PSS analysis uses this estimate to compute the precise period and the periodic solution waveforms.

To enable PSS analysis to compute the precise period, you specify a pair of nodes. PSS analysis monitors the potential difference between these nodes and uses this information to refine its estimate of the period.

Phases of Autonomous PSS Analysis

A PSS analysis has two phases,

- A transient analysis phase to initialize the circuit
- A shooting phase to compute the periodic steady state solution.

The transient analysis phase is divided into three intervals:

- A beginning interval that starts at `tstart`, which is normally 0, and continues through the onset of periodicity for the independent sources
- A second, optional stabilization interval of length `tstab`

- A final interval that is four times the estimated oscillation period specified in the PSS Analysis form. During the final interval, the PSS analysis monitors the waveforms in the circuit and improves the estimate of the oscillation period.

During the shooting phase, the circuit is simulated repeatedly over one period. The length of the period and the initial conditions are modified to find the periodic steady state solution.

Phase Noise and Oscillators

Oscillators tend to amplify any noise present near the oscillation frequency. The closer the noise frequency to the oscillation frequency, the greater the amplification. Noise amplified by the oscillator in this manner is called *phase noise*. Phase noise is the most significant source of noise in oscillators, and because phase noise is centered about the oscillation frequency, filtering can never completely remove the it.

You can understand phase noise if you recognize that the phase of an oscillator is arbitrary because there is no drive signal to lock to. Any waveform that is a solution to an oscillator can be shifted in time and still be a solution. If a perturbation disturbs the phase, nothing restores the phase, so it drifts without bound. If the perturbation is random noise, the drift is a random walk. Furthermore, the closer the perturbation frequency is to the oscillation frequency, the better it couples to the phase and the greater the drift. The perturbation need not come from random noise. Noise might also couple into the oscillator from other sources, such as the power supplies.

The first two examples in this chapter both examine phase noise.

Starting and Stabilizing the Oscillator

To simulate an oscillator using PSS analysis, you must first start it. You can start an oscillator by supplying either:

- A brief impulse stimulus

  The stimulus should couple strongly into the oscillatory mode of the circuit and poorly into other long-lasting modes such as bias circuitry.

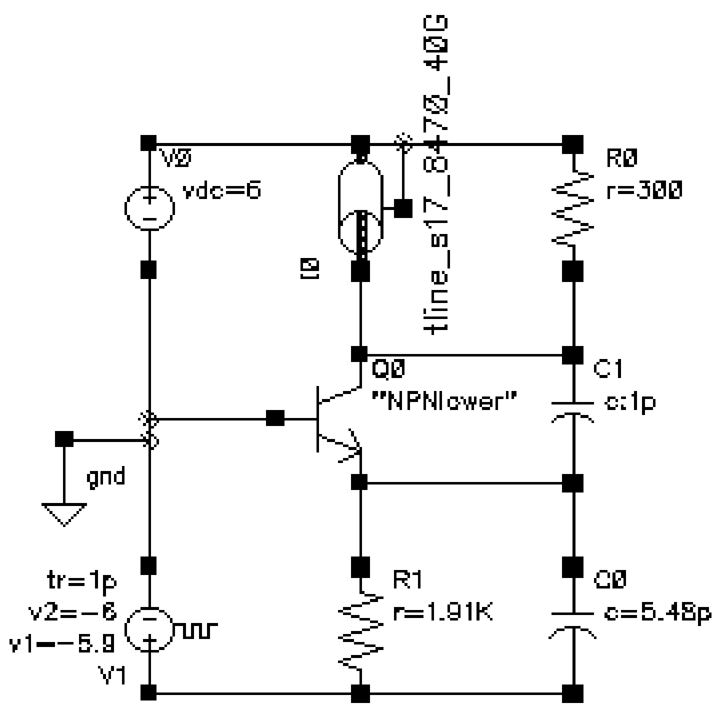
- A set of initial conditions for the components of the oscillator’s resonator

Regardless of which technique you use to start the oscillator, allow the oscillator to run long enough to stabilize before you start the shooting phase and compute the steady state solution. Adjust the *tstab* parameter to supply the additional stabilization period.

The tline3oscRF Oscillator Circuit

This example computes the periodic steady state solution and the phase noise for the *tline3oscRF* circuit shown in Figure 5-1.

Figure 5-1 Schematic for the tline3oscRF Oscillator Circuit



Phase noise computation is important because phase noise is the most significant source of noise in oscillators. Because phase noise is centered about the oscillation frequency, the noise can never be completely removed by filtering.

This circuit supplies the brief impulsive stimulus needed to start the oscillator, so you do not need to set initial conditions. You supply the data necessary for PSS analysis to estimate the period when you specify the fundamental frequency.

Simulating tline3oscRF Oscillator Circuit

Before you begin, perform the setup procedures described in Chapter 3, “Setting Up for the Examples.”

Opening the tline3oscRF Circuit in the Schematic Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the schematic.

2. Choose *my_rfExamples* for *Library Name*. Choose the editable copy of the *rfExamples* library you created. See Chapter 3, “Setting Up for the Examples” for more information.

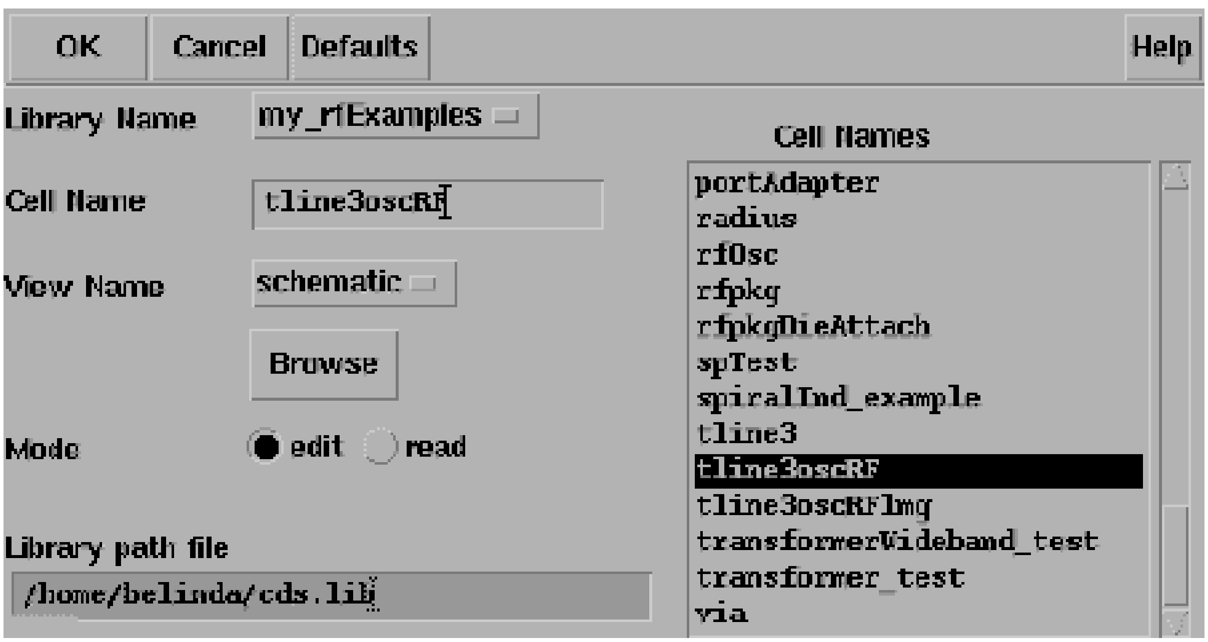
3. Choose *schematic* for *View Name*.

4. In the *Cell Names* list box, highlight *tline3oscRF*.

   tline3oscRF appears in the *Cell Name* field.

5. Highlight *edit* to choose the *Mode*.

   The completed Open File form appears like the one below.

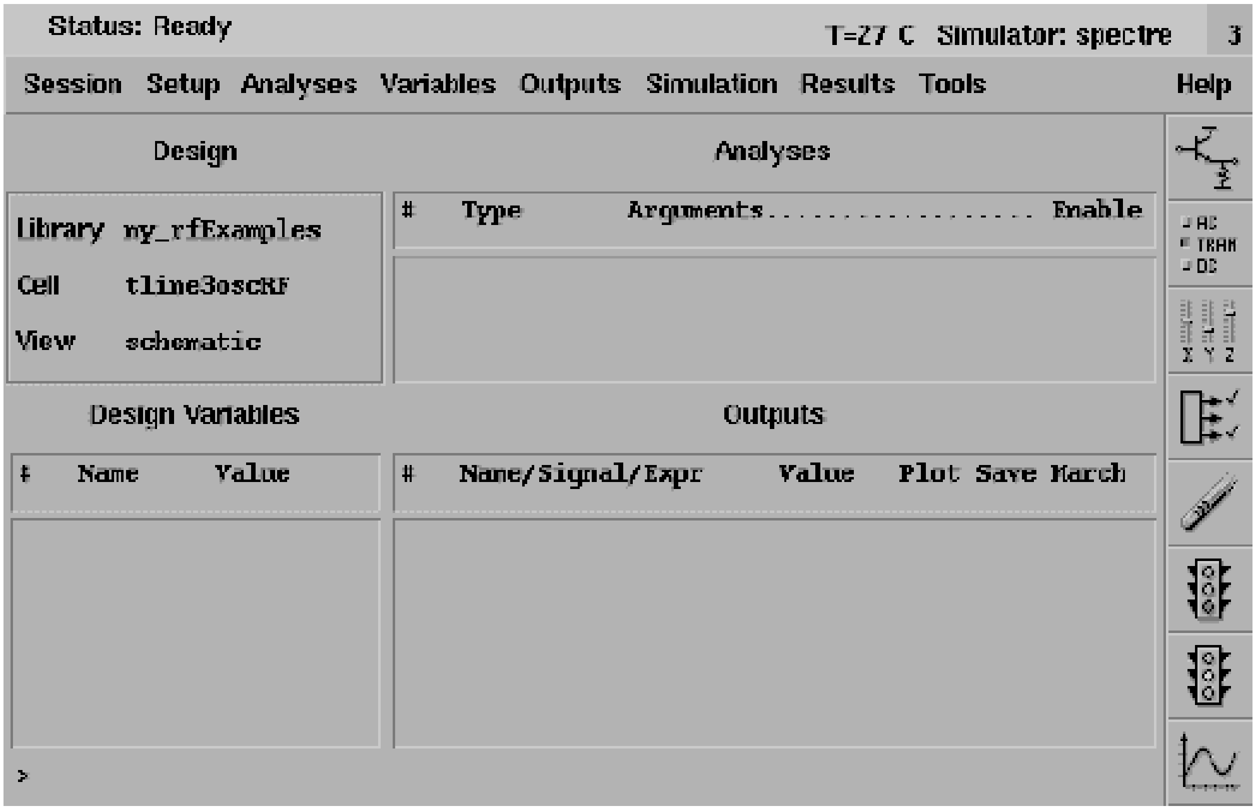


6. Click *OK*.

   The Schematic window for the *tline3oscRF* oscillator opens.

7. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window opens.



You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing the *tline_3oscRF* in the Choosing Design form.

Choosing Simulator Options

1. In the Simulation window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, do the following:

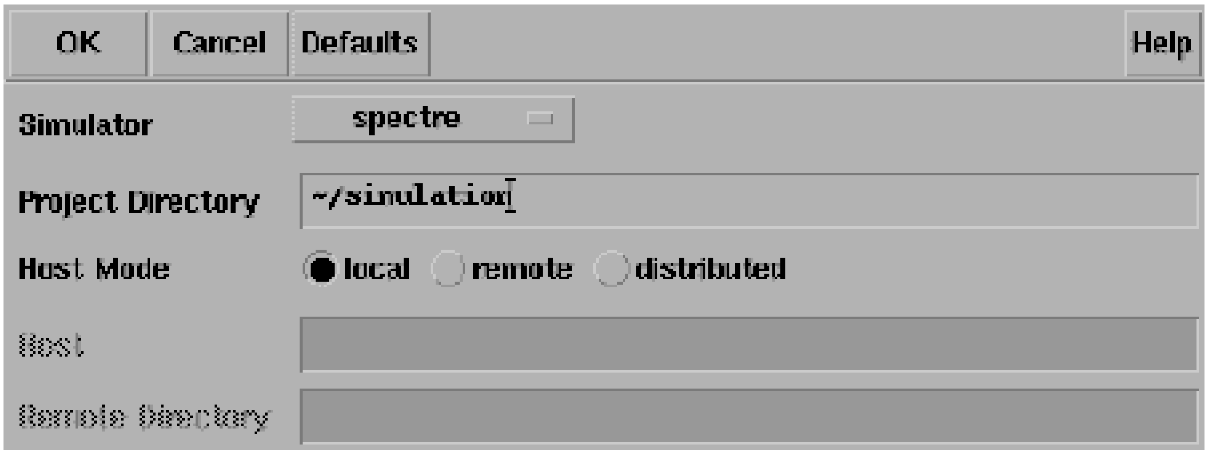
   a. Choose *spectre* for *Simulator*.

   b. Type in the name of the project directory, if necessary.

   c. Highlight the *local* or the *remote* button to specify the *Host Mode*.

For remote simulation, type the name of the host machine and the remote directory in the appropriate fields.

The completed form looks like this.



3. In the Choosing Simulator/Directory/Host form, click *OK*.

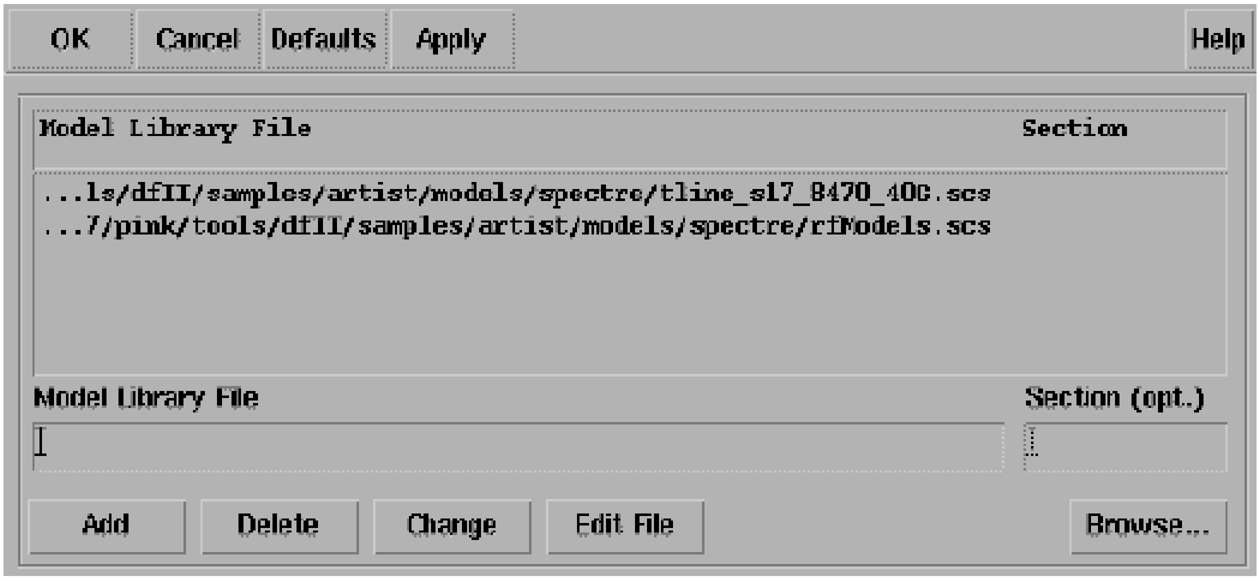
Setting Up Model Libraries

In the Simulation window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

1. In the *Model Library File* field, type the full path to the `rfModels.scs` model file including the file name, `rfModels.scs`.
2. Click on *Add*.
3. In the *Model Library File* field, type the full path to the `tline_s17_8470_40G.scs` model file including the file name, `tline_s17_8470_40G.scs` and click on *Add*.

The completed form looks like this.



4. In the Model Library Setup form, click *OK*.

Periodic Steady State and Phase Noise with PSS and Pnoise

This example computes the periodic steady state solution and phase noise for the *tline3oscRF* oscillator circuit. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

Setting Up the Simulation

1. If necessary, open the `tline3oscRF` circuit.
2. If necessary, specify the full path to the model files.
3. If necessary, in the Simulation window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the Simulation window to verify whether or not any analyses are enabled.)

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a transient analysis before you run this PSS analysis. From the transient analysis you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis.

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click on *pss*.

   The form changes to display options needed for PSS analysis.

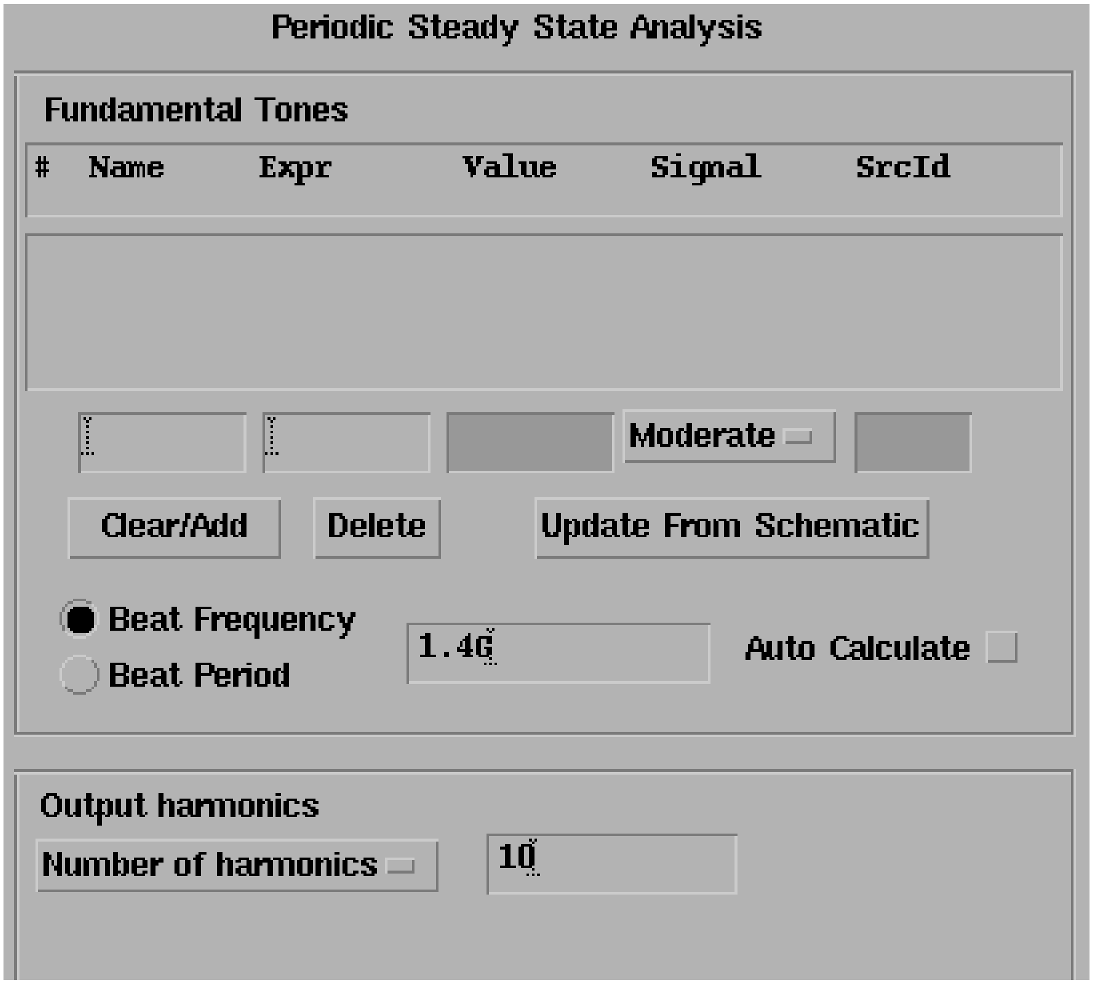
3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

   In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   Type `1.4G` in this example.

4. In the *Output harmonics* cyclic field, choose Number of harmonics and type in a reasonable number, such as `10`.

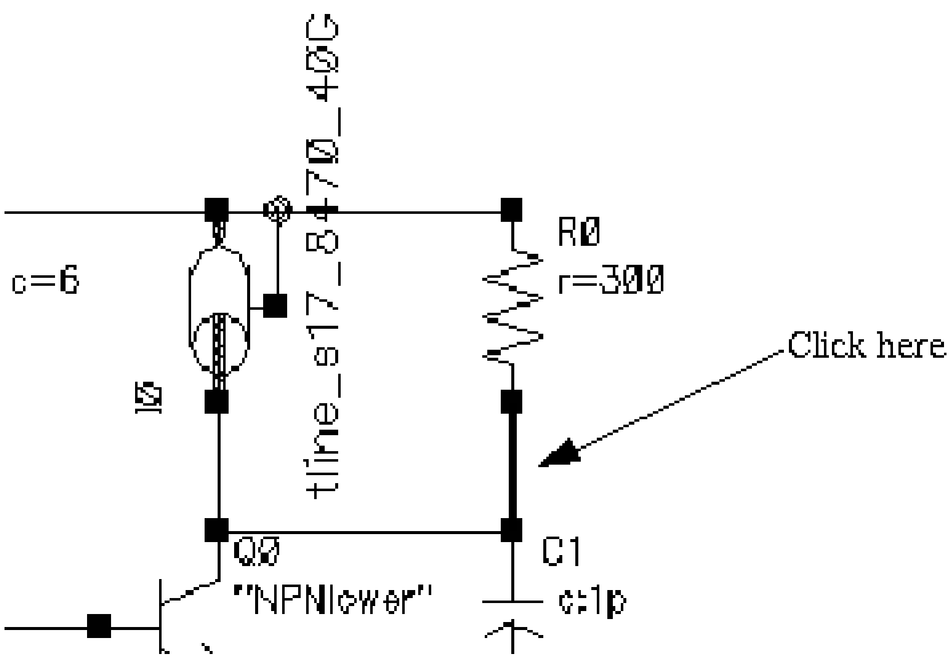
The top of the PSS Choosing Analysis form appears below.



5. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
6. Highlight *Oscillator*.

   The form changes to let you specify the two nodes needed for oscillator simulation.

7. Click on the *Oscillator node Select* button. Then click on the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose *net5*.



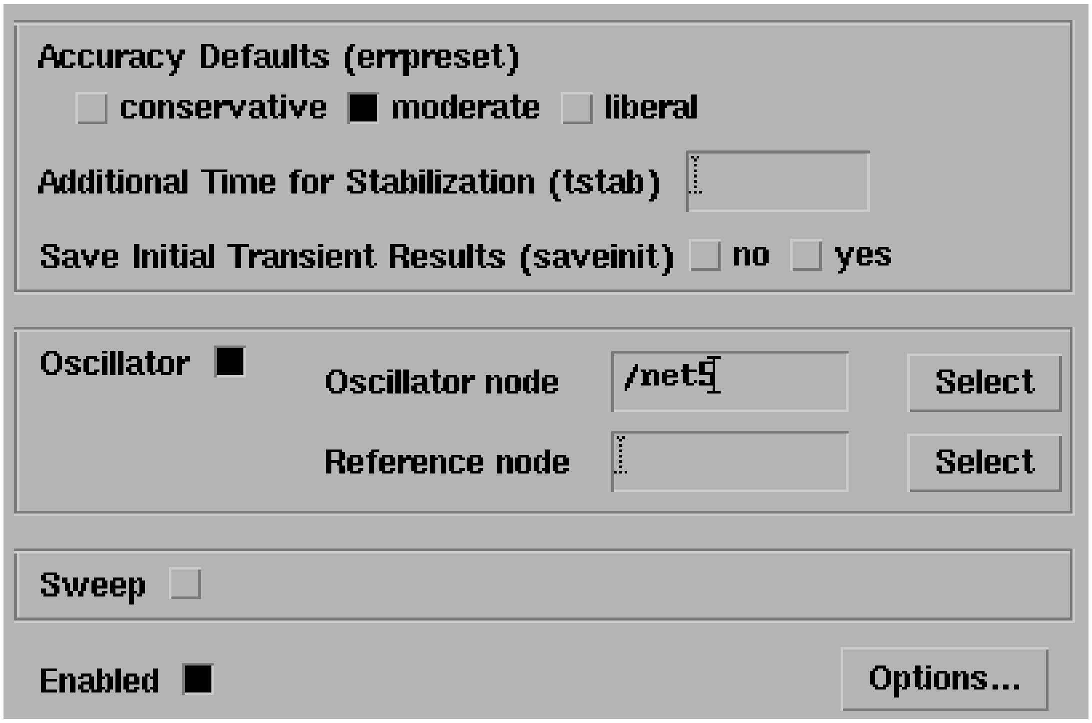
`/net5` appears in the *Oscillator node* field.

8. Leave the *Reference node* field empty.

   The *Reference node* field defaults to `/gnd!`. You can set *Reference node* to a different value by clicking on the *Reference node Select* button and then choosing the appropriate node in the schematic.

9. Verify that *Enabled* is highlighted.

The bottom of the PSS Choosing Analysis form appears below.



Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click on *pnoise*.

   The form changes to let you specify data for the Pnoise analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter `1` in the *Relative Harmonic* field.

   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify `2K`, you choose a value `2K` away from the fundamental frequency.

3. Choose *Start-Stop* for the type of *Frequency Sweep Range (Hz)*. Type in reasonable values, such as `1K` and `100M` as the *Start* and *Stop* values.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a log sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Number of Steps* and type `201` in the field.

   Note: Be sure to use a nonzero starting value for logarithmic sweeps.

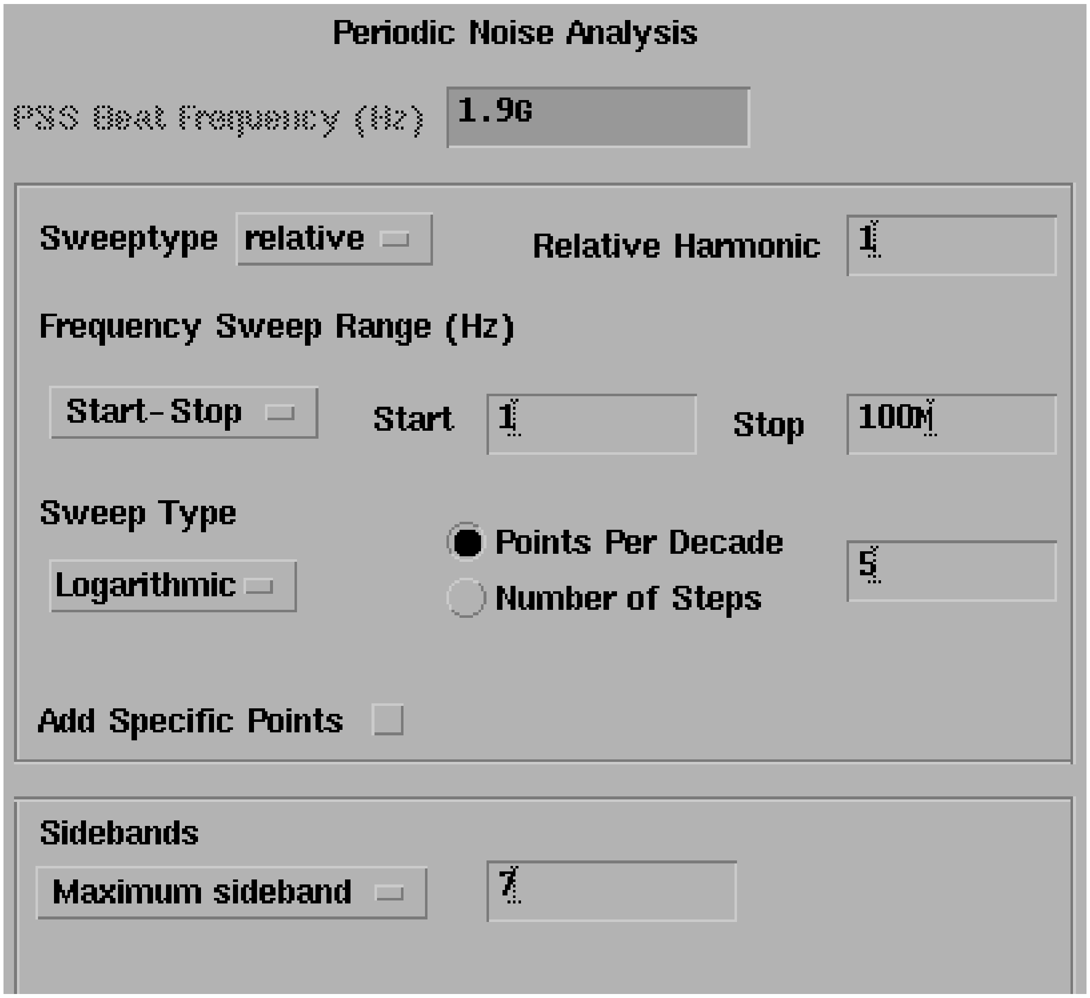
6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type 7 in the *Maximum sideband* field. The default parameter value is `maxsideband=7`.

   Begin with a *Maximum sideband* value of 7. Then, in subsequent simulations, increase the value to see whether the output noise changes. Continue to increase the *Maximum sideband* value until the output noise stops changing.

   For a fundamental oscillator, *Maximum sideband* must be at least `1` to see any flicker noise up conversion. In general, small values for *Maximum sideband* are not recommended.

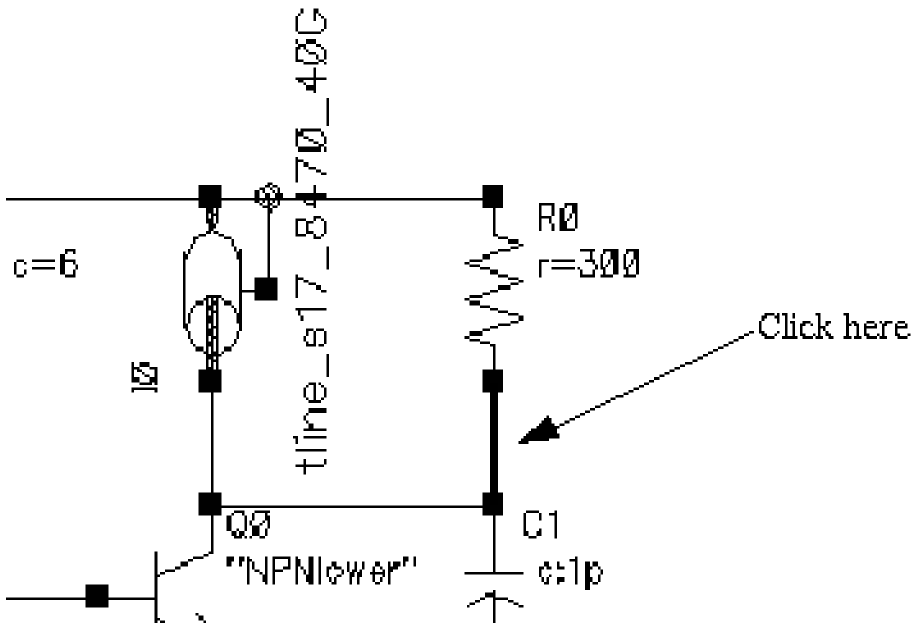
   Note: Be sure to use a nonzero *Maximum sideband* value.

   The top of the Pnoise Choosing Analysis form appears below.



7. In the *Output* cyclic field, choose *voltage*.

8. Click on the *Positive Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose net 5.



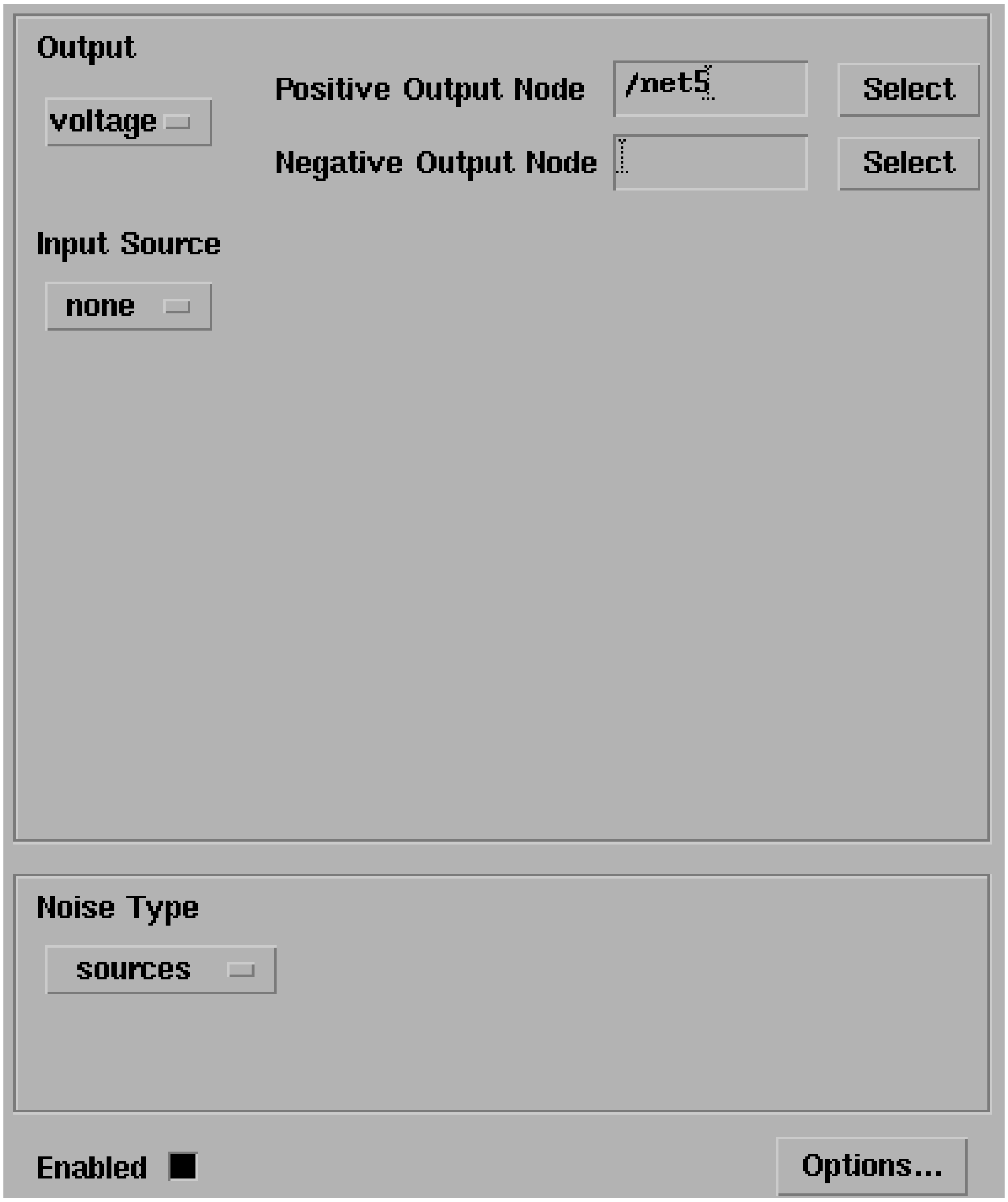
`/net5` appears in the *Positive Output Node* field.

9. Leave the *Negative Output Node* field empty.

This field defaults to `/gnd!`. You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then choosing the output node in the schematic.

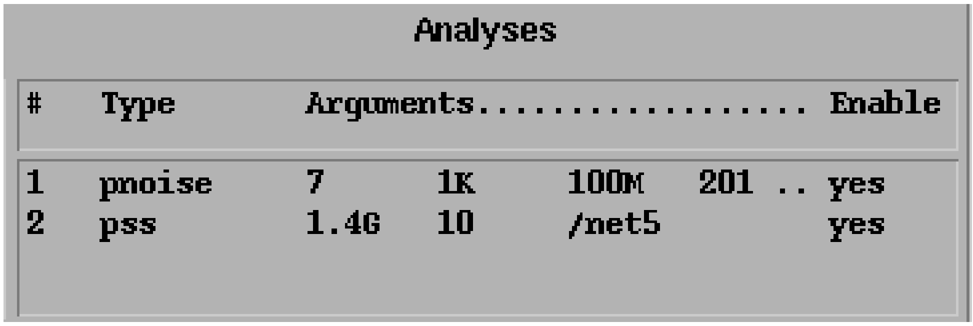
10. In the *Input Source* cyclic field, choose *none*.

The bottom of the Pnoise Choosing Analysis form appears below.



11. Verify that the Pnoise analysis is Enabled. Then click *OK* in the Choosing Analyses form.

The PSS and `Pnoise` options you choose appear in the *Analyses* list box in the Simulation window.



Analyses

#	Type	Arguments..................				Enable
1	pnoise	7	1K	100M	201 ..	yes
2	pss	1.4G	10	/net5		yes

Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Fundamental Frequency

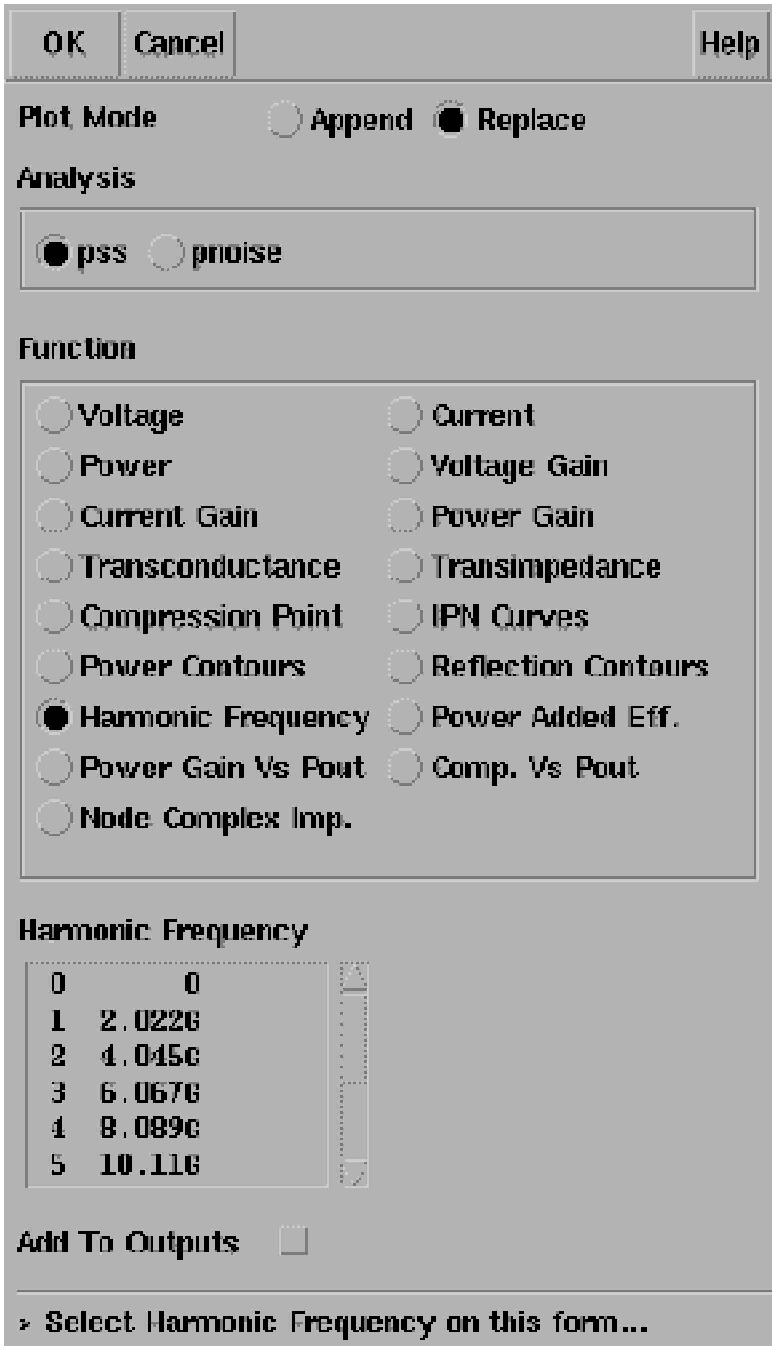
➤ To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

The Waveform window and the Direct Plot form both appear.

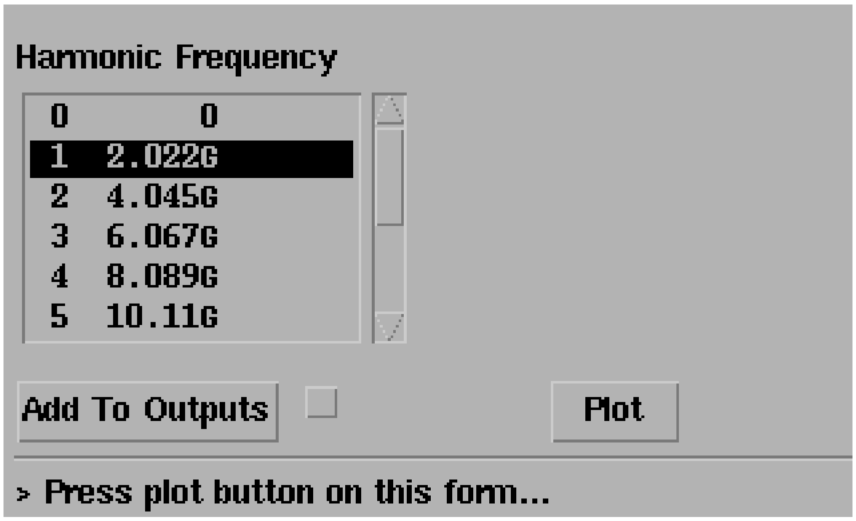
To plot the fundamental frequency, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pss* for *Analysis.*
3. Highlight *Harmonic Frequency* for *Function*.

The form changes to display the *Harmonic Frequency* list box.

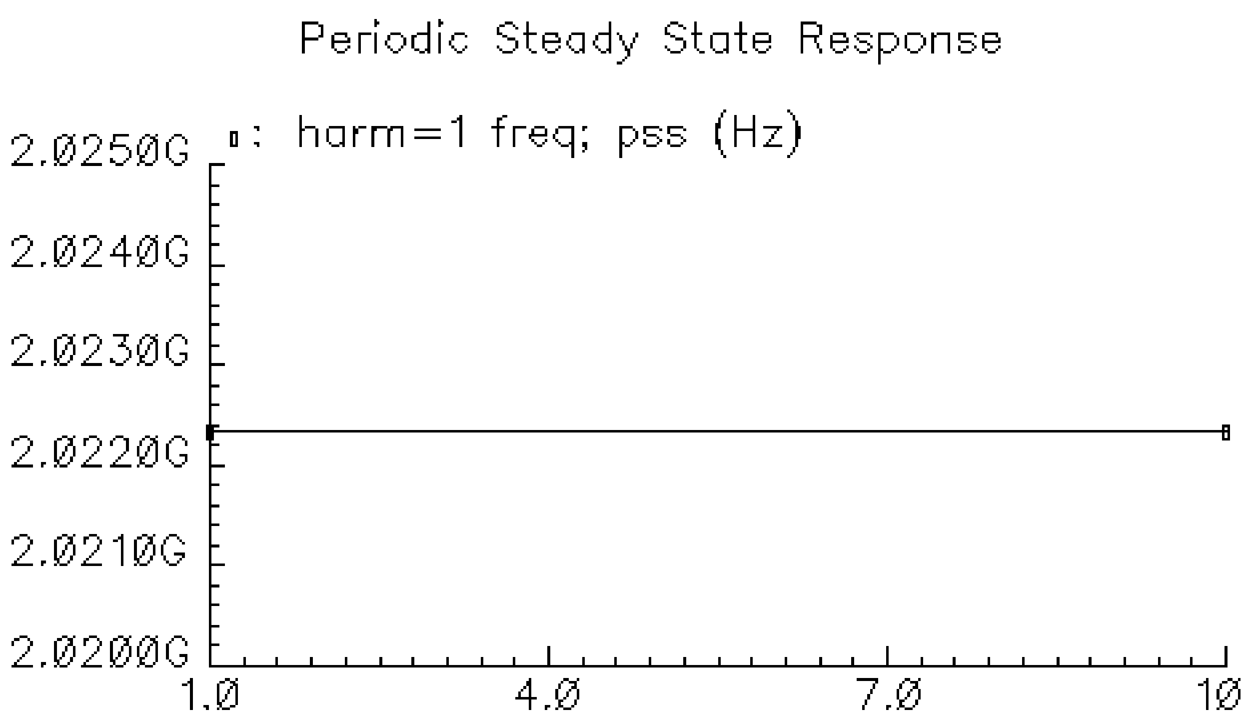


4. Follow the prompt at the bottom of the form—highlight the first harmonic in the *Harmonic Frequency* list box.



5. Follow the prompt at the bottom of the form—click *Plot*.

   The Waveform window displays the fundamental frequency.



Notice that the fundamental frequency value of `2.022G` is also displayed in the *Harmonic Frequency* list box in the Direct Plot form.

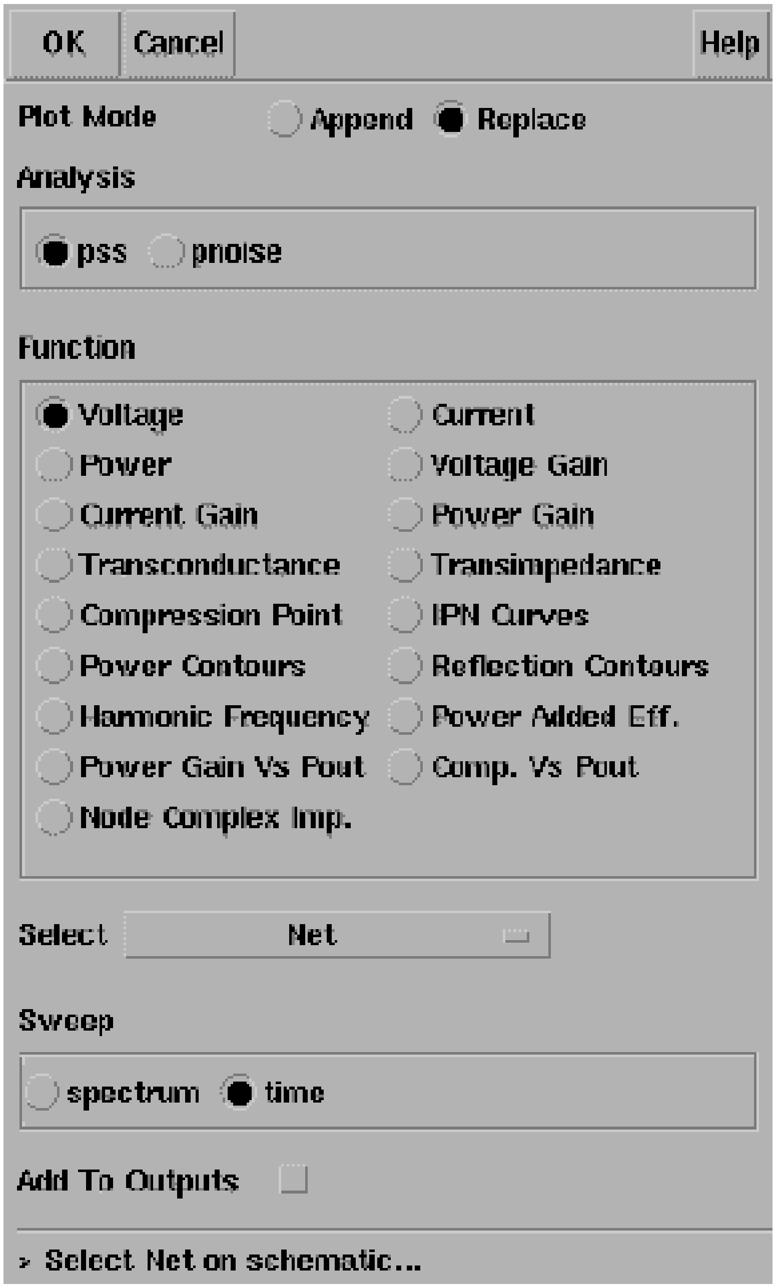
Plotting the Periodic Steady State Solution

If necessary (to display the Direct Plot form), in the Simulation window choose *Results – Direct Plot – Main Form* to open the Waveform window and the Direct Plot form.

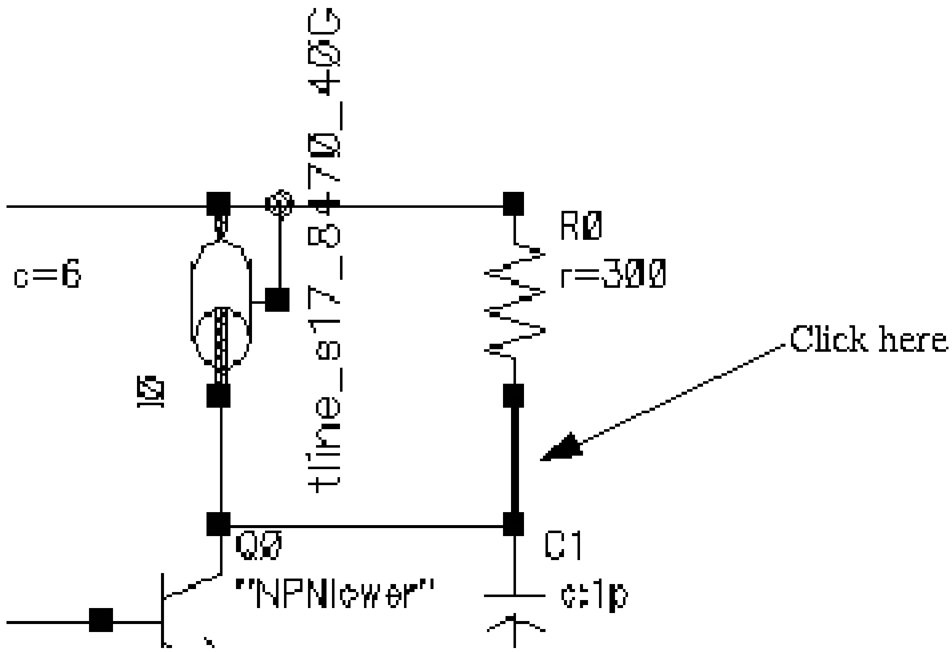
To plot the periodic steady state solution waveform for distortion, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. If necessary, highlight *pss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Highlight *Time* for *Sweep*.

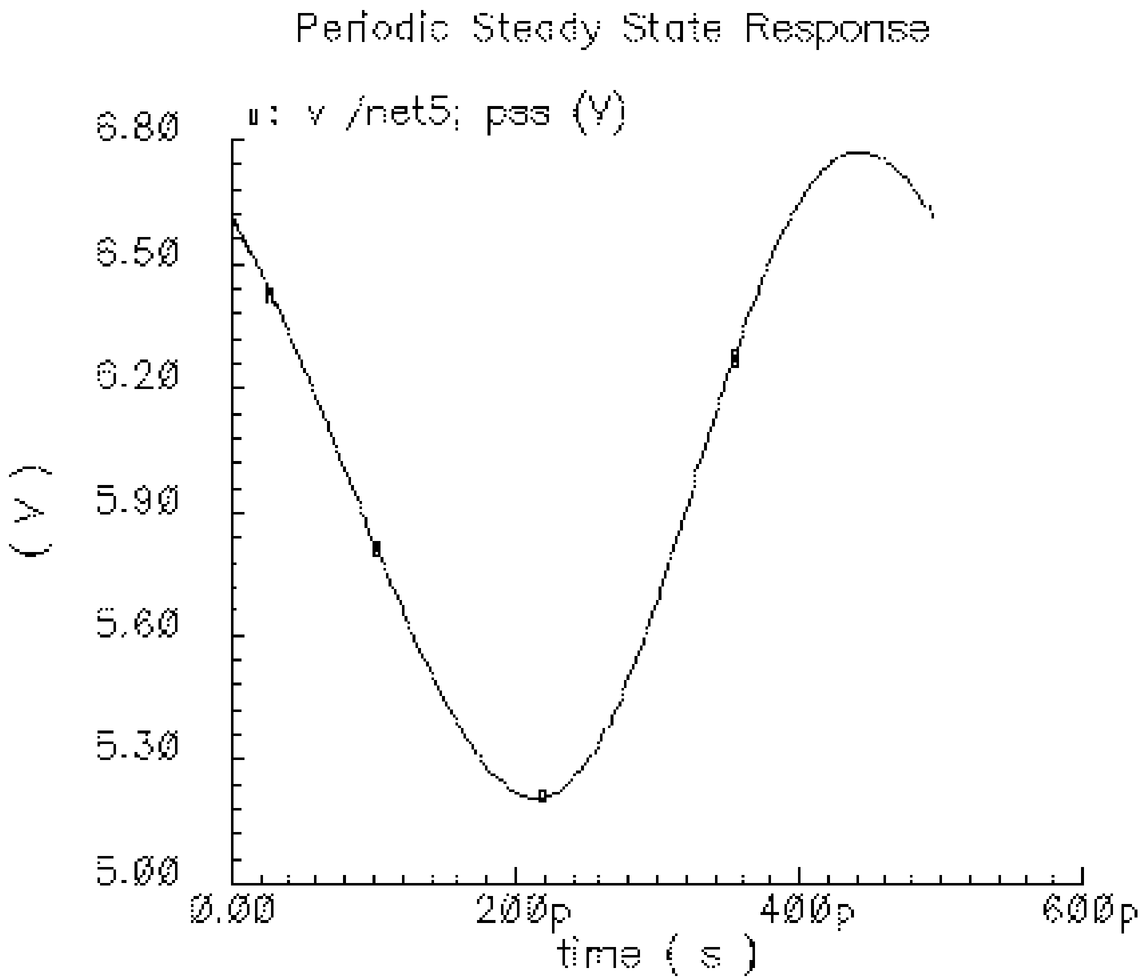
5. Notice that the default choice in the *Select* cyclic field is *Net*. Also notice the *Select net on schematic* prompt at the bottom of the form.



6. In the Schematic window, click on the appropriate wire to choose net 5.



The Waveform window displays the sweep of voltage versus time—the periodic steady state solution.



Plotting the Phase Noise

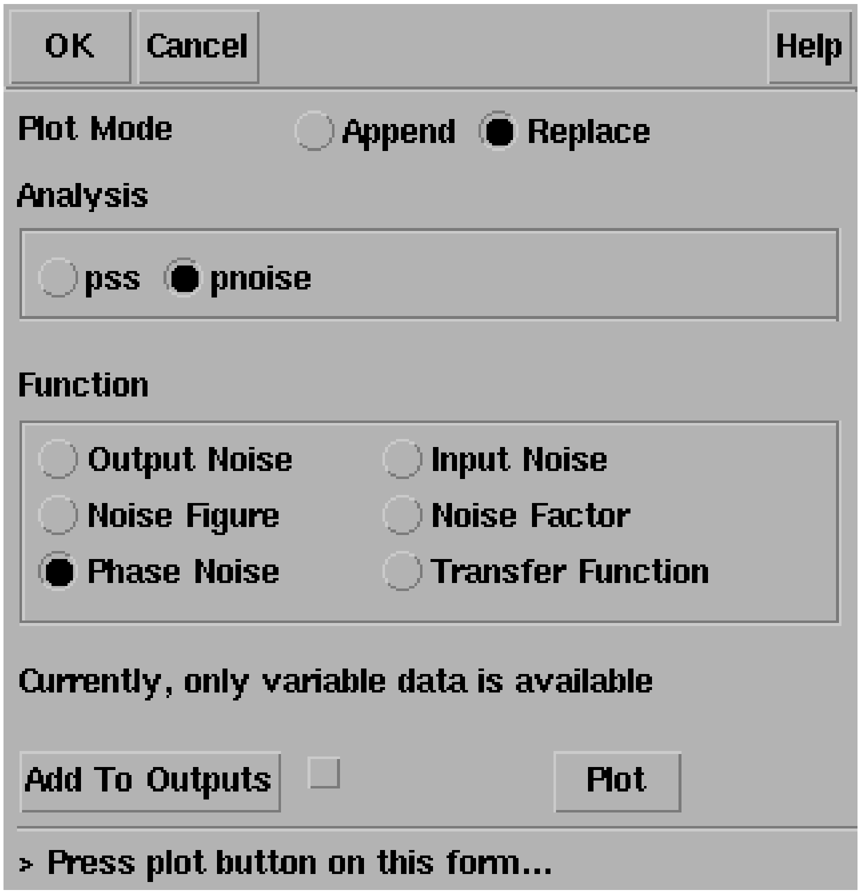
If necessary (to display the Direct Plot form), in the Simulation window choose *Results – Direct Plot – Main Form* to open the Waveform window and the Direct Plot form.

To plot the phase noise waveform, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise* for *Analysis*.

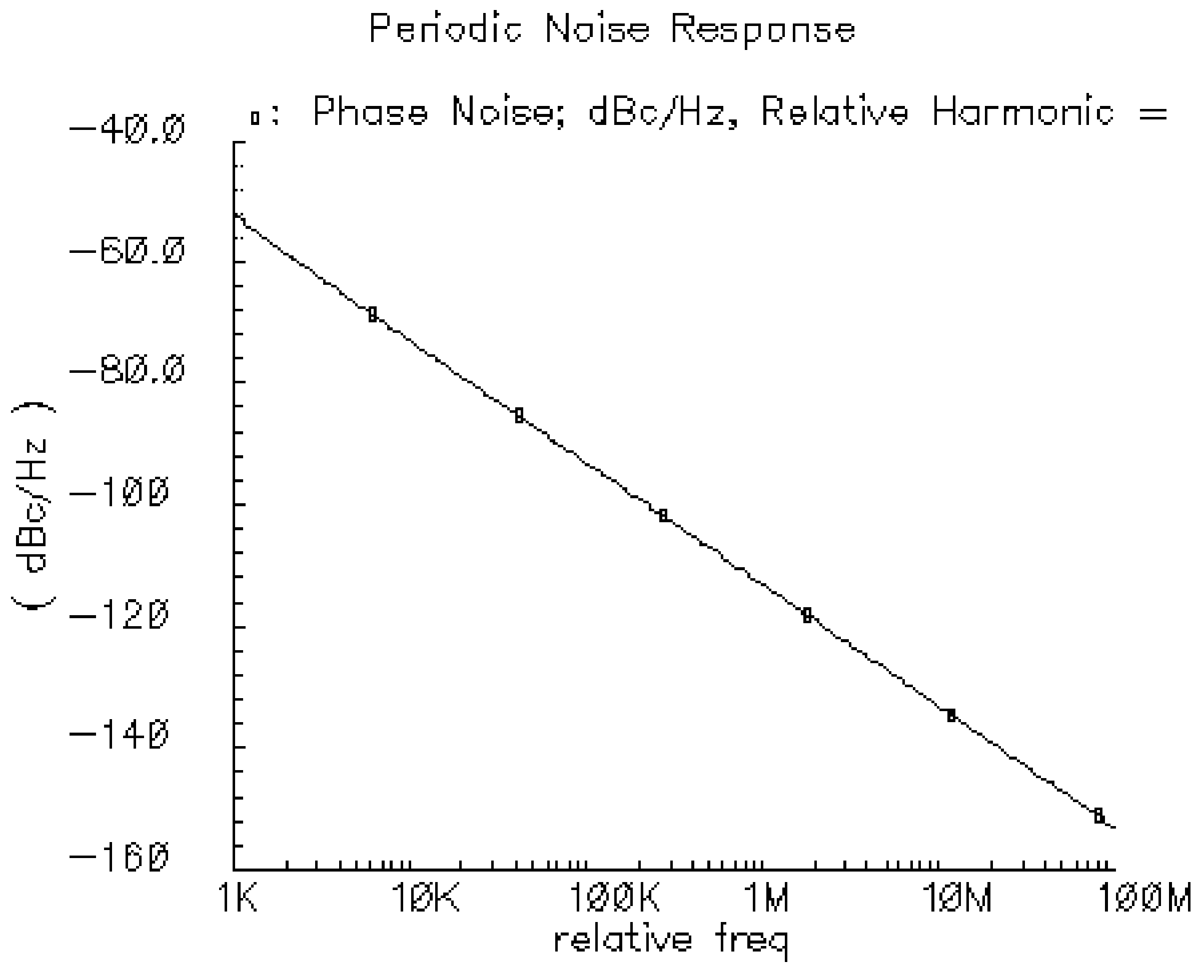
   The form changes to display fields relevant for the `Pnoise` analysis.

3. Highlight *Phase Noise* for *Function*.



4. Click on *Plot*.

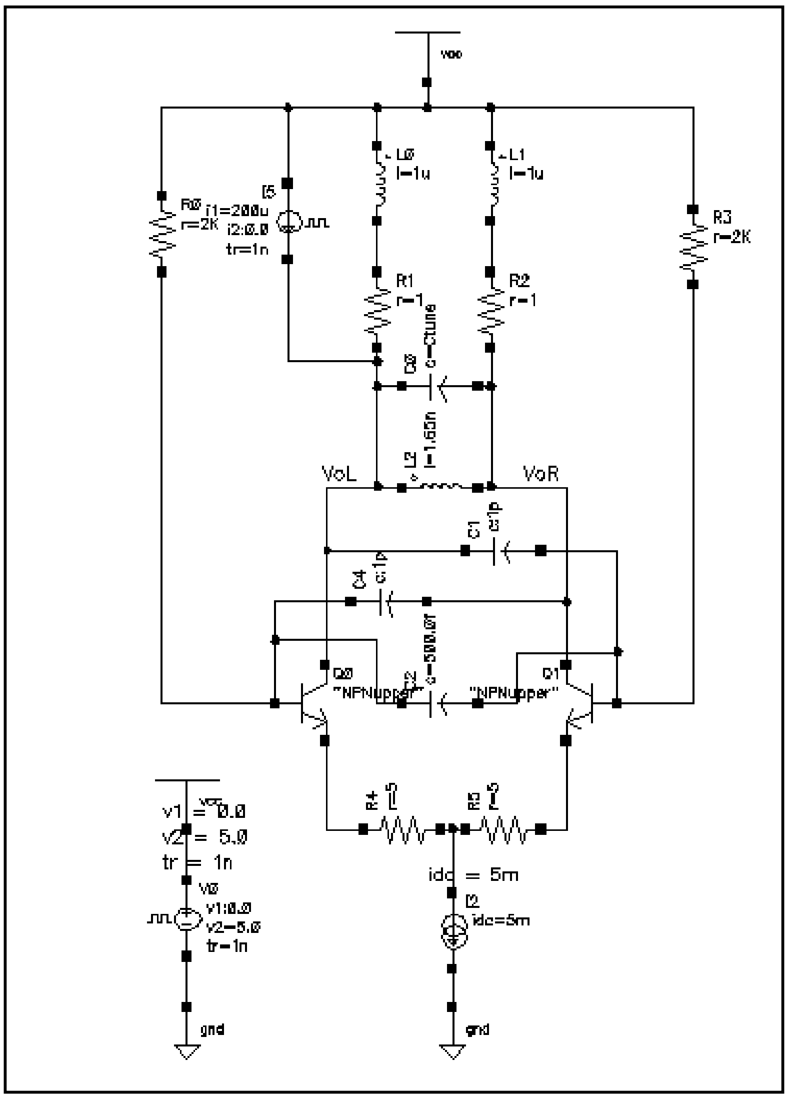
The Waveform window displays the phase noise versus frequency with the frequency in logarithmic scale. When you plot phase noise, you get more useful information if you use the log scale.



The oscDiff Circuit: A Balanced, Tunable Differential Oscillator

This example computes the fundamental frequency, output noise and phase noise for the *oscDiff* circuit, the balanced, tunable differential oscillator shown in Figure 5-2.

Figure 5-2 Schematic for the oscDiff Oscillator Circuit



The *oscDiff* circuit generates sinusoids in antiphase for differential nodes *VoL* and *VoR*. The oscillation frequency is primarily governed by inductor *L0* and capacitor *C3*. The *C3* capacitance value is voltage-dependent and obeys Equation 5-1, the standard Schottky barrier capacitance equation.

(5-1)
$$Ctune = \frac{Cj0}{\left(1 + \frac{Vcntrl}{phi}\right)^{gamma}}$$

where

Ctune	*C3* capacitance value
Cj0	Zero-bias junction capacitance
Vcntrl	Applied varactor voltage
phi	Barrier potential
gamma	Junction grading coefficient

Simulating the oscDiff Circuit

Before you begin, be sure that you have performed the setup procedures described in Chapter 3, “Setting Up for the Examples.”

Opening the oscDiff Circuit in the Simulation Window

1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the schematic.

2. Choose *rfExamples* for *Library Name*. Choose the editable copy of the *rfExamples* library you created. See Chapter 3, “Setting Up for the Examples” for more information.

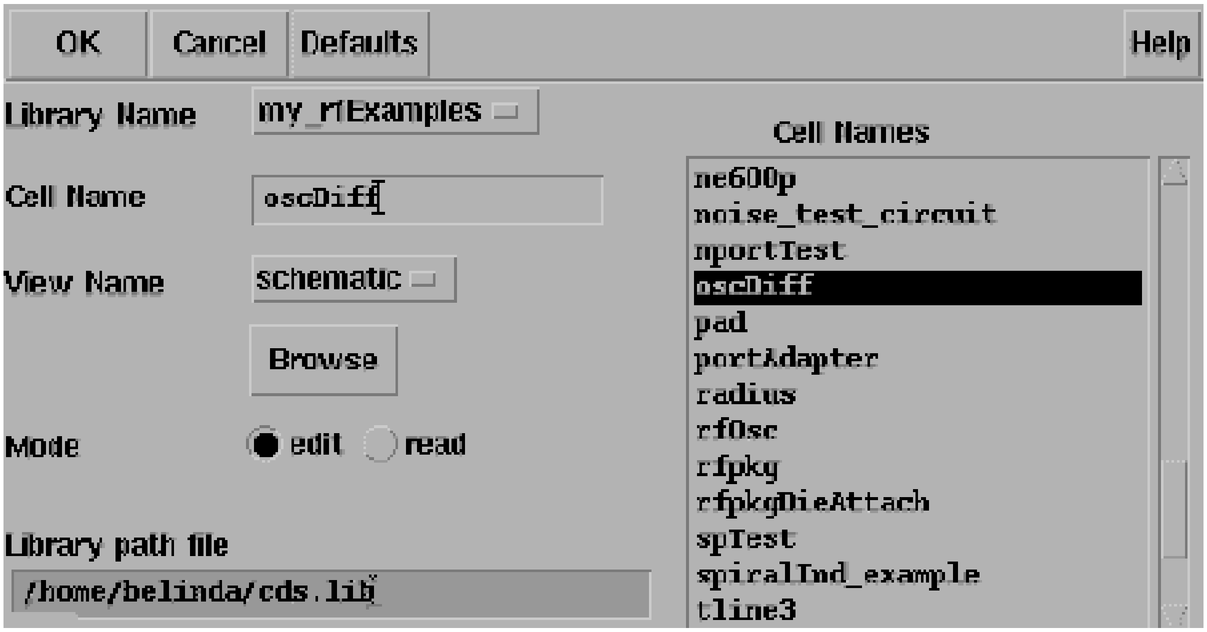
3. Choose *schematic* for *View Name*.

4. In the *CellNames* list box, highlight *oscDiff*.

   oscDiff appears in the Cell Name field.

5. Highlight *edit* for *Mode*.

The completed Open File form appears like the one below.

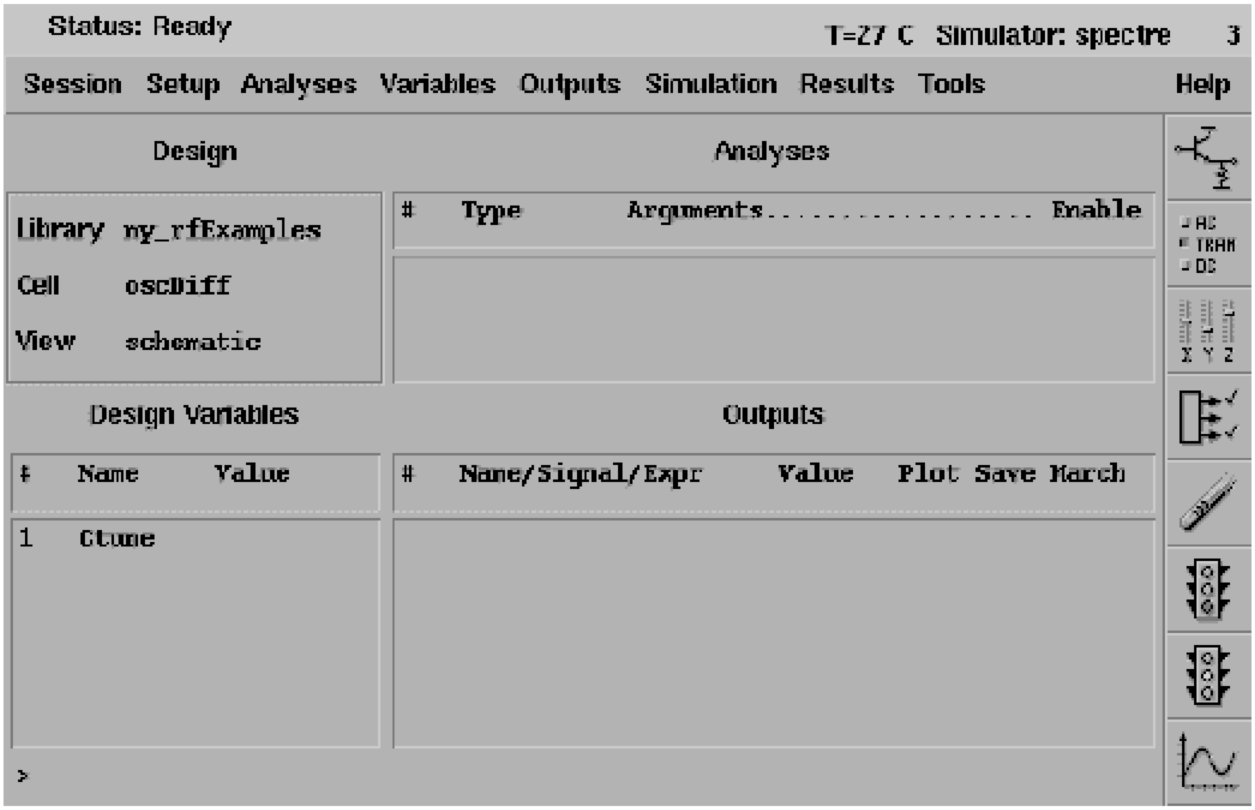


6. Click *OK*.

The Schematic window for the oscDiff oscillator opens as shown in Figure 5-2 on page 356.

7. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window appears.



You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing the *oscDiff* in the Choosing Design form.

Choosing Simulator Options

1. In the Simulation window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. In the Choosing Simulator/Directory/Host form, do the following:

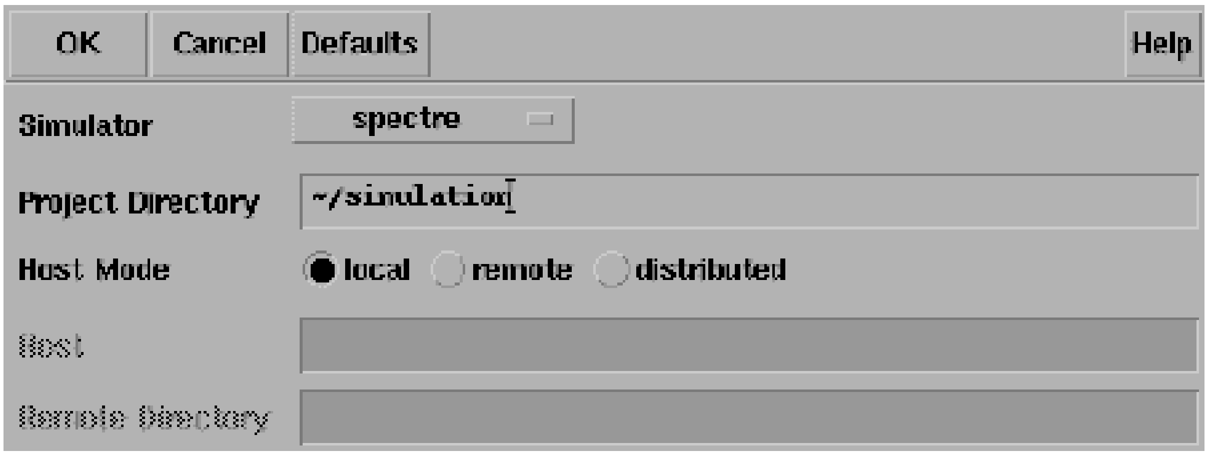
   a. Choose *spectre* for *Simulator*.

   b. Type in the name of the project directory, if necessary.

   c. Highlight the *local* or the *remote* button to specify the *Host Mode*.

For remote simulation, type in the name of the host machine and the remote directory in the appropriate fields.

The completed form appears like the one below.



3. In the Choosing Simulator/Directory/Host form, click *OK*.

Setting Up Model Libraries

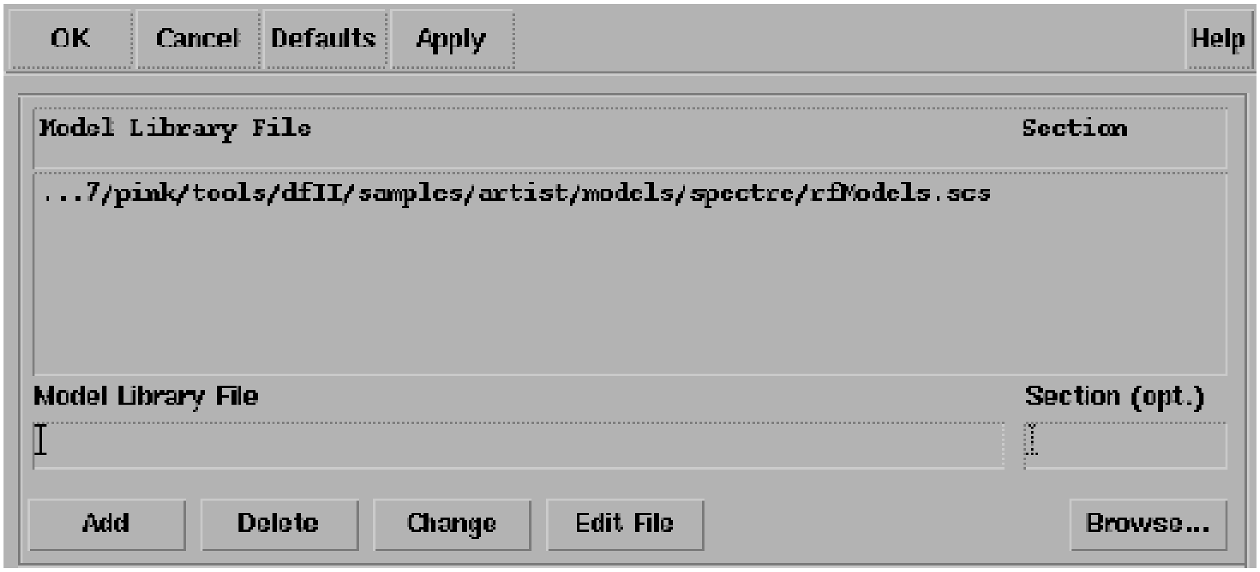
1. In the Simulation window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

2. In the Model Library File field, type the full path to the model file including the file name, *rfModels.scs*.

3. Click *Add*.

The completed form looks like this.



4. Click *OK*.

Fundamental Frequency, Output Noise, and Phase Noise with PSS and Pnoise

This example computes the fundamental frequency, output noise and phase noise for the oscDiff differential oscillator circuit. You perform a PSS analysis first because the periodic steady-state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the fundamental frequency, output noise and phase noise.

Setting Up the Simulation

1. If necessary, open the *oscDiff* circuit.
2. If necessary, specify the full path to the model files.
3. If necessary, in the Simulation window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the Simulation window to verify whether or not any analyses are enabled.)

Editing Design Variables

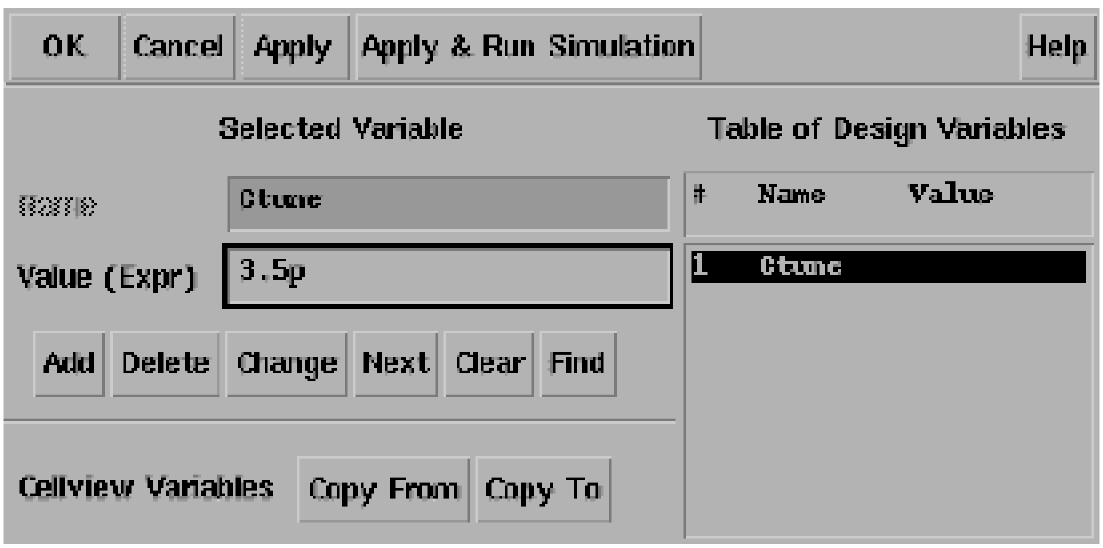
1. In the Simulation window, choose *Variables – Edit*.

   The Editing Design Variables form appears. For this example, you use this form to set the value of the variable *Ctune*.

In the Editing Design Variables form, do the following:

1. In the Table of Design Variables list box, highlight Ctune.
2. Type 3.5p in the Value (Expr) field.
3. Click on Change.

   The completed form looks like this.



4. Click *OK*.

   Ctune appears in the *Design Variables* section of the Simulation window with its modified value

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this PSS analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis.

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the analysis.

   The form changes to display options needed for the PSS analysis. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

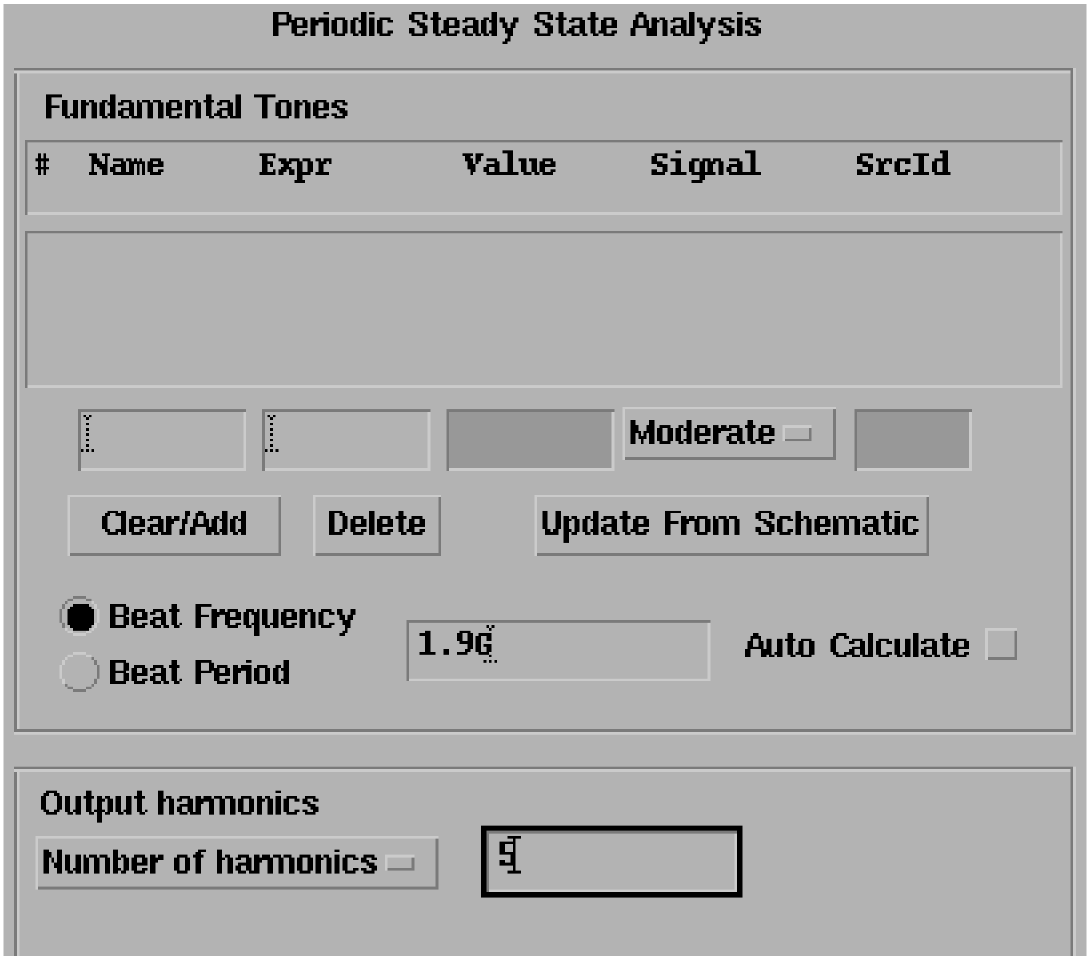
3. In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Be sure the *Auto Calculate* button is *not* highlighted.

   In the field next to the *Beat Frequency* and *Beat Period* buttons, type your best estimate of the oscillation frequency. (Your estimate can be the node voltages saved from the transient analysis mentioned above.)

   Type `1.9G` in this example.

4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type `5` in the field.

The top of the Pnoise Choosing Analysis form appears below.



5. Highlight *moderate* for the *Accuracy Default (errpreset)*.
6. Type `100.5n` in the *Additional Time for Stabilization (tstab)* field.

   You can use a slightly different value as long as it is above 100.
7. If you want to save the results of the initial transient analysis, set *Save Initial Transient Results (saveinit)* to *yes*.
8. Highlight *Oscillator*.

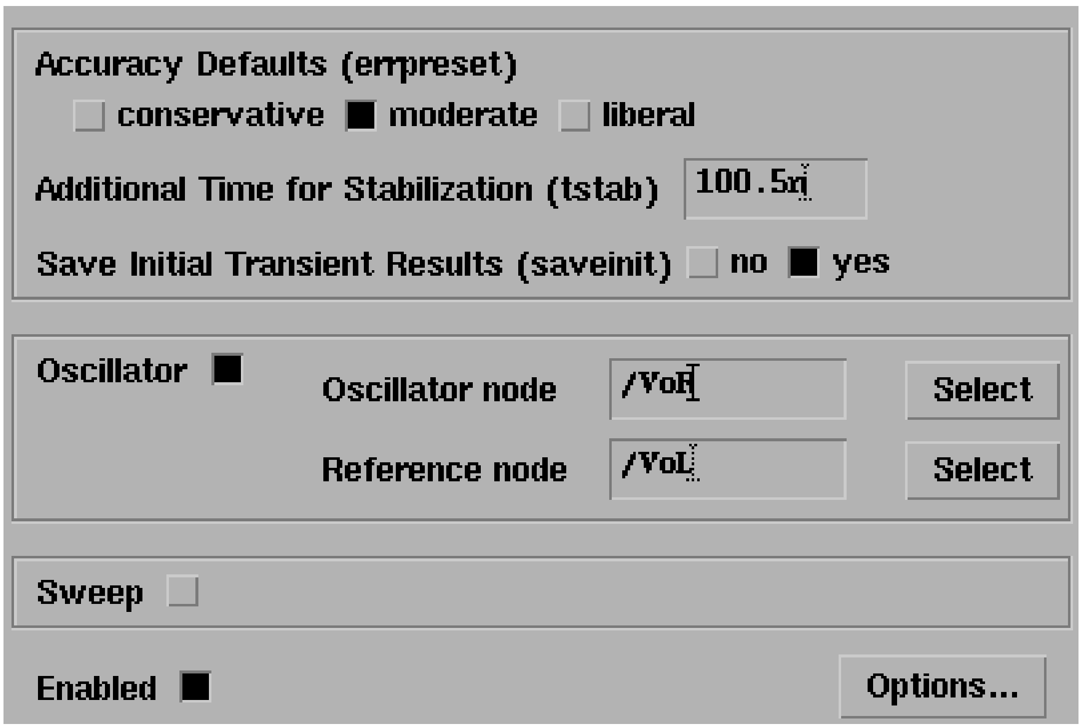
   The form changes to let you specify the two nodes needed for oscillator simulation.
9. Click on the *Oscillator node Select* button. Then click on the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled *VoR*.

`/VoR` appears in the *Oscillator node* field.

10. Click on the *Reference node Select* button. Then click on the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node labeled *VoL*.

    `/VoL` appears in the *Reference node* field. Because this is a differential oscillator, the reference node is not `/gnd!` as it was in the *tline3oscRF* example.

    The bottom of the Choosing Analyses form for PSS looks like the following.



Verify that *Enabled* is highlighted for the PSS analysis.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click on *pnoise*.

   The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter 1 in the *Relative Harmonic* field.

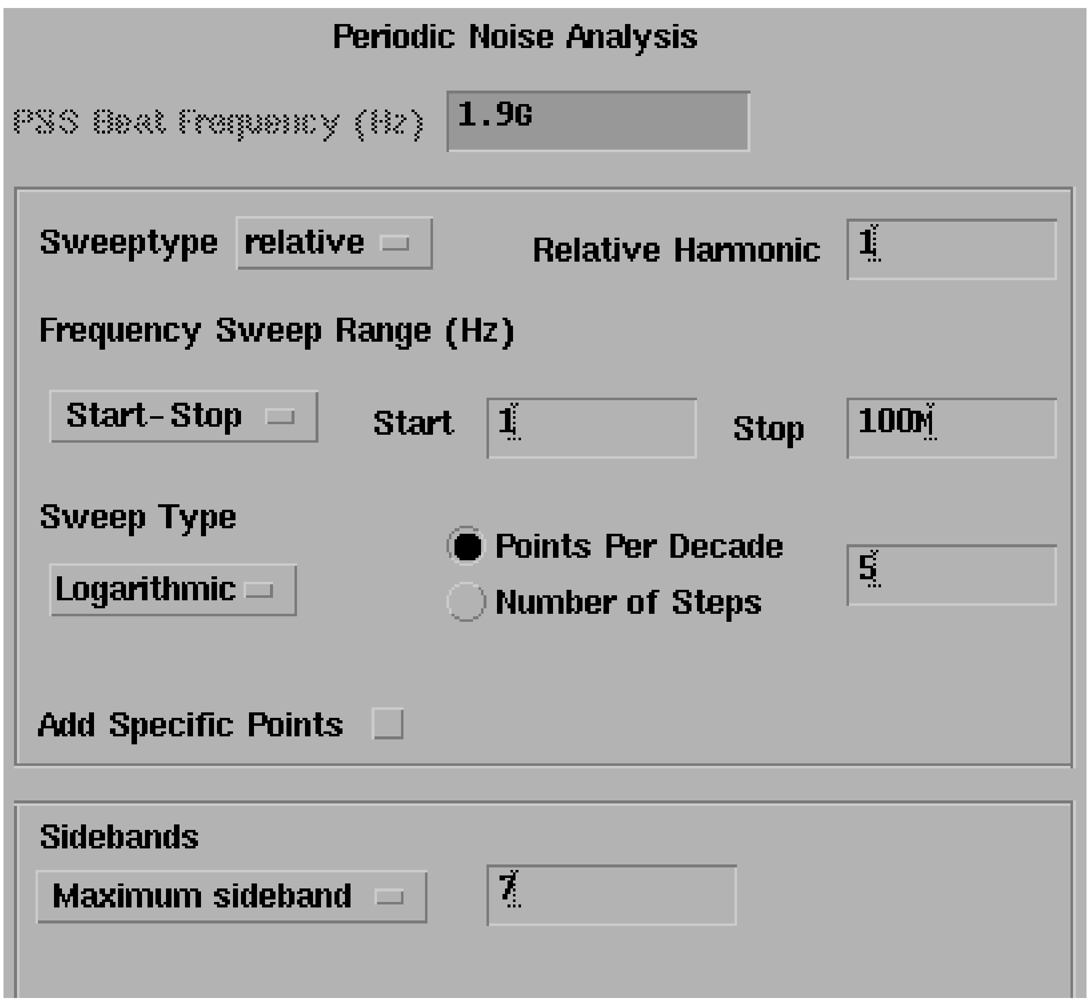
   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify 2K, you choose a value 2K away from the fundamental frequency.

3. In the *Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `1` and `100M` as the *Start* and *Stop* values, respectively.
4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a log sweep when you plot phase noise because of the size of the phase noise values.
5. Highlight *Points Per Decade* and type 5 in the field.

   Note: Be sure to use a nonzero starting value for logarithmic sweeps.
6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type 7 in the *Sidebands* field. The default value is 7.

   Note: Be sure to use a nonzero *Maximum sideband* value.

   The top of the Choosing Analyses form for the `Pnoise` analysis looks like this.



7. In the *Output* cyclic field, choose *voltage*.

8. Click on the *Positive Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose *VoR*.

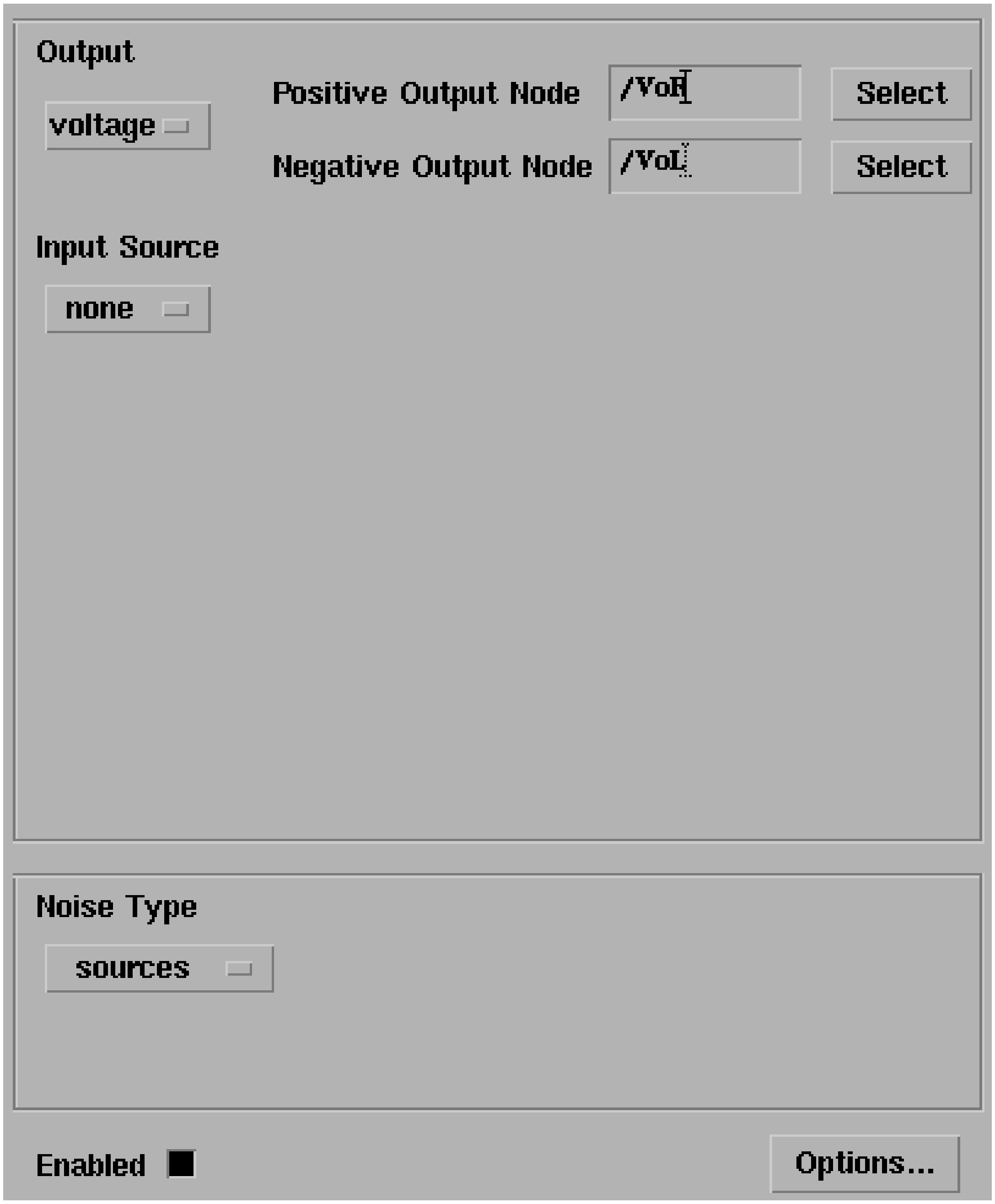
`/VoR` appears in the *Positive Output Node* field.

9. Click on the *Negative Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose *VoL*.

`/VoL` appears in the *Negative Output Node* field. Because this is a differential oscillator, the negative output node is not `/gnd!` as it was for the *tline3oscRF* oscillator.

10. In the *Input Source* cyclic field, choose *none*.

The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.



11. In the Choosing Analyses form, verify that the *Enabled* field at the bottom of the Pnoise form is highlighted. Then click *OK* at the top of the Choosing Analyses form.

The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the Simulation window.

Analyses

#	Type	Arguments..................					Enable
1	pnoise	7	1	100M	5	..	yes
2	pss	1.9G	5	/VoR	/VoL		yes

Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Fundamental Frequency

➤ To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

The Waveform window and the Direct Plot form appear.

To plot the fundamental frequency, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pss* for *Analysis*.
3. Highlight *Harmonic Frequency* for *Function*.

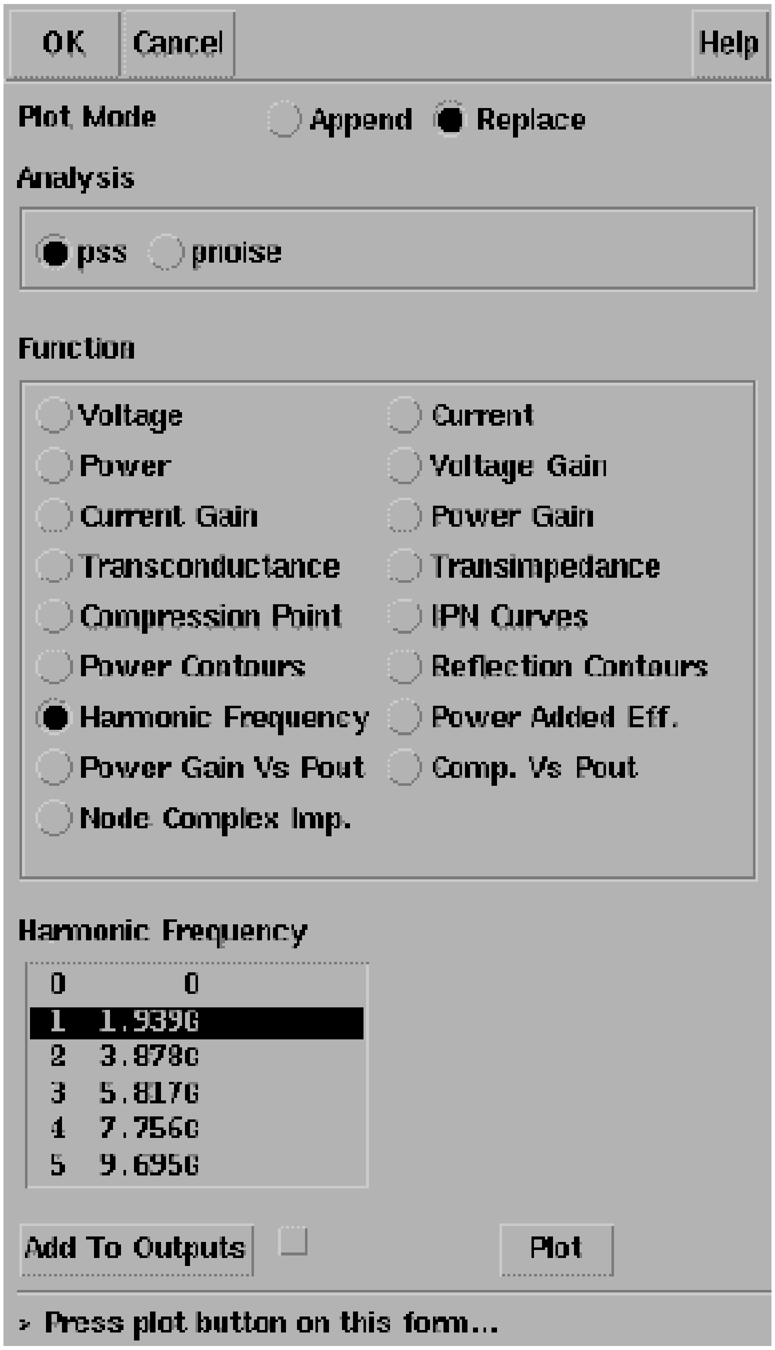
   The Direct Plot form changes to display available frequencies in the *Harmonic Frequency* list box.

4. Follow the prompt at the bottom of the Direct Plot form,

   Select Harmonic Frequency on this form...

   then highlight the first harmonic in the *Harmonic Frequency* list box.

The Direct Plot form appears as follows.

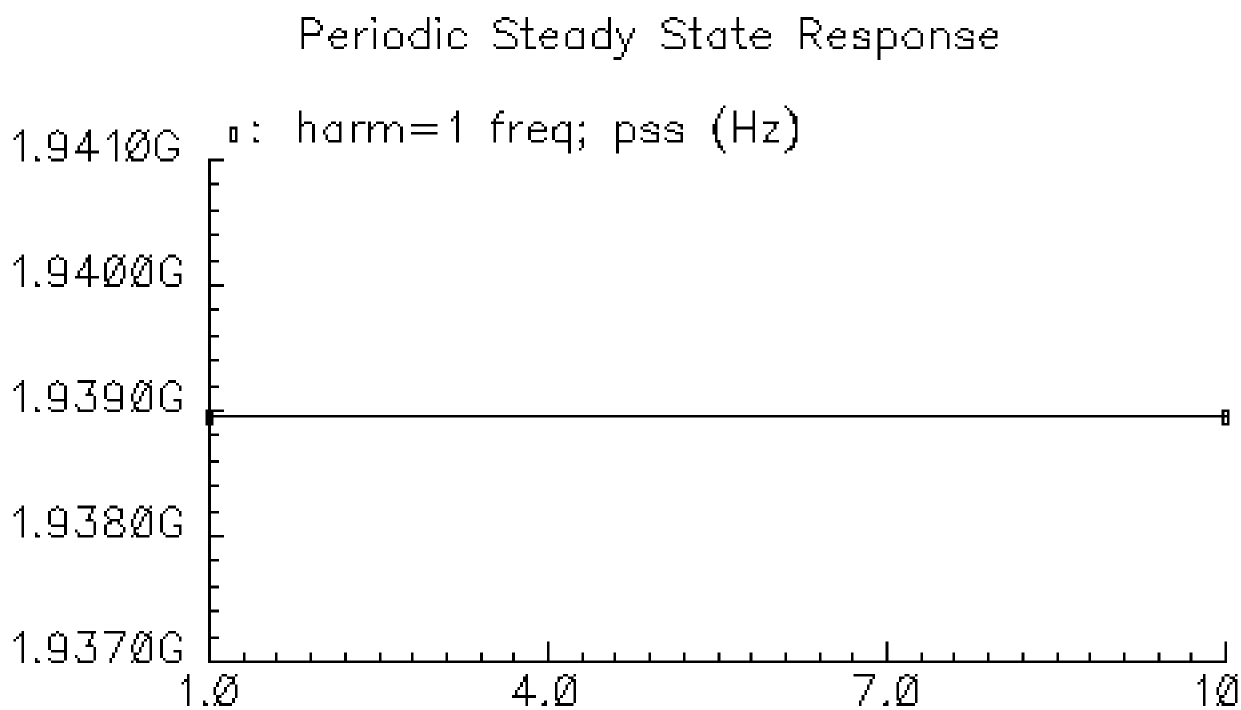


5. Follow the prompt at the bottom of the Direct Plot form,

Press plot button on this form...

then click *Plot*.

The Waveform window displays the fundamental frequency.



Notice that the fundamental frequency value for the first harmonic of `1.939 G` is also displayed in the Harmonic Frequency list box in the Direct Plot form.

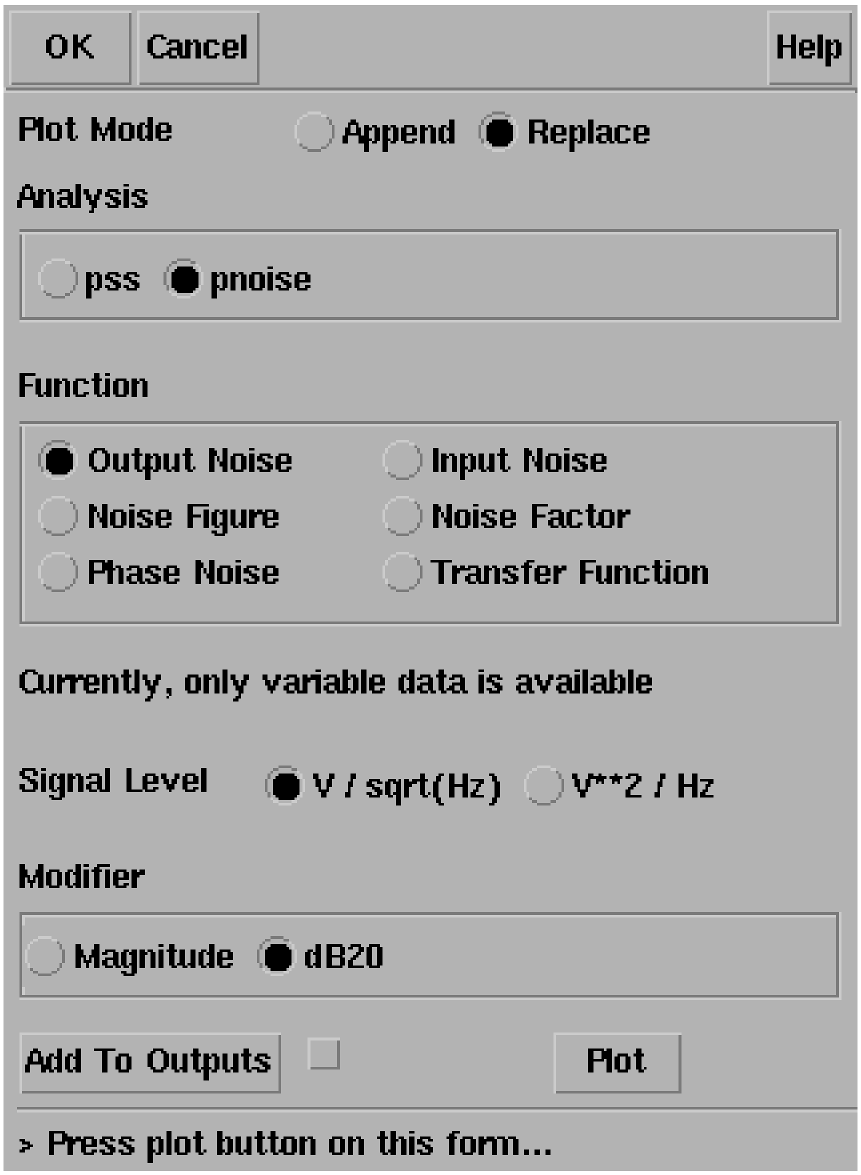
Plotting the Output Noise and Phase Noise

If necessary, in the Simulation window choose *Results - Direct Plot - Main Form* to open the Waveform window and the Direct Plot form.

To plot the output noise, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. If necessary, highlight *pnoise* for *Analysis*.
3. Highlight *Output Noise* for *Function*.
4. Highlight *V/sqrt(Hz)* for *Signal Level*.

5. Highlight *dB20* for *Modifier*.

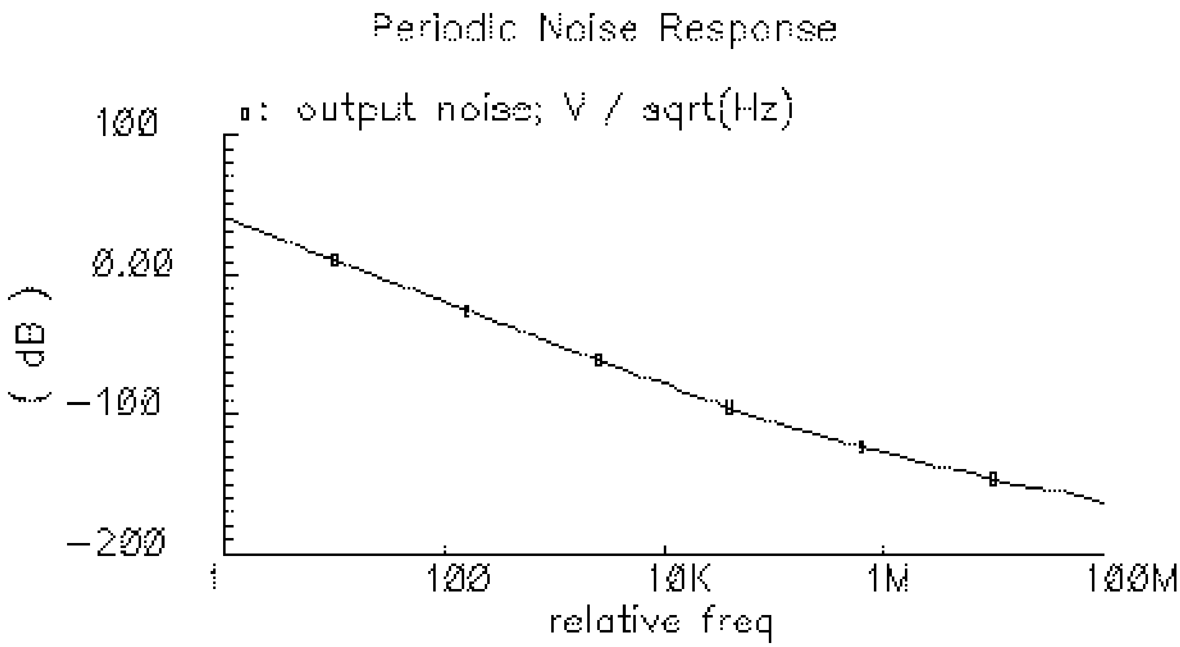


6. Follow the prompt at the bottom of the Direct Plot form,

   Press plot button on this form...

   then click *Plot*.

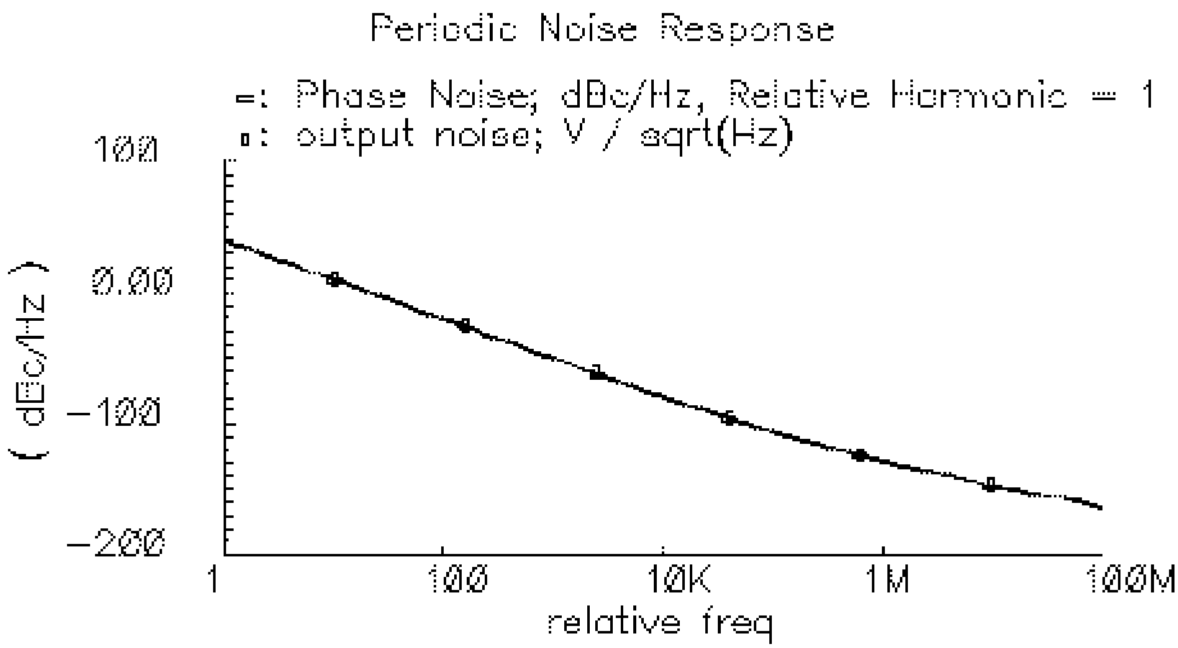
The output noise is plotted in the Waveform window.



To plot phase noise on the same graph with output noise, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *Phase Noise* for *Function*.
3. Click on *Plot*.

The phase noise plot is added to the output noise plot.

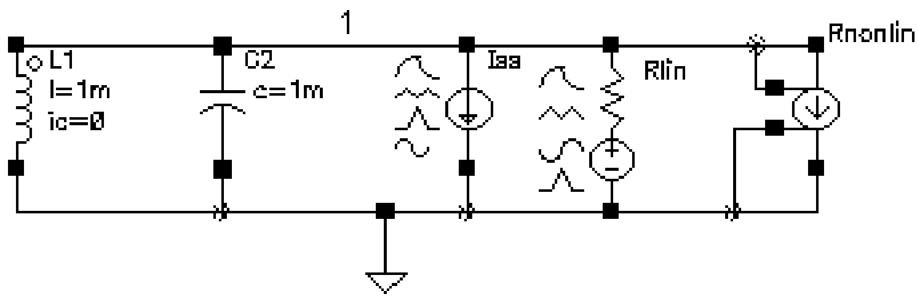


By plotting the phase noise on the same graph as the output noise, you can see the linear relationship between the output noise and the phase noise. The output noise is scaled by the carrier amplitude to produce the phase noise value.

The Van der Pol Circuit: Measuring AM and PM Noise Separation

As a simple example, consider the `vdp_osc` circuit, a feedback amplifier circuit shown in Figure 5-3. This example separates oscillator noise into AM and PM components. It also calculates USB and LSB noise for the oscillator.

Figure 5-3 Schematic for the vdp_osc Feedback Amplifier Circuit



The `vdp_osc` is a parallel RLC circuit with a nonlinear transconductance represented by a polynomial voltage controlled current source. At small capacitor voltages, the transconductance is negative; that is, it is an active device which creates positive feedback that seeks to increase the voltage on the capacitor. At larger capacitor voltages, where the transconductance term goes into compression, it effectively acts as a positive resistor (that is, it creates negative feedback) and limits the voltage. You use a cubic polynomial model for the circuit.

Simulating the vdp_osc Circuit

Before you begin, be sure that you have performed the setup procedures described in Chapter 3, “Setting Up for the Examples.”

Opening the vdp_osc Circuit

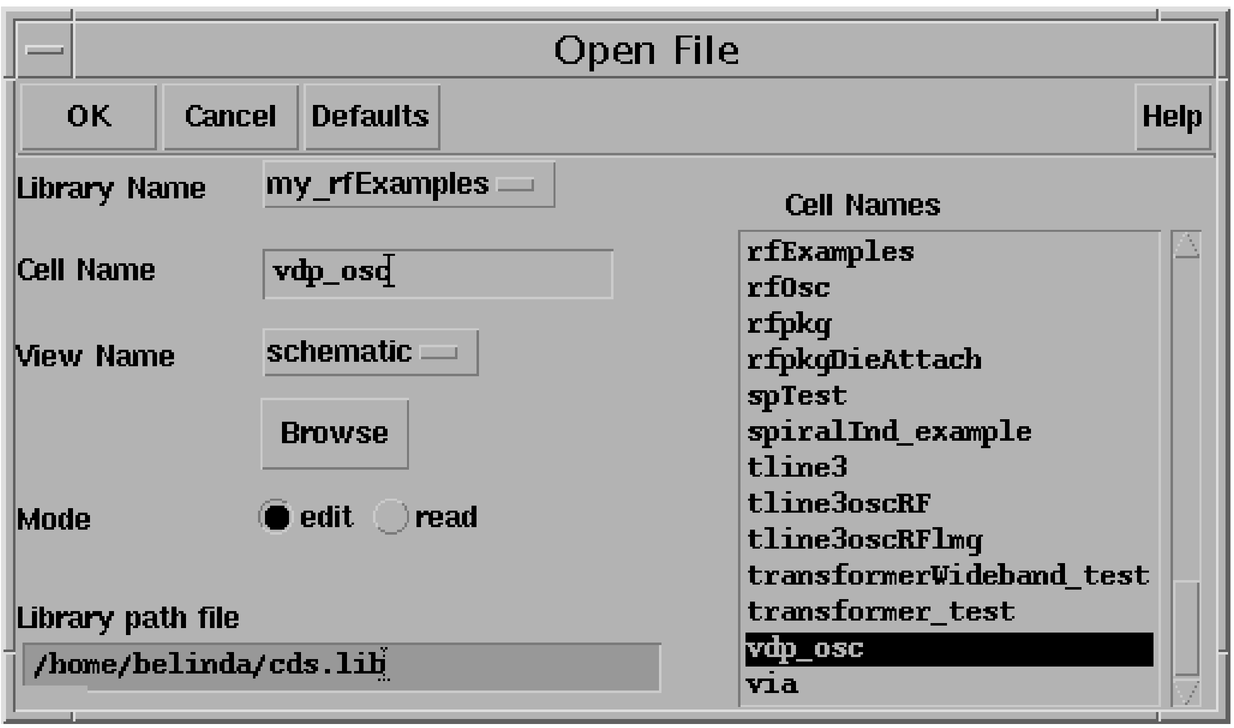
1. In the Command Interpreter Window (CIW), choose *File – Open*.

   The Open File form appears. Filling in the Open File form opens the vdp_osc schematic.

2. Choose *rfExamples* for *Library Name*. Choose the editable copy of the *rfExamples* library you created. See <u>Chapter 3, “Setting Up for the Examples”</u> for more information.
3. Choose *schematic* for *View Name*.
4. In the *CellNames* list box, highlight *vdp_osc*.

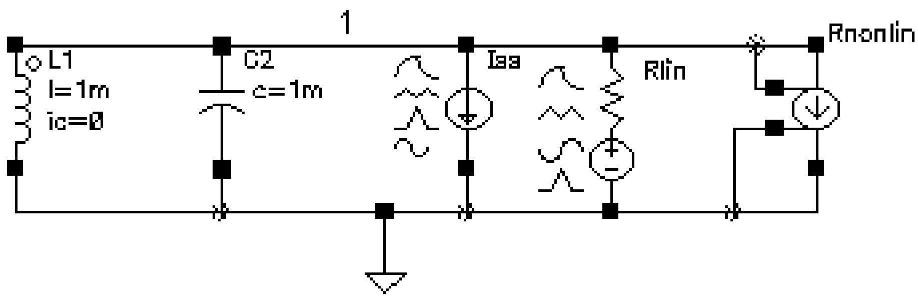
   vdp_osc appears in the Cell Name field.
5. Highlight *edit* for *Mode*.

   The completed Open File form appears like the one below.



6. Click *OK*.

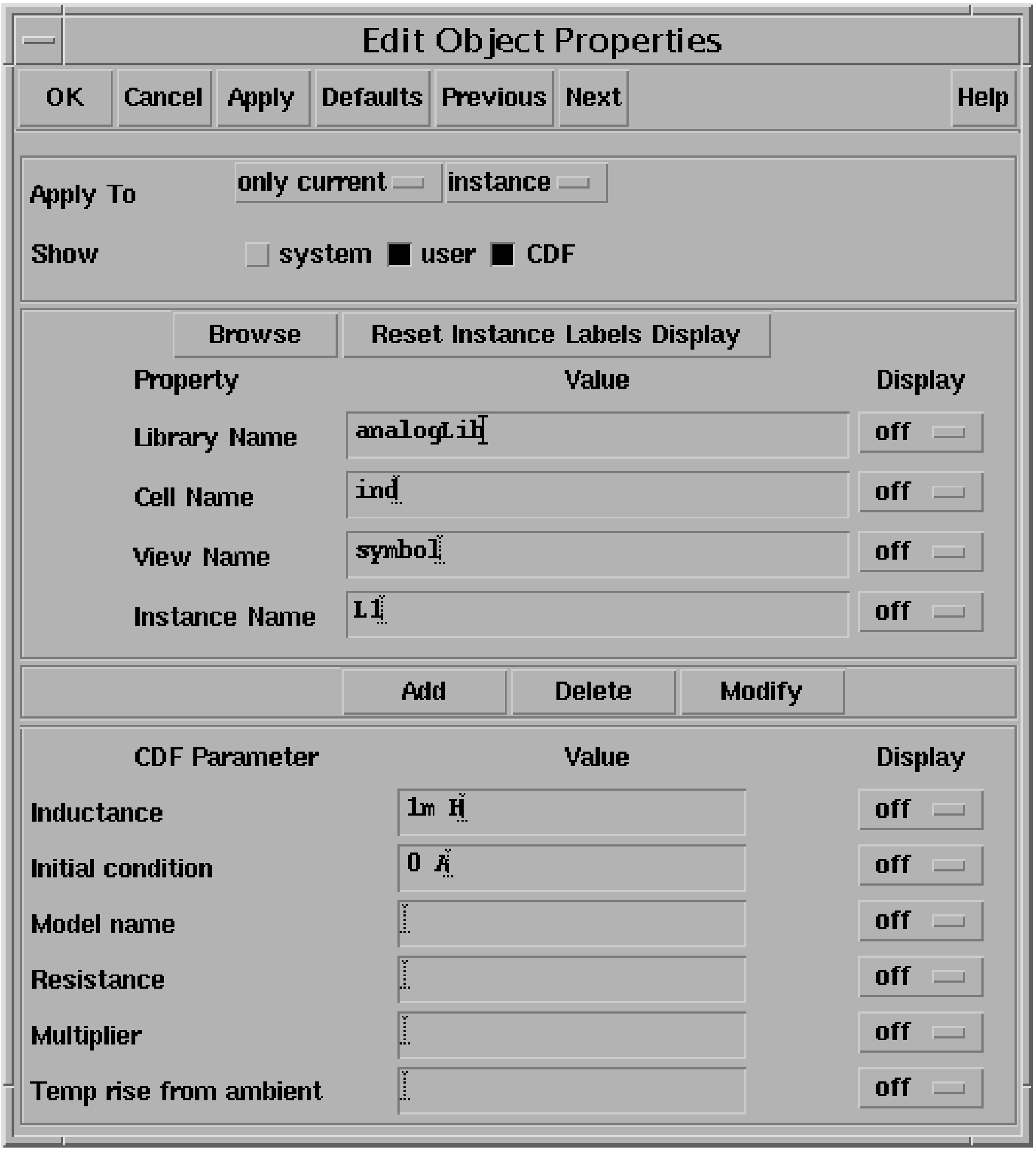
The Schematic window for the `vdp_osc` oscillator opens.



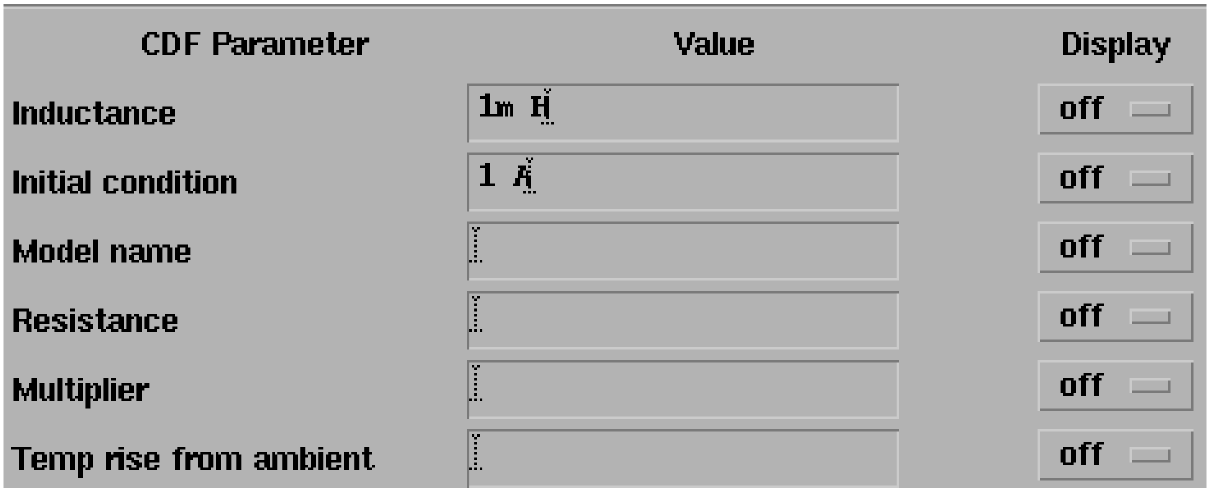
Editing Properties for the Inductor

1. In the Schematic window, select the L1 inductor.
2. With the L1 inductor selected, choose *Edit-Properties-Objects*.

The Edit Object Properties form displays for the L1 inductor.



3. Edit the *Initial condition* value by replacing the `0` with `1`.



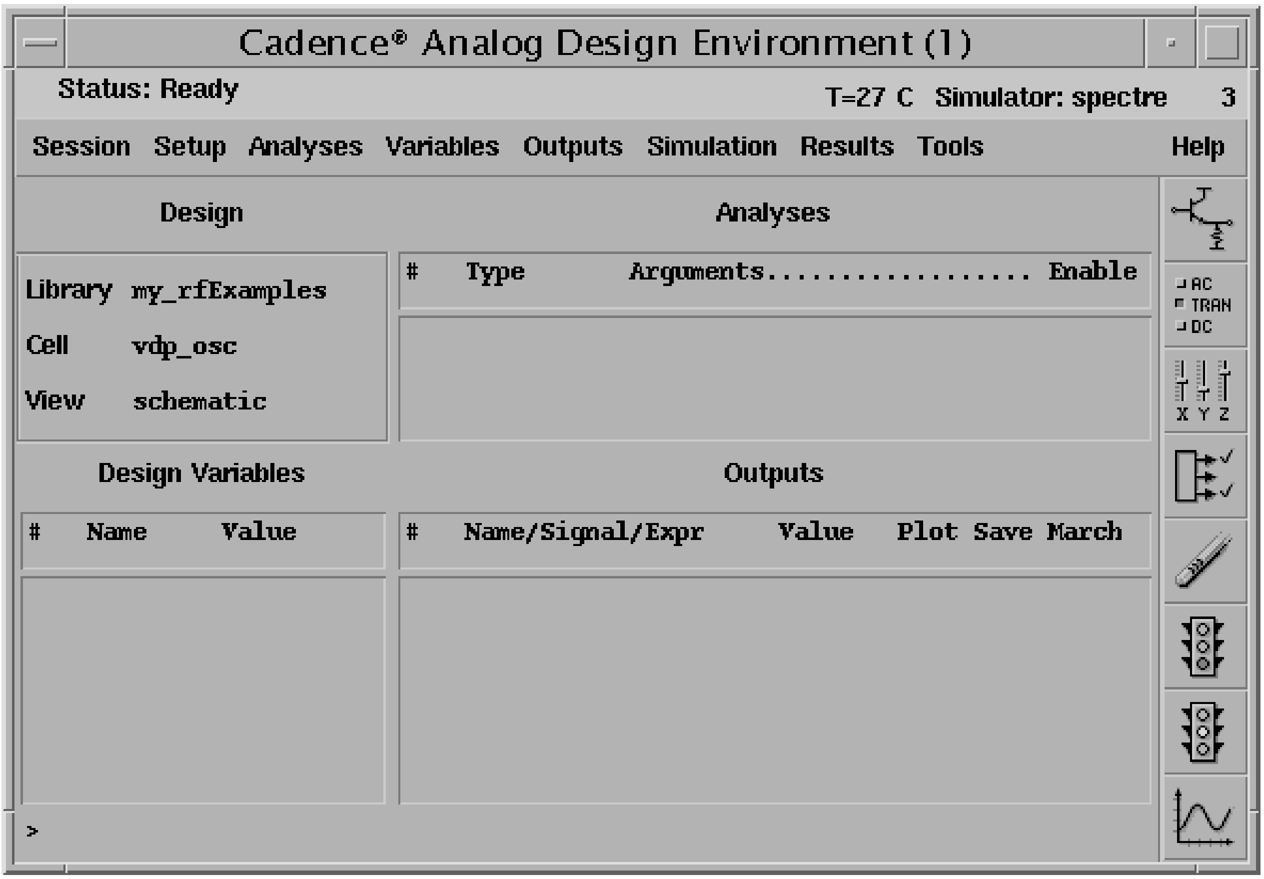
CDF Parameter	Value	Display
Inductance	1m H	off
Initial condition	1 A	off
Model name		off
Resistance		off
Multiplier		off
Temp rise from ambient		off

4. Click *OK* in the Edit Object Properties form.

Opening the Simulation Window

1. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window appears.



You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing *vdp_osc* in the Choosing Design form.

Measuring AM and PM Conversion with PSS and Pnoise

This example computes AM and PM conversion for the `vdp_osc` oscillator circuit. You perform a PSS analysis first because the periodic steady-state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the AM and PM noise components from the results of the PSS and `Pnoise` analyses.

Setting Up the Simulation

1. If necessary, open the `vdp_osc` circuit.
2. If necessary, in the Simulation window use *Analysis - Disable* to disable any analyses you ran previously. (Check the *Analyses* area in the Simulation window to verify whether or not any analyses are enabled.)

Setting Up the PSS Analysis

Note: If your oscillator circuit does not contain a stimulus to start the oscillator, you must run a `transient` analysis before you run this PSS analysis. From the `transient` analysis, you save the node voltages and use them as initial conditions to start the oscillator in the PSS analysis (see step 3).

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, choose *pss* for the *Analysis*.

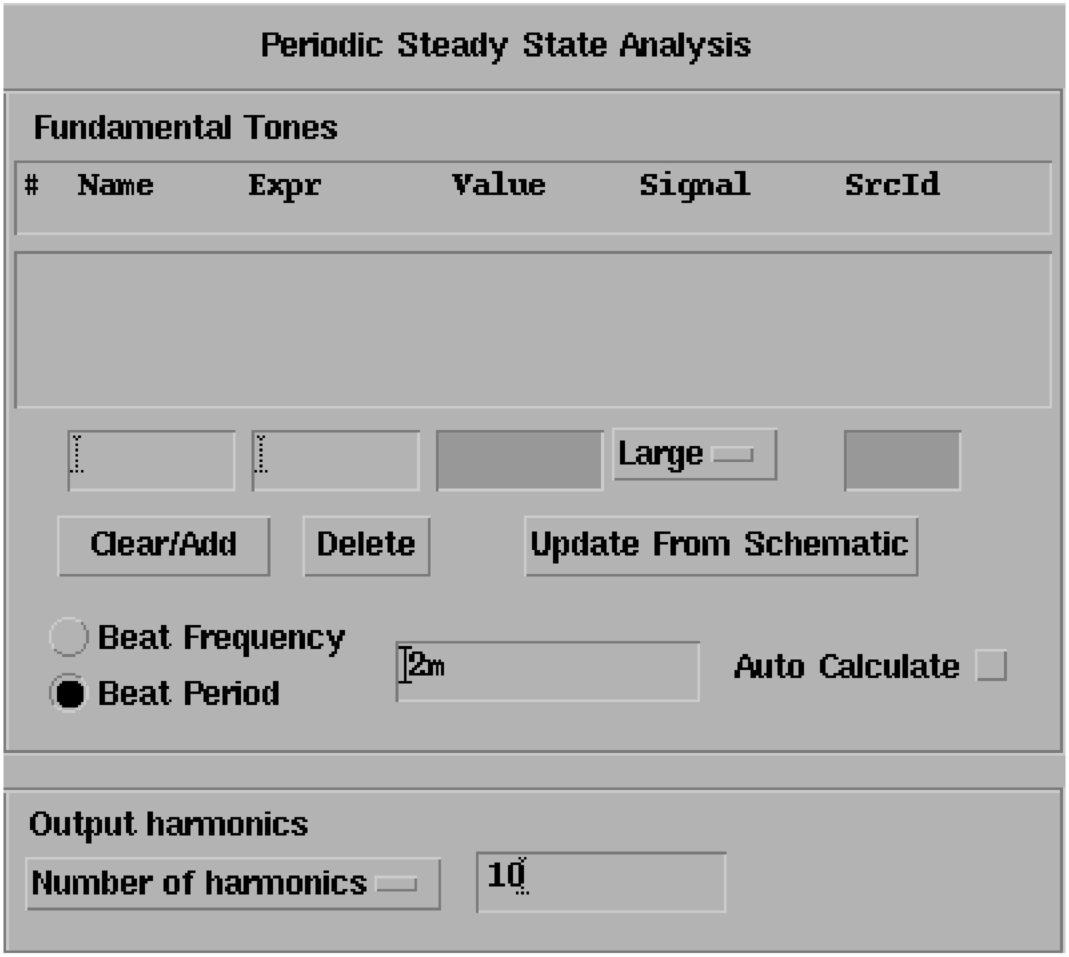
   The form changes to display options needed for the PSS analysis. You perform a PSS analysis first because the periodic steady state solution must be determined before you can perform a `Pnoise` small-signal analysis to determine the phase noise.

3. In the *Fundamental Tones* area, highlight *Beat Period.* Be sure the *Auto Calculate* button is *not* highlighted.

   In the field next to *Beat Period*, type `2m`. This is a estimate based either on the results of a prior transient analysis or quick manual calculations from LC oscillator equations.

4. In the *Output harmonics* cyclic field, choose *Number of harmonics.* Type `10` in the *Number of harmonics* field.

The top of the PSS Choosing Analysis form appears below.

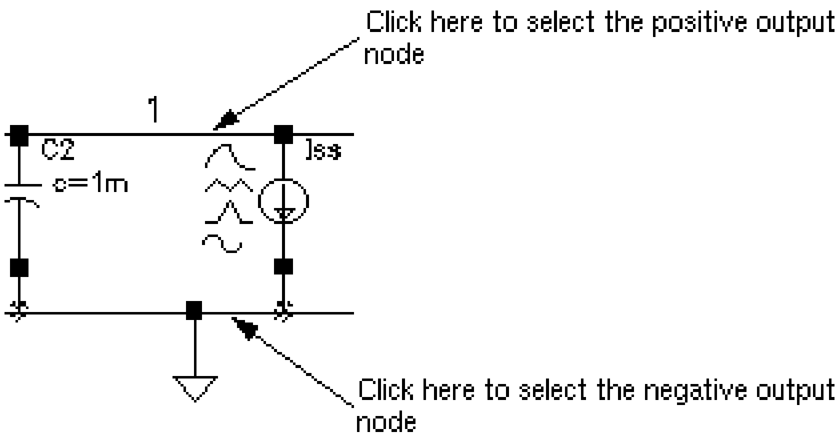


5. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.

6. Highlight *Oscillator*.

The form changes to let you specify the two nodes needed for oscillator simulation.

7. Click on the *Oscillator node Select* button. Then click on the appropriate wire in the Schematic window to choose the oscillator node. In this example, choose the node labelled `1`.



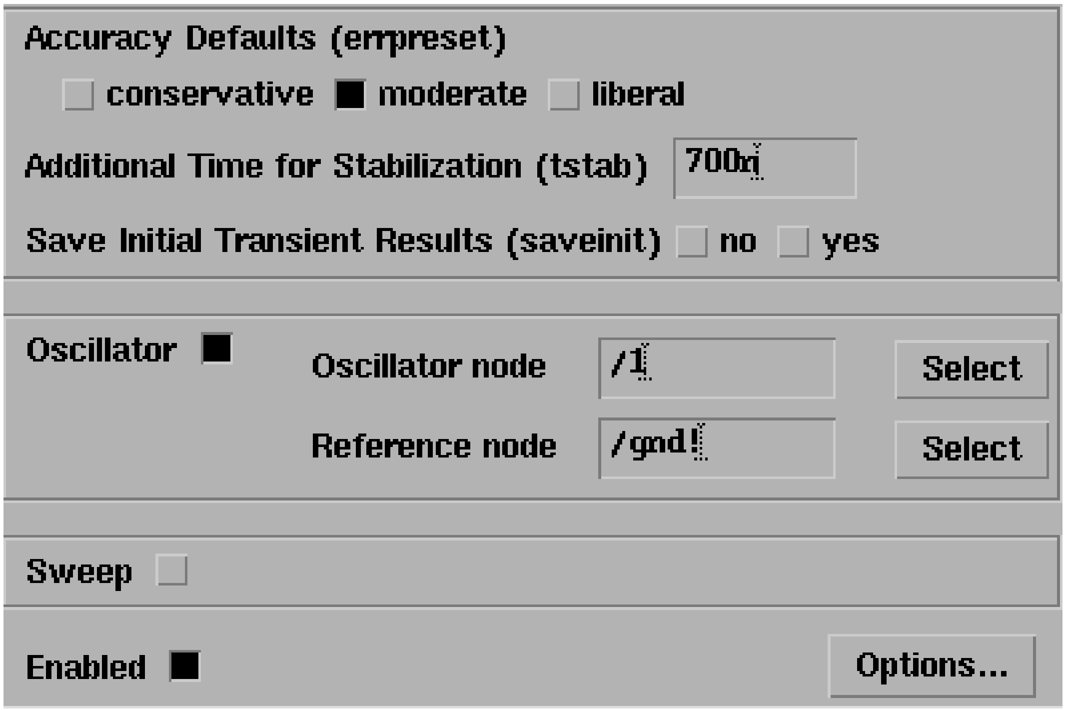
`/1` appears in the *Oscillator node* field.

8. Click on the *Reference node Select* button. Then click on the appropriate wire in the Schematic window to choose the reference node. In this example, choose the node where the ground is located.

`/gnd!` appears in the *Reference node* field.

You can also leave the *Reference node* field empty as the *Reference node* field defaults to `/gnd!`.

The bottom of the Choosing Analyses form for PSS looks like the following.



9. Verify that *Enabled* is highlighted for the PSS analysis and click *Apply*.

   Correct any errors reported for the PSS analysis after you click *Apply*.

Setting Up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click on *pnoise*.

   The form changes to let you specify data for the `Pnoise` analysis.

2. In the *Sweeptype* cyclic field, choose *Relative*. Enter `1` in the *Relative Harmonic* field.

   This choice specifies that the *Frequency Sweep Range (Hz)* values you choose represent frequency values away from the fundamental frequency. For example, if you specify 2K, you choose a value 2K away from the fundamental frequency. This example uses 1, the first harmonic, since this is the best way to present phase noise for an oscillator which shows up next to the oscillator's fundamental frequency.

3. In the *Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop* and type `5` and `10k` as the *Start* and *Stop* values, respectively.

4. In the *Sweep Type* cyclic field, choose *Logarithmic*. Use a log sweep when you plot phase noise because of the size of the phase noise values.

5. Highlight *Number of Steps* and type `501` in the *Number of Steps* field.

   Note: Be sure to use a nonzero starting value for a logarithmic sweep.

   This example uses 501 steps since it is a large number of steps and will you to see all the small features and get very accurate noise calculations. Since this is a very small circuit, we can use any number of steps. Your circuit might require a different number of steps.

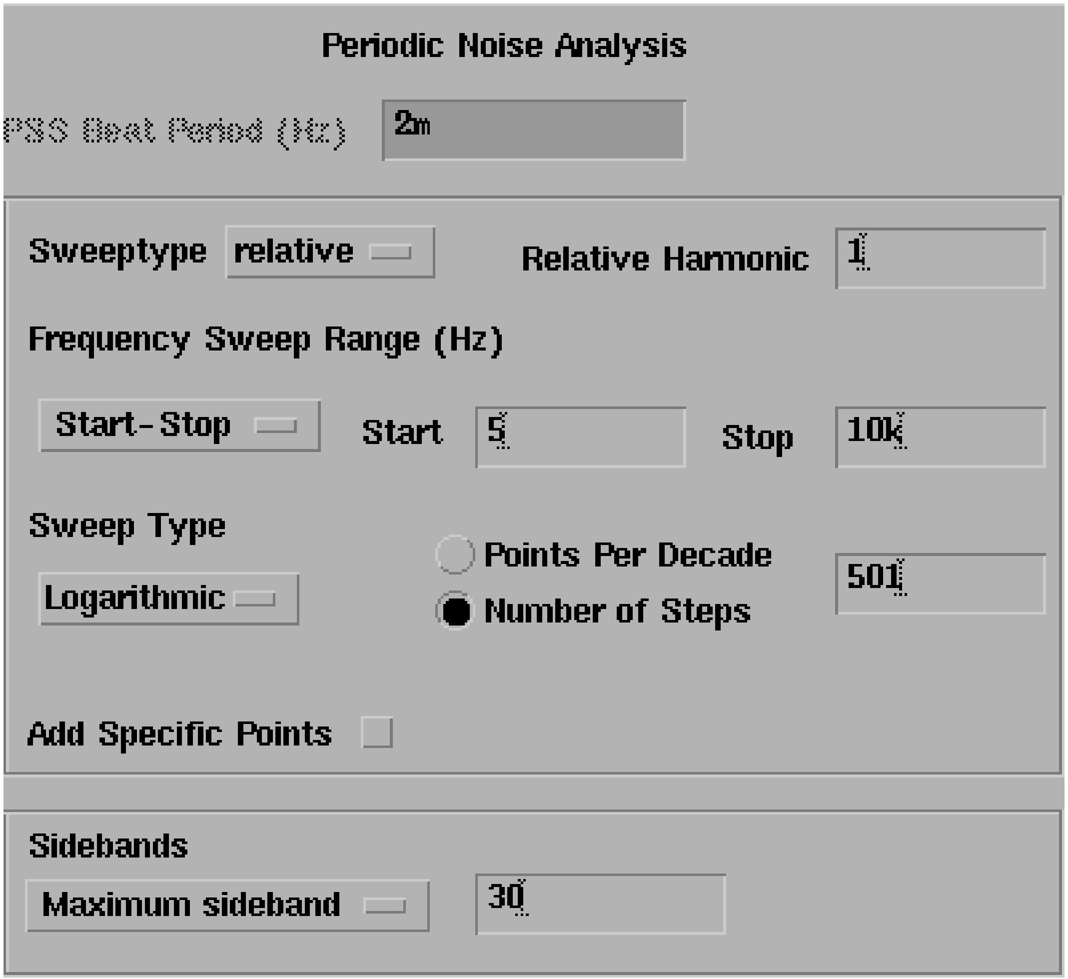
6. In the *Sidebands* cyclic field, choose *Maximum sideband* and type `30` in the *Sidebands* field.

   Note: Be sure you always use a nonzero *Maximum sideband* value. The default `maxsideband` parameter value is `7`.

   As in step 5, a large *Maximum sideband* value provides more accurate noise calculations by allowing for more harmonic folding. In fact, in this small circuit the large *Maximum sideband* value provides no advantage.

   In your circuit you will have to experiment by increasing the *Maximum sideband* value in a series of simulations until the change in additional accuracy diminishes to an amount you are satisfied with.

The top of the Choosing Analyses form for the Pnoise analysis looks like this.

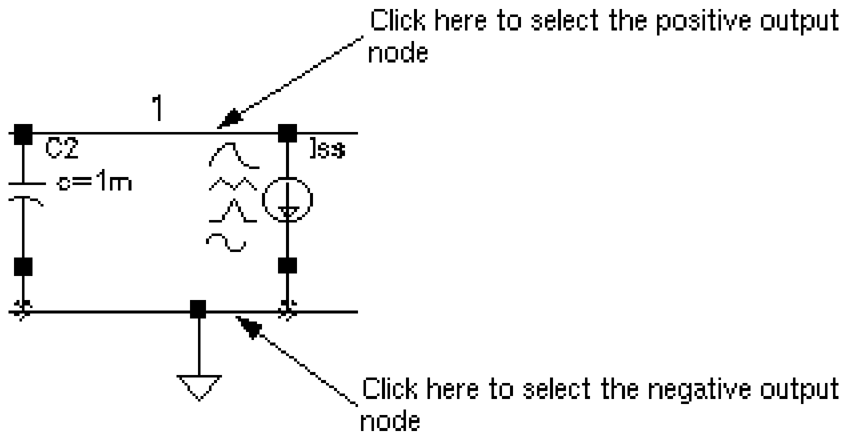


7. In the *Output* cyclic field, choose *voltage*.

Notice the informational message that displays in the CIW.

```
_asii_spectre1_analysis_form->pnoise_outType->value = “voltage”
*info*  Since voltage was selected as the output measurement
        technique for the pnoise analysis, spectre will NOT
        subtract any load resistance contribution from the Noise
        Figure calculation. If this is undesirable, please select
        probe as the output measurement technique and select a port.
“voltage”
```

8. Click on the *Positive Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose */1*.



`/1` appears in the *Positive Output Node* field.

9. Click on the *Negative Output Node Select* button. Then click on the appropriate wire in the Schematic window to choose the ground.

`/gnd!` appears in the *Negative Output Node* field.

10. In the *Input Source* cyclic field, choose *none*.

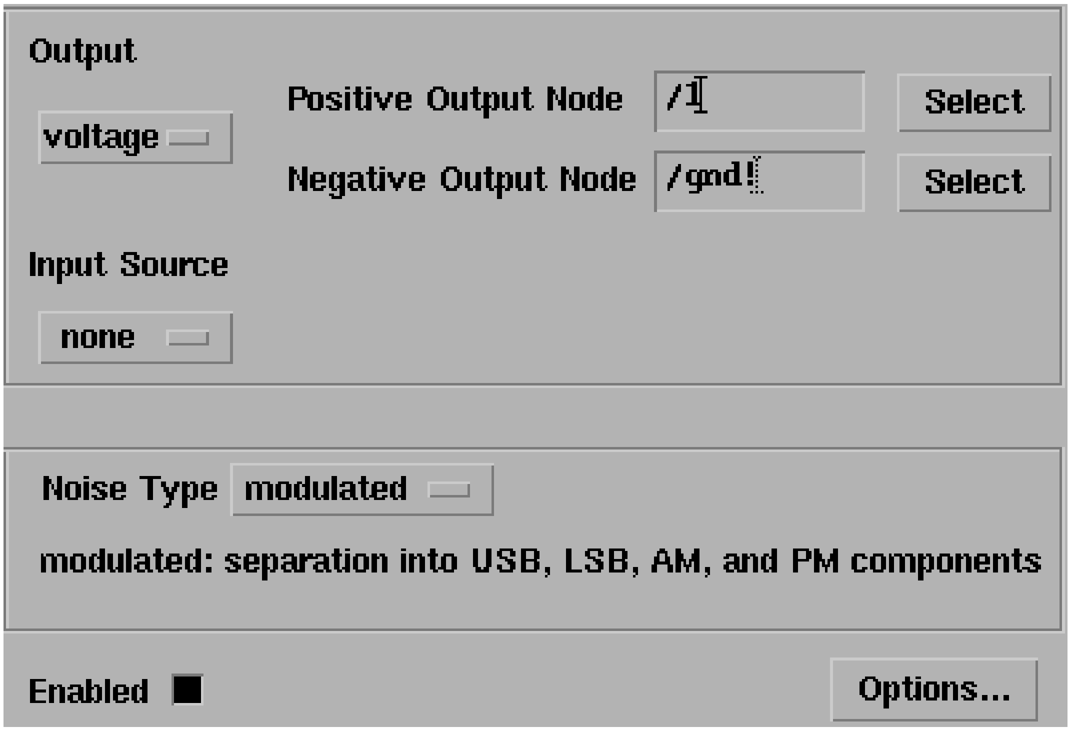
In this case, since we are calculating Total Noise and its AM and PM components, the Input Source is not needed. (Using an Input Source generates NF, IRN, and other parameters.) Input Source is not usually used in oscillator simulations.

11. In the *Noise Type* cyclic field, choose *modulated*.

The following text appears below the *Noise Type* cyclic field.

modulated: separation into USB, LSB, AM, and PM components

The bottom of the Choosing Analyses form for the `Pnoise` analysis looks like this.



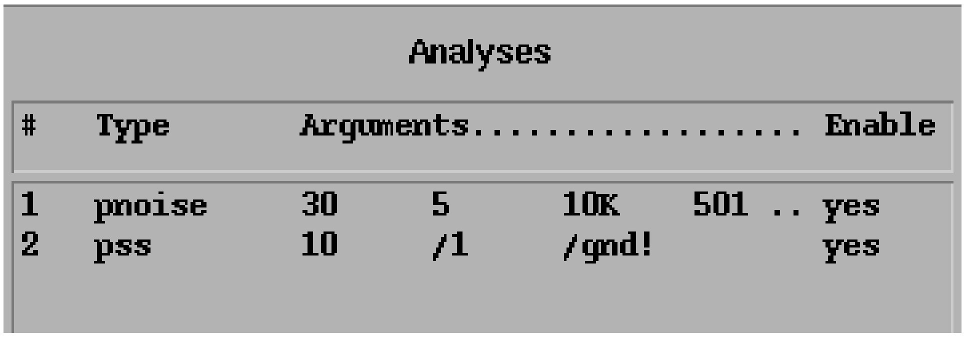
12. Verify that the *Enabled* field for the `Pnoise` analysis is highlighted. Then click *Apply* at the top of the Choosing Analysis form.

Correct any errors reported for the `Pnoise` analysis after you click *Apply*.

13. Click *OK.*

The Choosing Analyses form closes.

The `pss` and `pnoise` analyses you set up appear in the *Analyses* list box in the Simulation window.



Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting Modulated Pnoise with the dBV Modifier

The dBc Modifier on the Direct Plot form plots the noise power in decibels, relative to 1 volt, on the Y axis

Open the Direct Plot form.

➤ To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

   The Waveform window and the Direct Plot form appear.

To plot the modulated pnoise, start by plotting USB noise and then plot AM and PM noise in the same Waveform window.

Plotting USB Noise

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise modulated* for *Analysis*.

This selects the results of the modulated pnoise analysis for display and changes the appearance of the Direct Plot form.

3. Highlight *USB* for *Noise Type*.

   This selects to plot the standard pnoise analysis results.

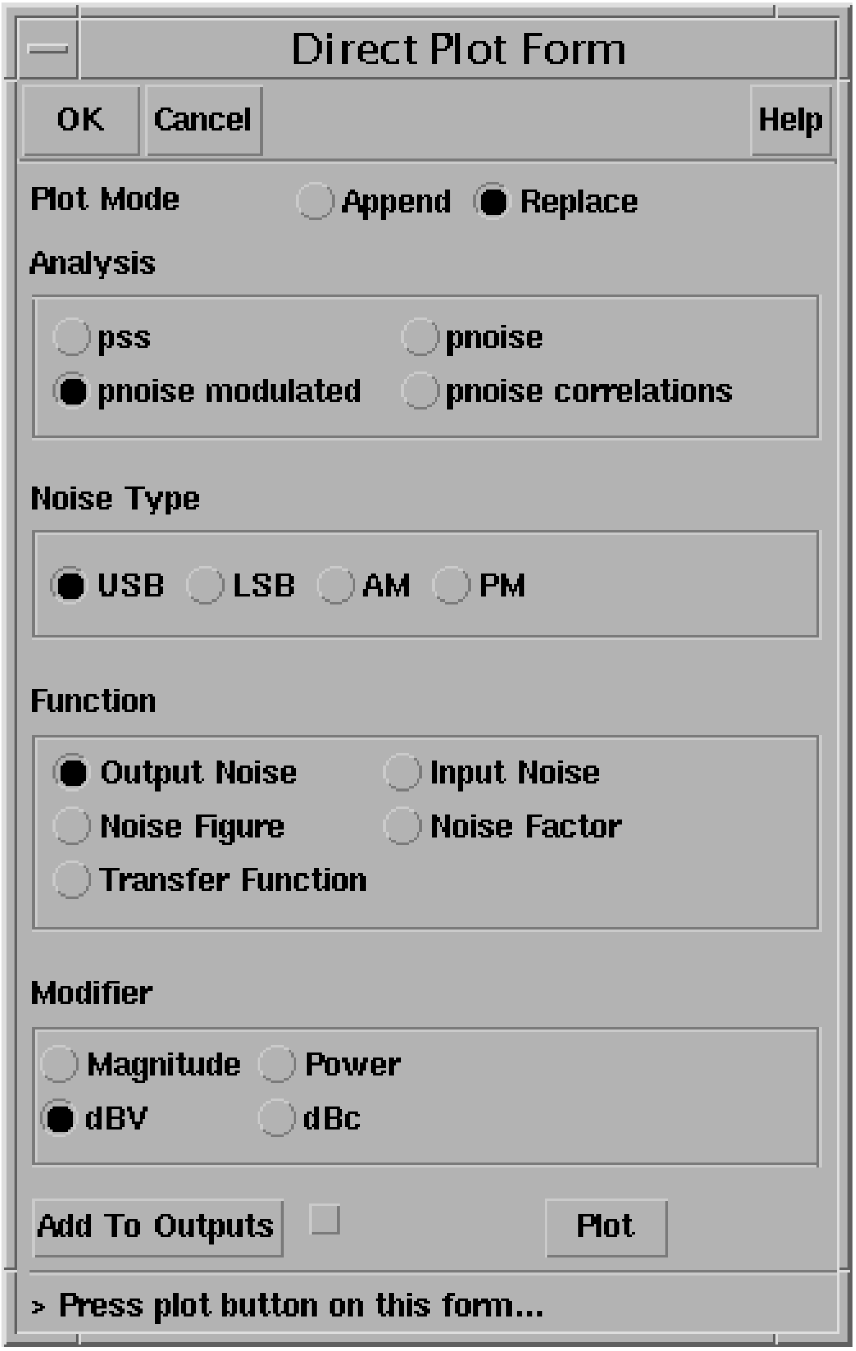
4. Highlight *Output Noise* for *Function*.

   This selects to plot Output noise.

5. Highlight *dBV* for *Modifier*.

   Selecting dBV plots the noise power in decibels, relative to 1 volt, on the Y axis.

The Direct Plot form appears as follows.

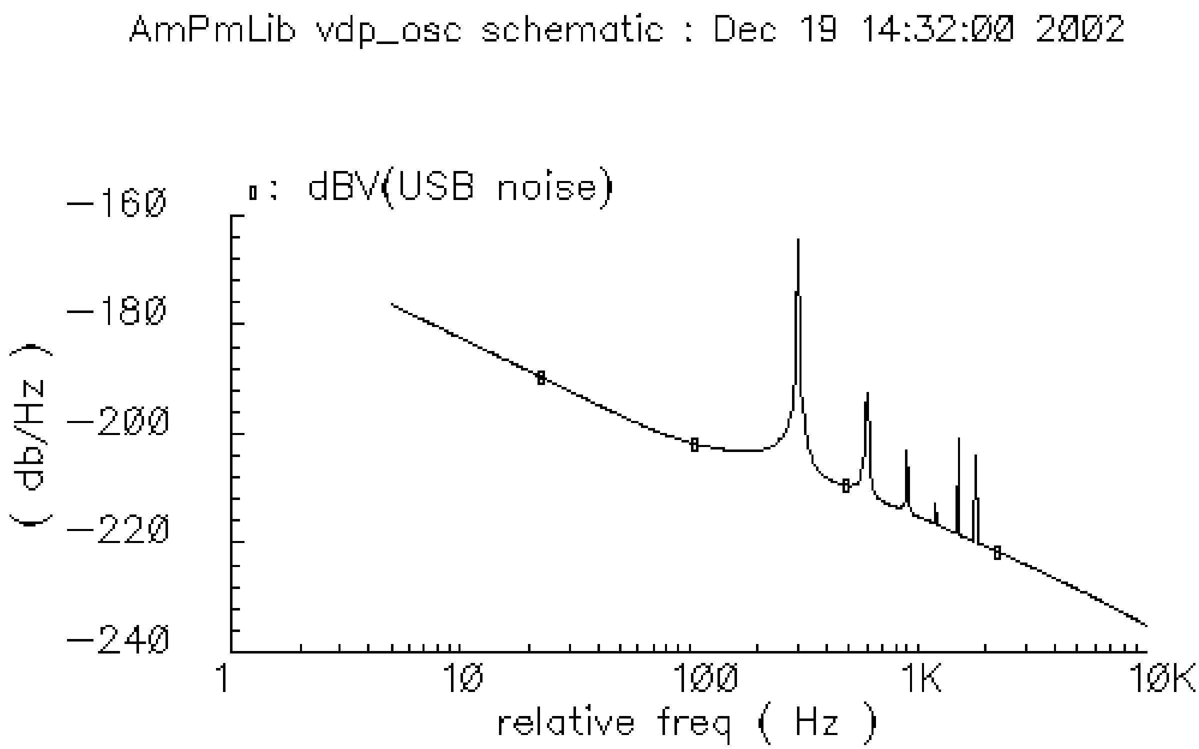


6. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The Waveform window displays the USB noise function.



Upper sideband noise appears in the Direct Plot form.

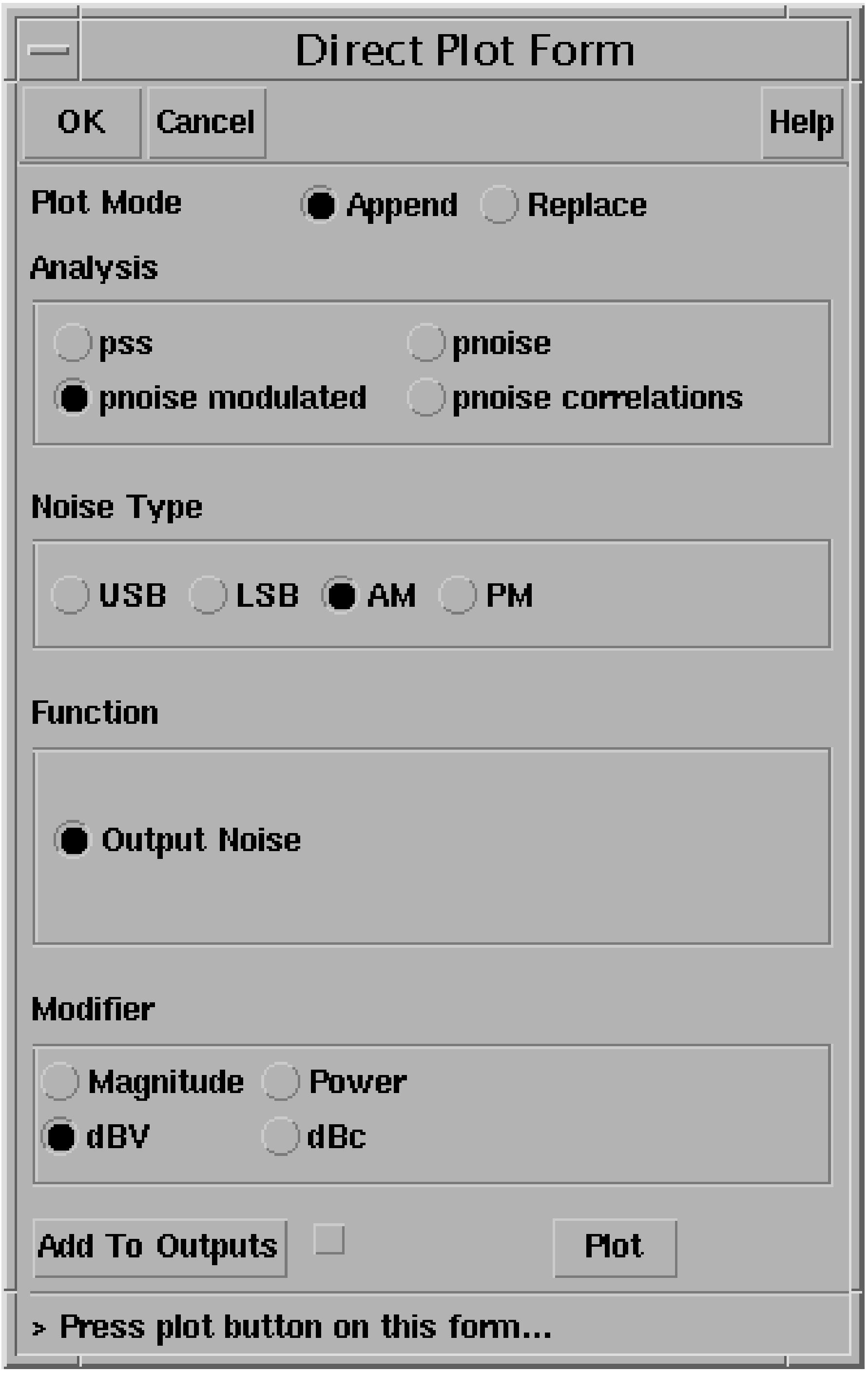
Plotting AM Noise

Plot the AM and PM noise in the same Waveform window from the same Direct Plot form. Start by plotting AM noise:

1. Highlight *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *AM* for *Noise Type*.

   When you highlight *AM* for *Noise Type*, all *Function* selections other than *Output Noise* go away.
4. Keep *dBV* for *Modifier*.

The Direct Plot form appears as follows.

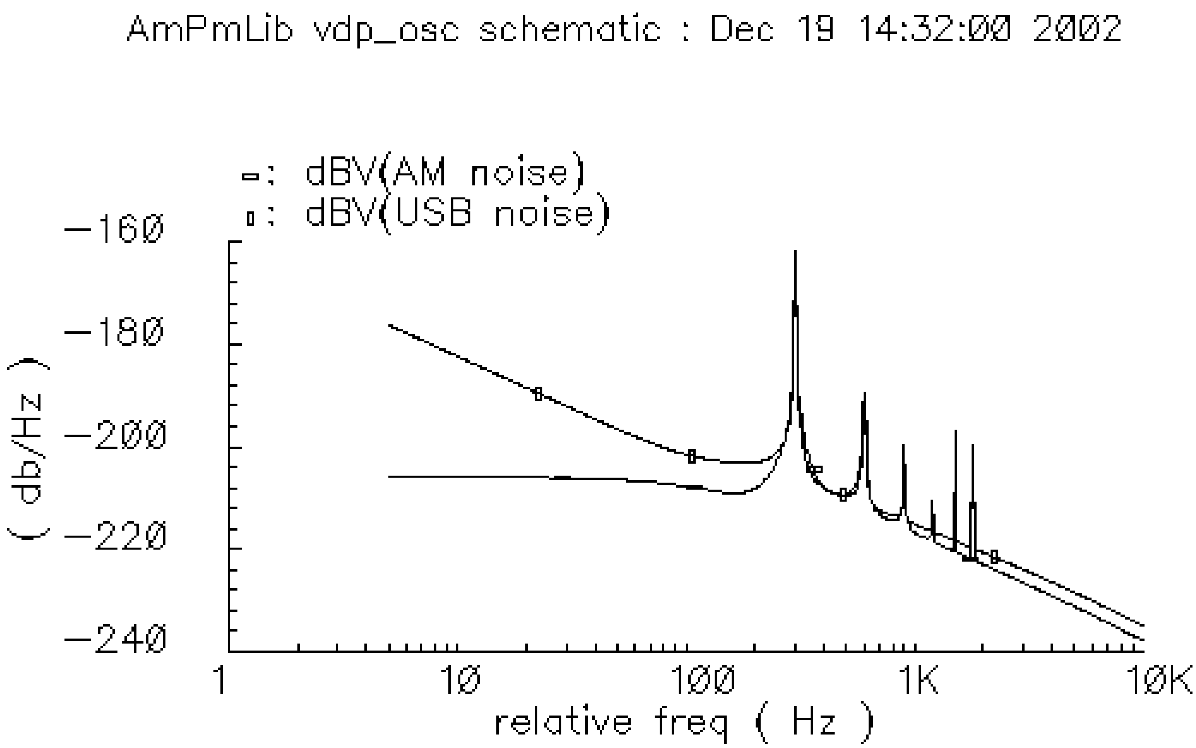


5. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The Waveform window displays the USB and AM noise functions.



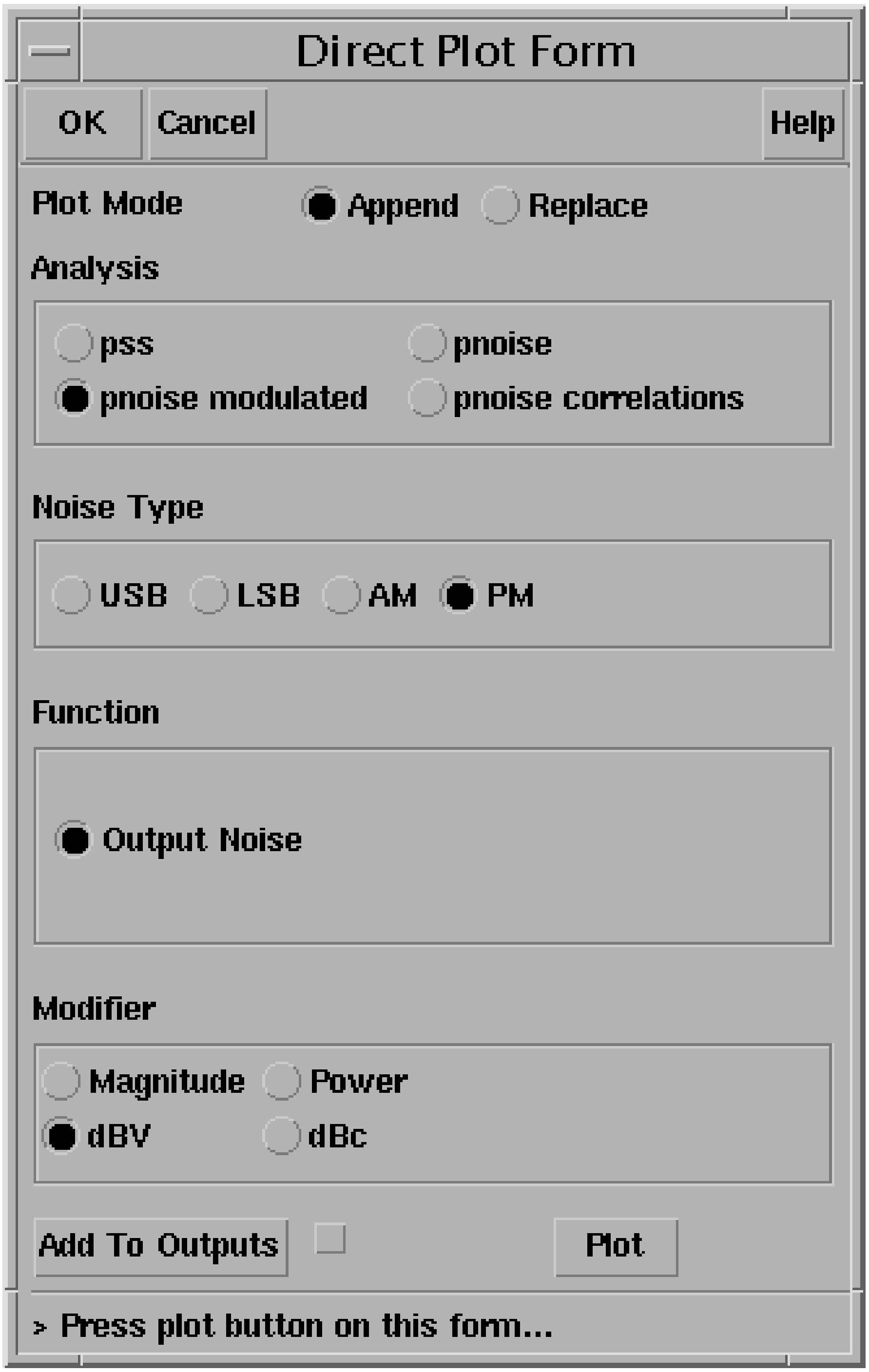
AM noise is added to the Direct Plot form.

Plotting PM Noise

Finally, plot the PM noise in the same Waveform window from the same Direct Plot form.

1. Keep *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *PM* for *Noise Type*.
4. Keep *Output Noise* for *Function*.
5. Keep *dBV* for *Modifier*.

The Direct Plot form appears as follows.

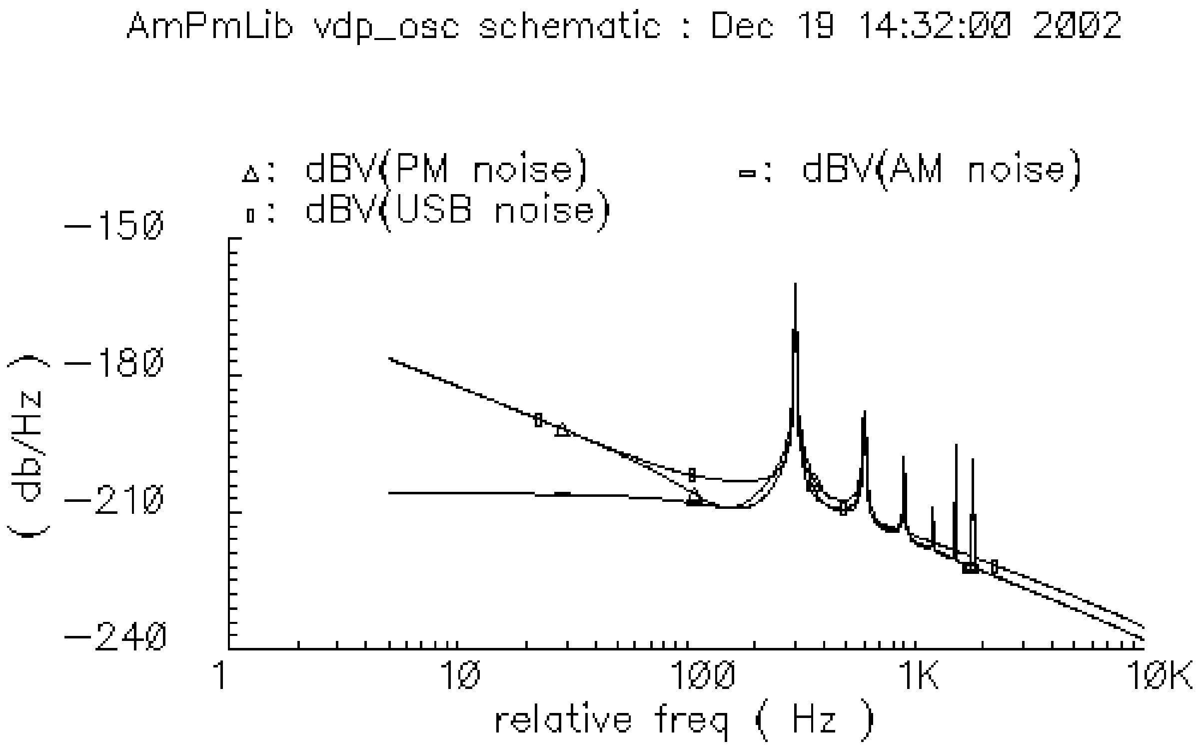


6. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

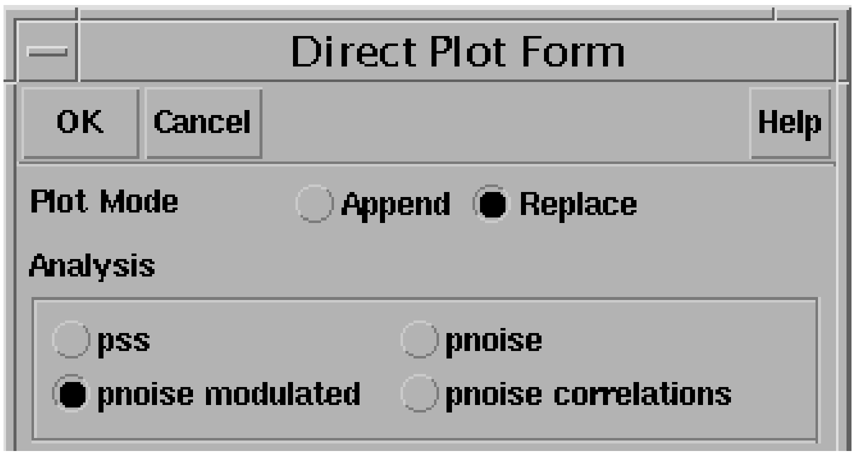
The Waveform window displays the USB, AM and PM noise functions.



PM noise is added to the Direct Plot form.

A Note on Pnoise Correlations

The *Analysis* choice *pnoise correlations* is available in the Direct Plot form as a by-product of the *pnoise modulated* simulation. In fact the *pnoise modulated* measurement is derived from the results of the other two analyses, *pnoise* and *pnoise correlations*.



While *pnoise* and *pnoise correlations* are analyses native to SpectreRF, *pnoise modulated* is a measurement which is derived from *pnoise* and *pnoise correlations*. This means that *pnoise modulated* is available only when you use SpectreRF in the analog design environment. and *pnoise modulated measurements* are available only from the analog design environment's Direct Plot form.

Plotting Modulated Pnoise with the dBc Modifier

The dBc Modifier on the Direct Plot form plots the noise power in decibels, relative to the carrier, on the Y axis.

If necessary, open the Direct Plot form.

➤ To open the Direct Plot form, choose *Results – Direct Plot – Main Form* in the Simulation window.

The Waveform window and the Direct Plot form appear.

To plot the modulated pnoise, start by plotting USB noise and then plot AM and PM noise in the same Waveform window.

Plotting USB Noise

1. Highlight *Replace* for *Plot Mode*.

2. Highlight *pnoise modulated* for *Analysis*.

   This selects the results of the modulated pnoise analysis for display.

3. Highlight *USB* for *Noise Type*.

   This selects to plot the standard pnoise analysis results.

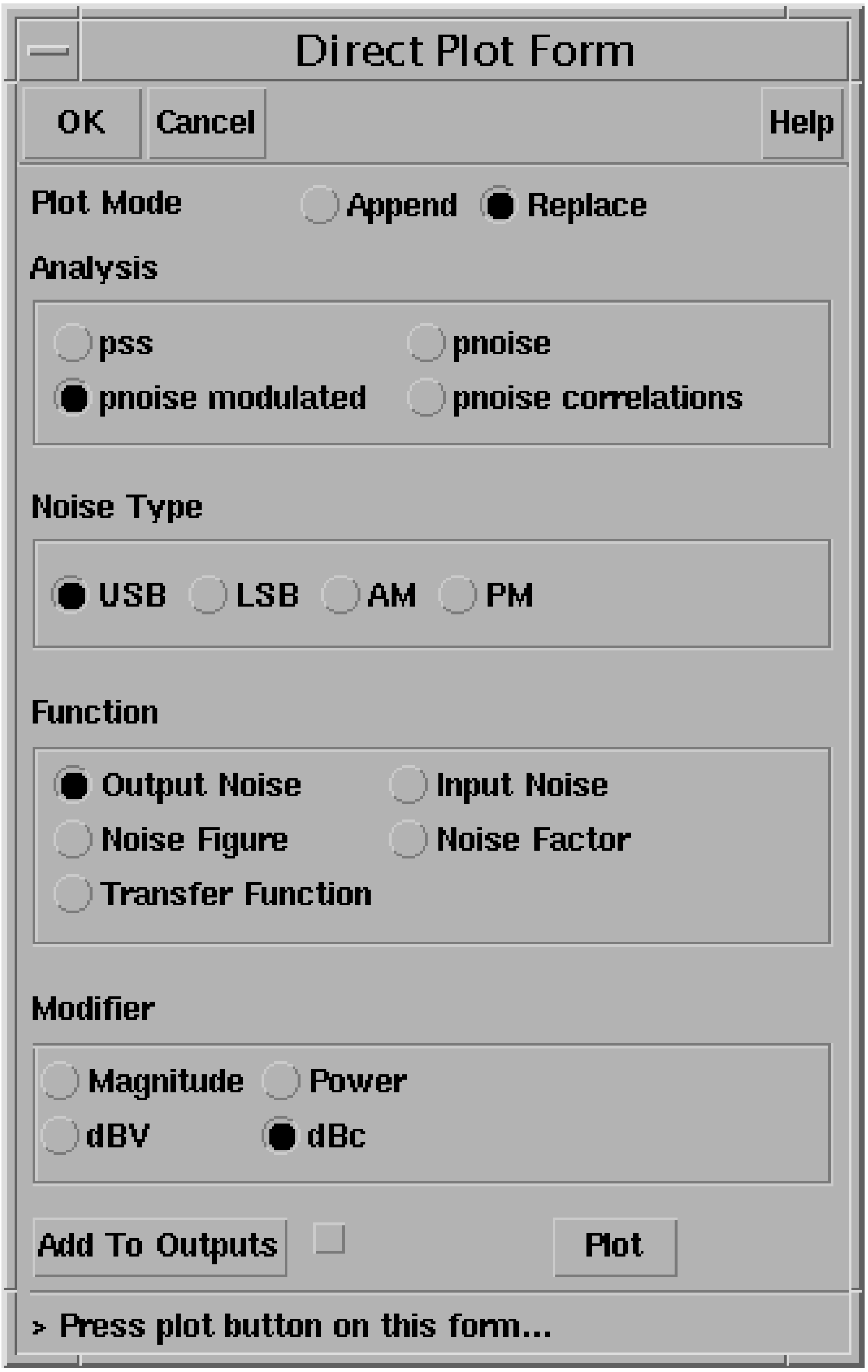
4. Highlight *Output Noise* for *Function*.

   This selects to plot Output noise.

5. Highlight *dBc* for *Modifier*.

   Selecting dBc plots the noise power in decibels, relative to the carrier, on the Y axis.

The Direct Plot form appears as follows.

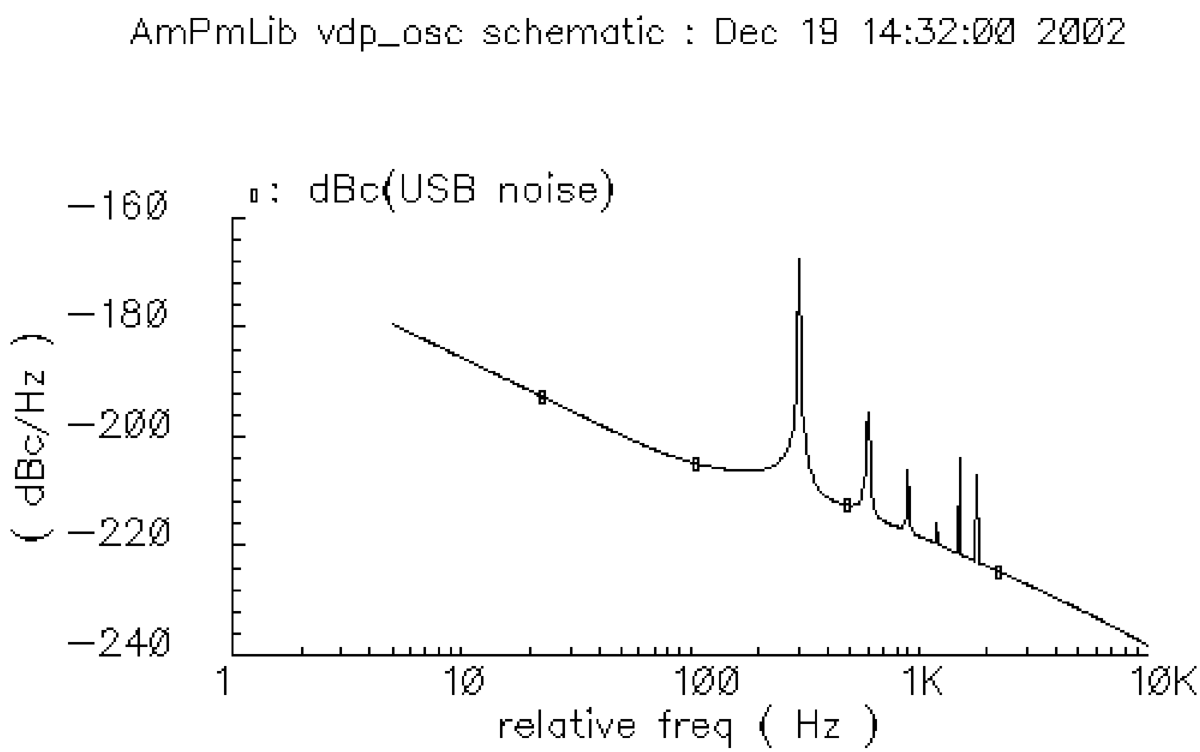


6. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The Waveform window displays the USB noise function.



The upper sideband noise appears in the Direct Plot form.

Plotting AM Noise

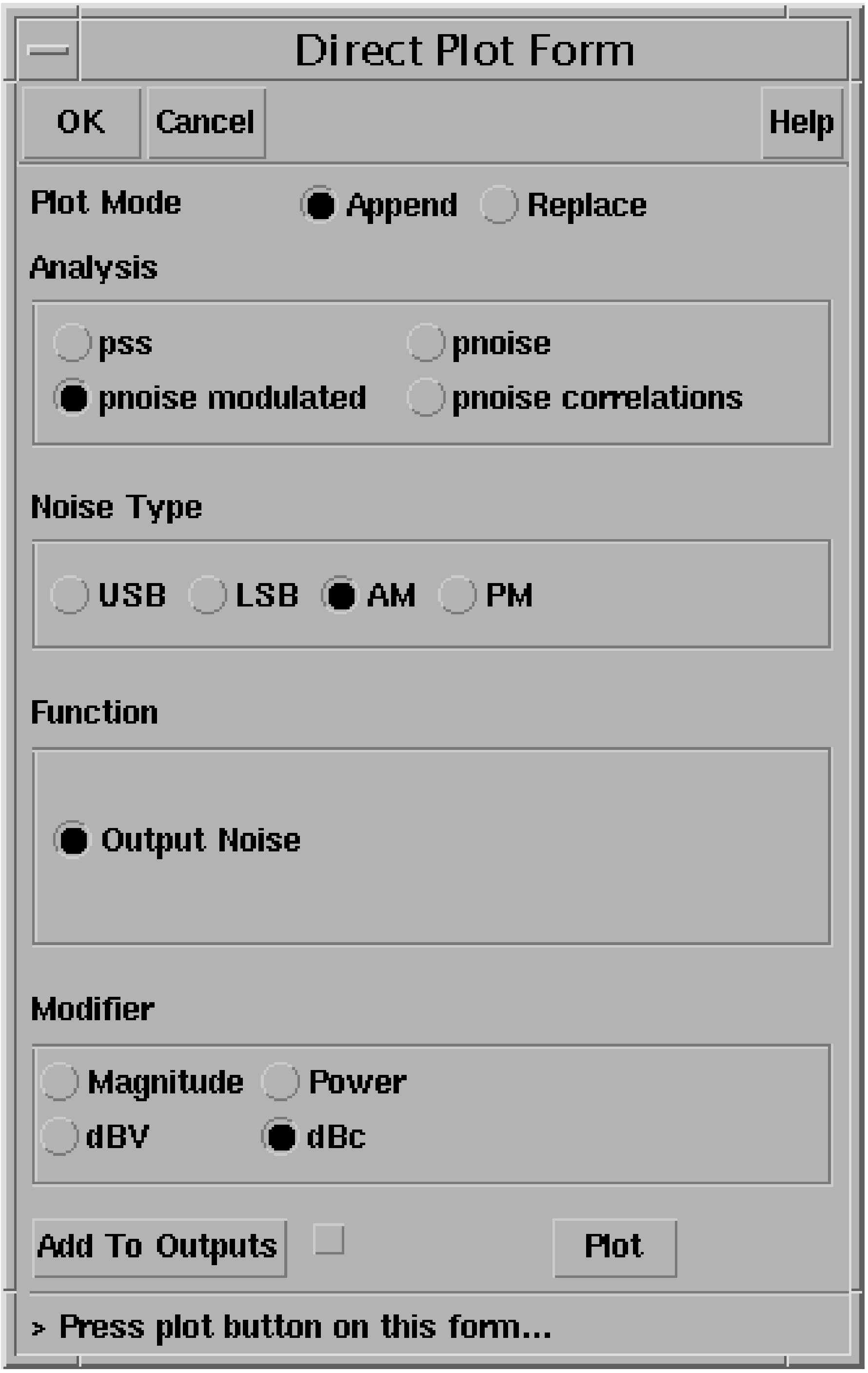
Plot the AM and PM noise in the same Waveform window from the same Direct Plot form.

Start by plotting AM noise:

1. Highlight *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *AM* for *Noise Type*.

   When you highlight *AM* for *Noise Type*, all *Function* selections other than *Output Noise* go away.
4. Keep *dBc* for *Modifier*.

The Direct Plot form appears as follows.

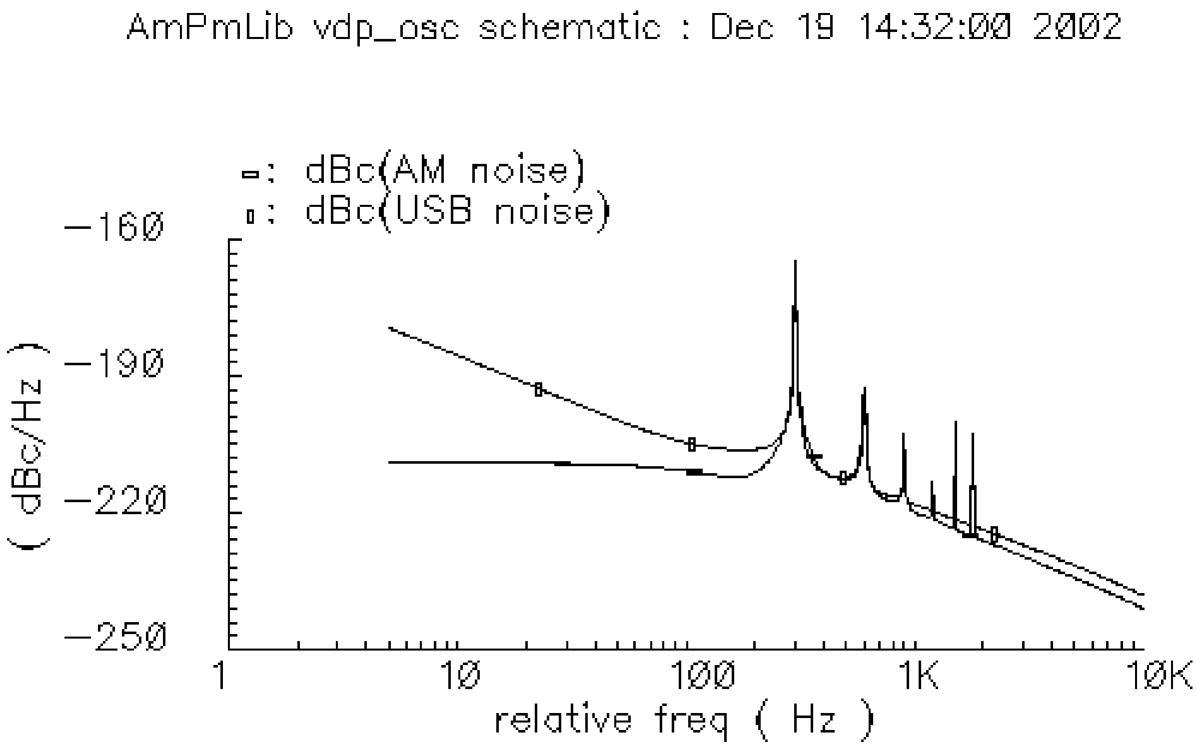


5. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The Waveform window displays the USB and AM noise functions.



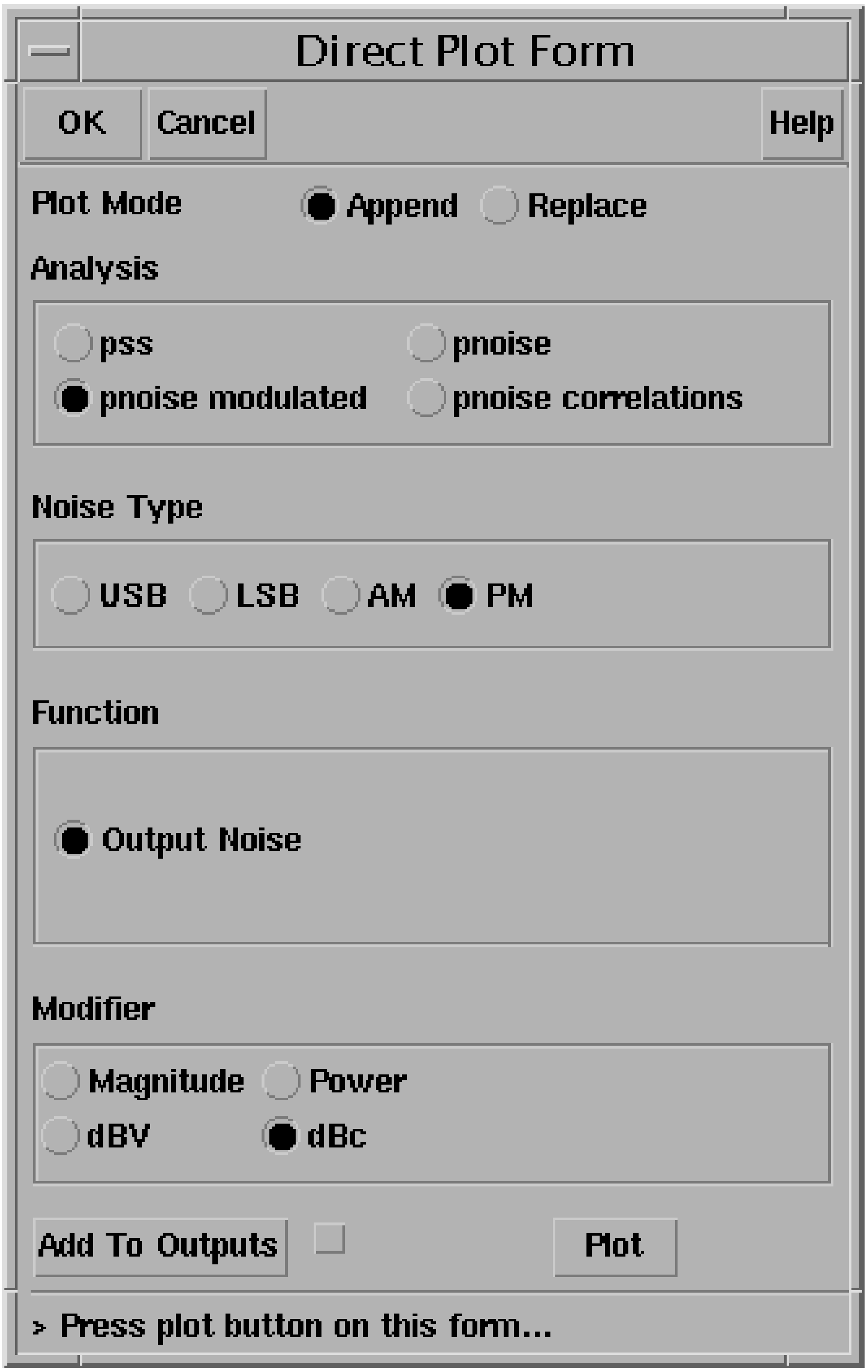
AM noise is added to the Direct Plot form.

Plotting PM Noise

Finally, plot the PM noise in the same Waveform window from the same Direct Plot form.

1. Keep *Append* for *Plot Mode*.
2. Keep *pnoise modulated* for *Analysis*.
3. Highlight *PM* for *Noise Type*.
4. Keep *Output Noise* for *Function*.
5. Keep *dBc* for *Modifier*.

The Direct Plot form appears as follows.

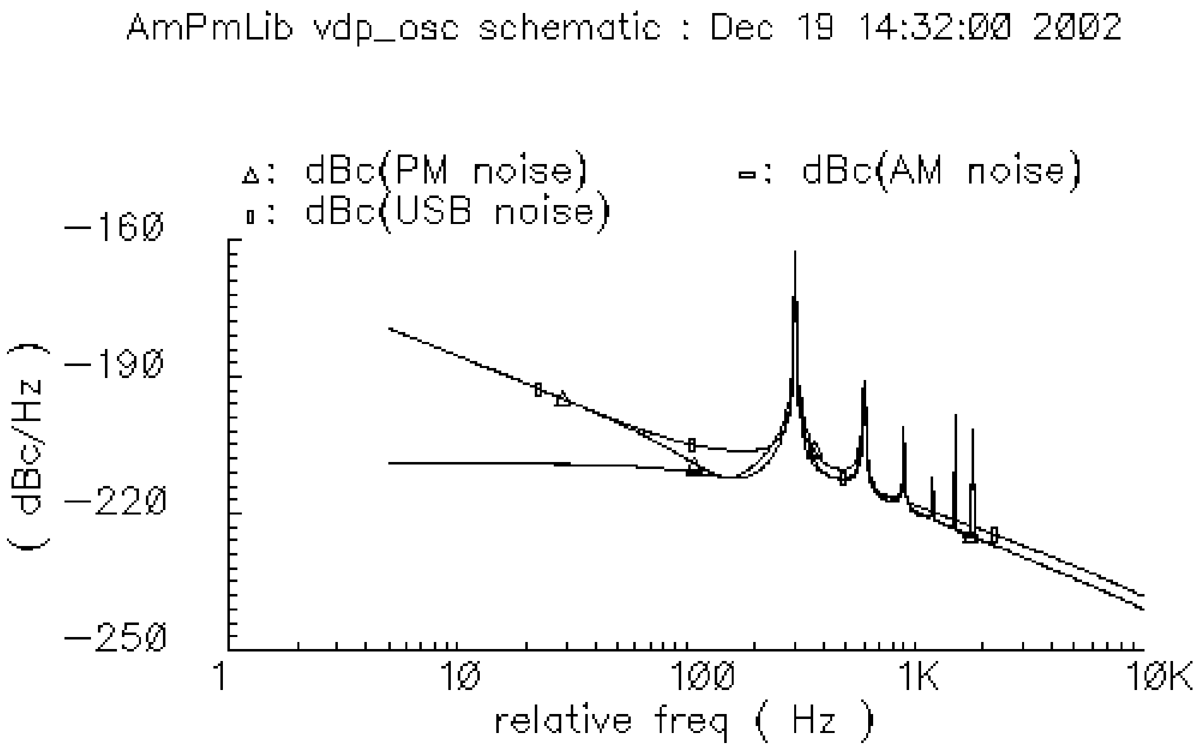


6. Follow the prompt at the bottom of the Direct Plot form

Press plot button on this form...

and click *Plot*.

The Waveform window displays the USB, AM and PM noise functions.



PM noise is added to the Direct Plot form.

Troubleshooting for Oscillator Circuits

If your oscillator simulation fails to converge, try the following techniques to help the SpectreRF simulator find the solution. You can find more information about each of these techniques in the SpectreRF Theory document available in the CDSdoc online documentation library.

- Increase the value of the tstab parameter.
- Be sure that you successfully started the oscillator.

  PSS fails to converge if the circuit cannot sustain an autonomous oscillation. Start the oscillator so that it responds with a signal level between 25 percent and 100 percent of the expected final level. Also, do not "kick" the oscillator so hard that its response is unnecessarily nonlinear. If possible, avoid exciting response modes that are unrelated to the oscillation, especially those associated with longtime constants.
- Improve your estimate of the period.

  Be careful that the period you specify is not too short. Overestimating the period is better than underestimating the period.
- When possible, use *gear2only* as the value of the method parameter.
- Increase the value of the maxperiods parameter to increase the maximum number of iterations for the shooting methods.
- If the shooting iteration approaches convergence and then fails, increase the value of the steadyratio parameter.

  The *steadyratio* parameter controls how much the final solution can deviate from being periodic.
- Change the value of the tolerance parameter.

  Changing the value of the *tolerance* parameter does not change the accuracy of the final solution, but it increases the chance that the simulation converges.
- Tighten the normal simulation tolerances by changing the values of the maxstep, reltol, lteratio, and errpreset parameters. Avoid setting *errpreset* to *liberal*.

6

Simulating Low-Noise Amplifiers

To use this chapter, you must be familiar with the SpectreRF simulator analyses as well as know about low-noise amplifier design. For more information about the SpectreRF simulator analyses, see Chapter 1, “SpectreRF Analyses.”

Analyses and Measurement Examples in this Chapter

This chapter shows you how to simulate and plot the following:

- Voltage Gain
- Output Voltage Distribution
- S-Parameter Analysis, including the Voltage Standing Wave Ratio
- S-Parameter Noise Analysis, including
  - Noise Figure and Minimum Noise Figure
  - Equivalent Noise Resistance
  - Load and Source Stability Circles
  - Noise Circles
- Noise Figure Calculations with the Pnoise Analysis
- The 1dB Compression Point
- The Third-Order Intercept Point
- Conversion Gain and Power Supply Rejection with the Periodic Transfer Function Analysis

Simulating the lnaSimple Example

Before you start, perform the setup procedures described in Chapter 3, "Setting Up for the Examples."

Opening the lnaSimple Circuit in the Schematic Window

1. Interpreter Window (CIW), choose *File – Open*.

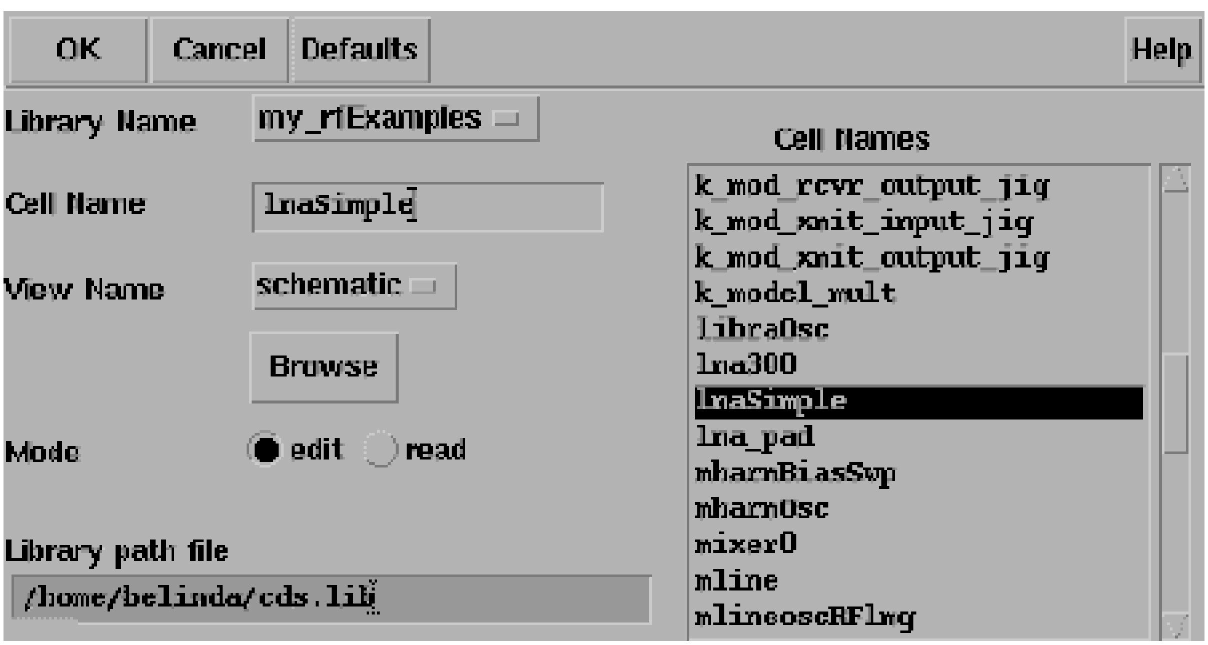
   The Open File form appears.

2. Choose *my_rfExamples* for *Library Name*. This is the editable copy of the *rfExamples* library you created following the directions given in Chapter 3, "Setting Up for the Examples."

3. Choose *lnaSimple* in the *Cell Names* list box.

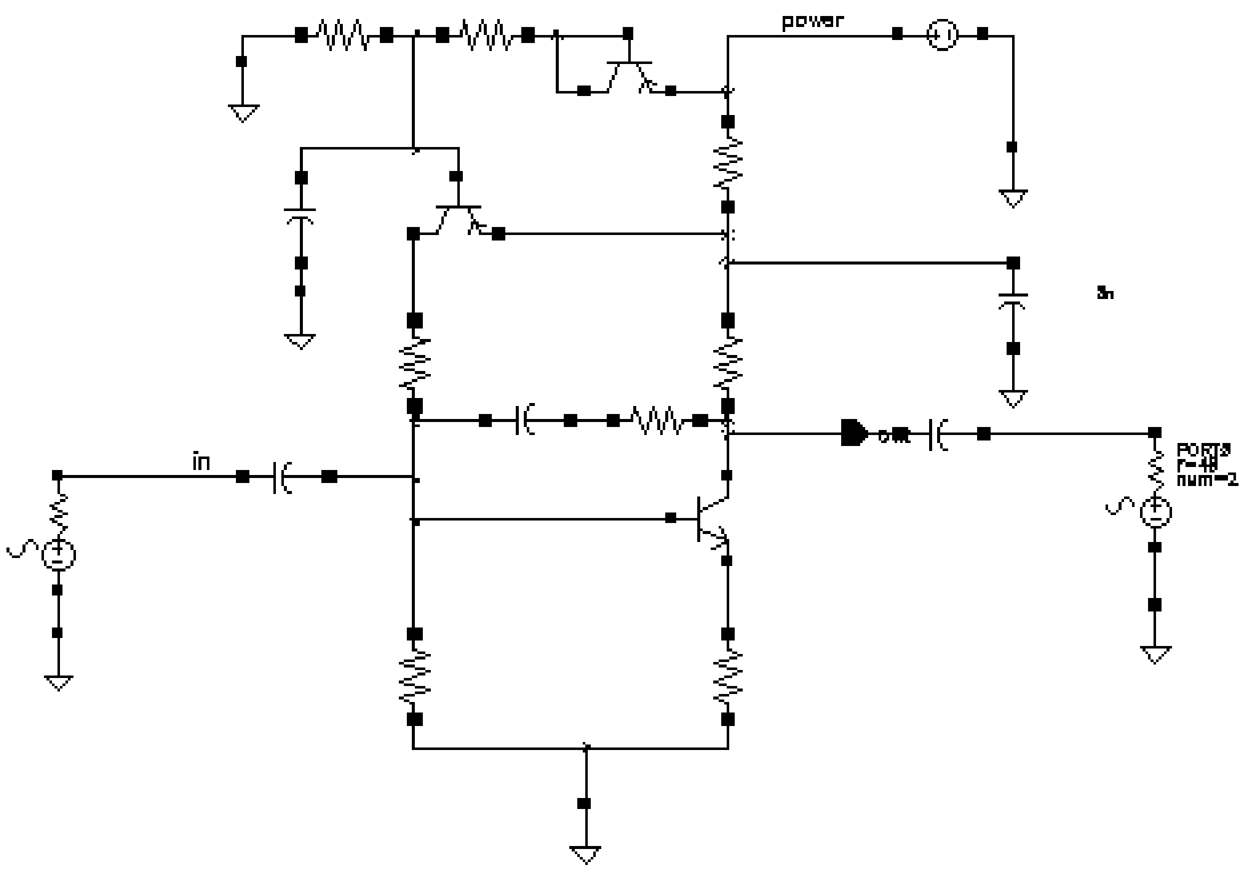
4. Choose *schematic* for *View Name*.

   The filled-in form appears like the one below.



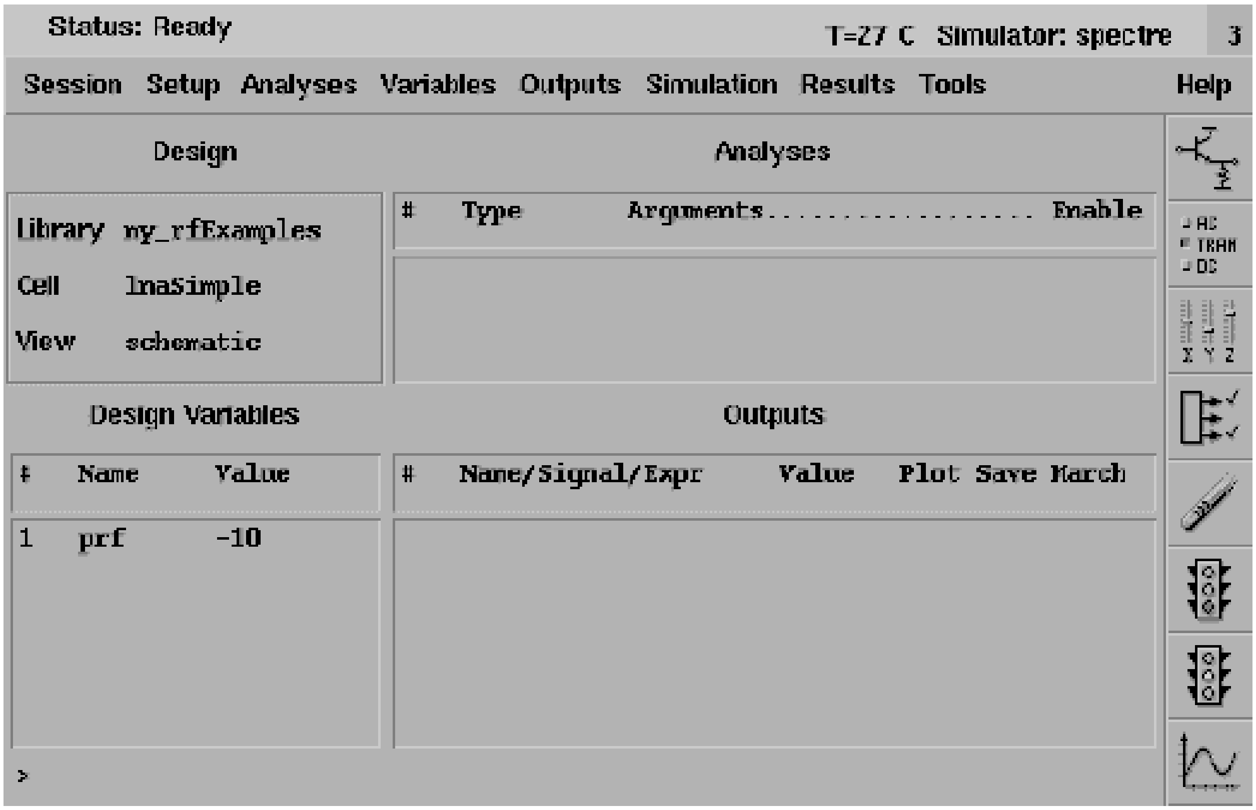
5. In the Open File form, click *OK*.

The Schematic window now shows the *lnaSimple* schematic.



6. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window appears.



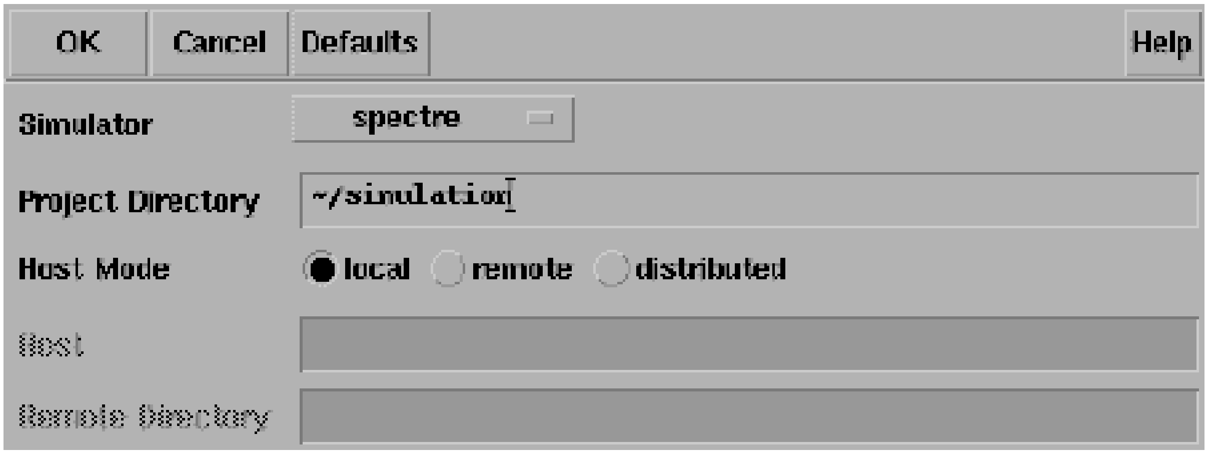
Choosing Simulator Options

1. In the Simulation window, choose *Setup – Simulator/Directory/Host*.

   The Choosing Simulator/Directory/Host form appears.

2. Choose *spectre* for *Simulator*.
3. Type the name of the project directory, if necessary.
4. Highlight the *local* or the *remote* button to specify the *Host Mode*.
5. For remote simulation, type in the name of the host machine and the remote directory in the appropriate fields.

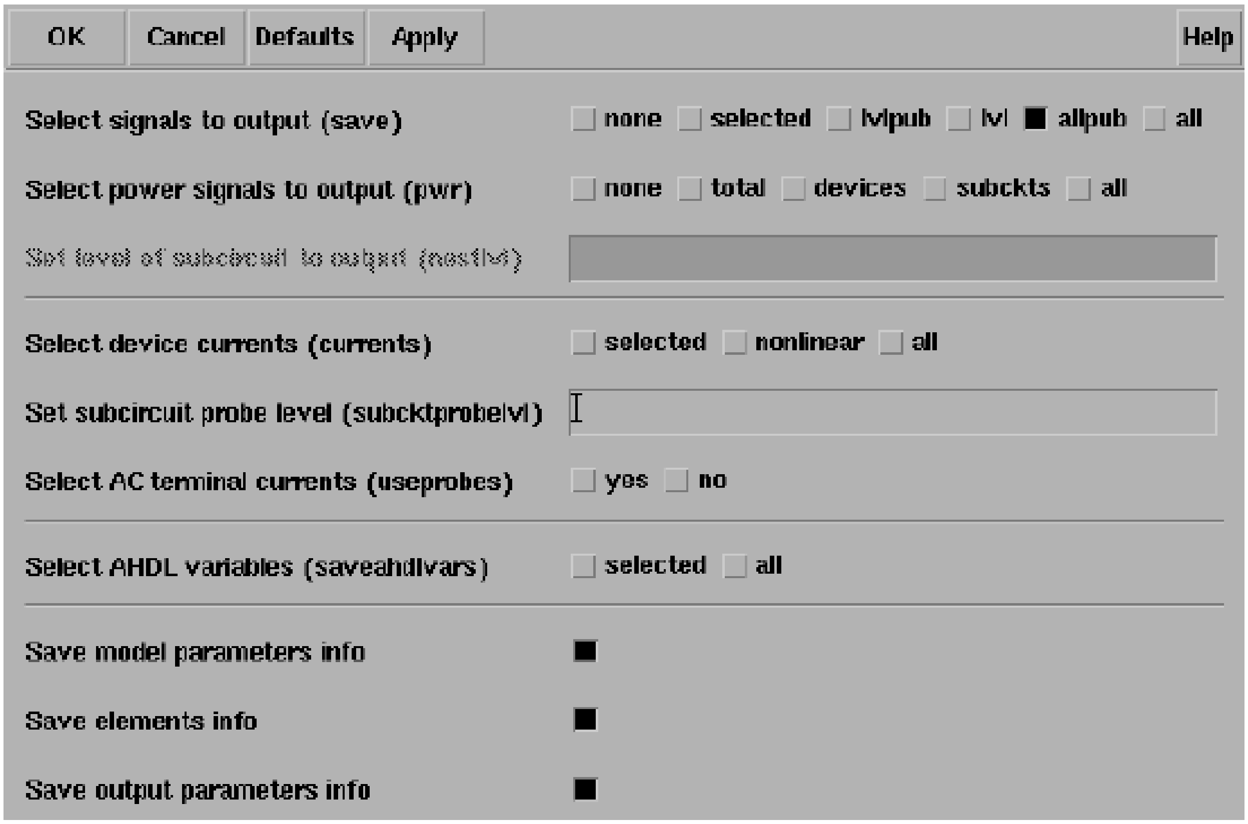
The completed form looks like this.



6. In the Choosing Simulator/Directory/Host form, click *OK*.
7. In the Simulation window, choose *Outputs – Save All*.

   The Save Options form appears.

8. In the *Select signals to output* section, be sure *allpub* is highlighted.



Setting Up Model Libraries

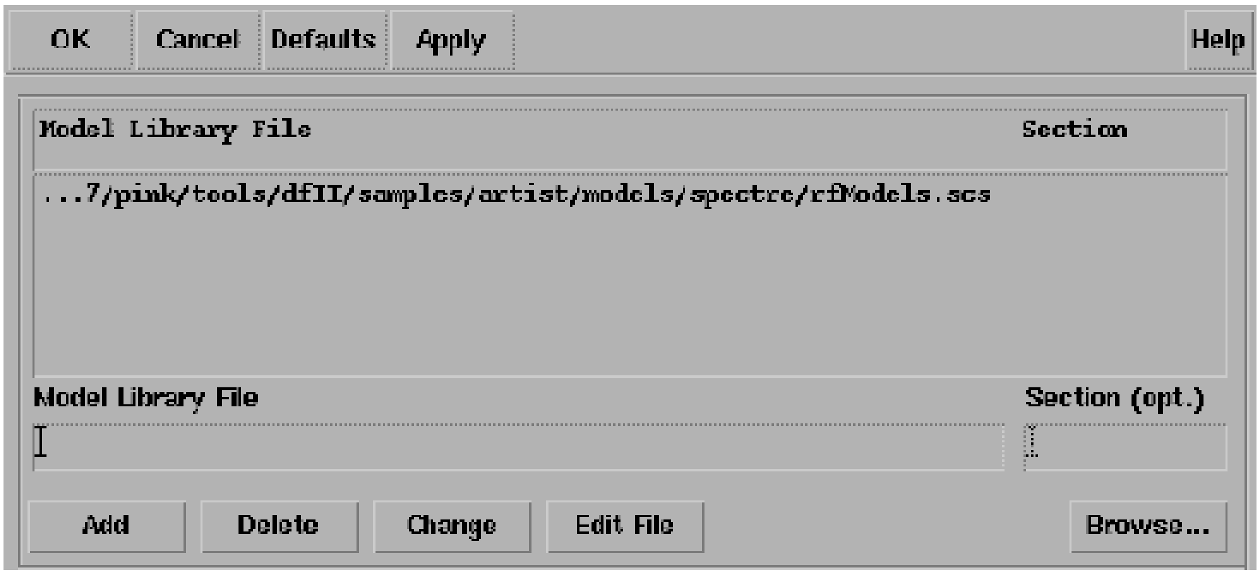
1. In the Simulation window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.

1. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs`.

2. In the Model Library Setup form, click on *Add*.

The completed form looks like this.



3. In the Model Library Setup form, click *OK*.

Calculating Voltage Gain with PSS

The most important characteristic of an amplifier is its gain. In the following example, a PSS analysis determines the voltage gain of the low noise amplifier.

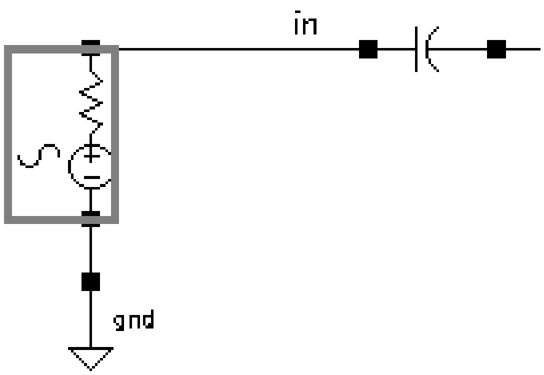
Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file
4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click on the RF voltage source.



2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form for the port appears.

CDF Parameter	Value
Frequency name	frf
Second frequency name	
Noise file name	
Number of noise/freq pairs	0
Resistance	40 Ohms
Port number	1
DC voltage	
Source type	sine
Delay time	
Sine DC level	

3. If necessary, in the *Source Type* field type `sine` and click *OK* in the Edit Object Properties form.

4. In the Schematic window, choose *Design – Check and Save*.

Setting Up the PSS Analysis

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

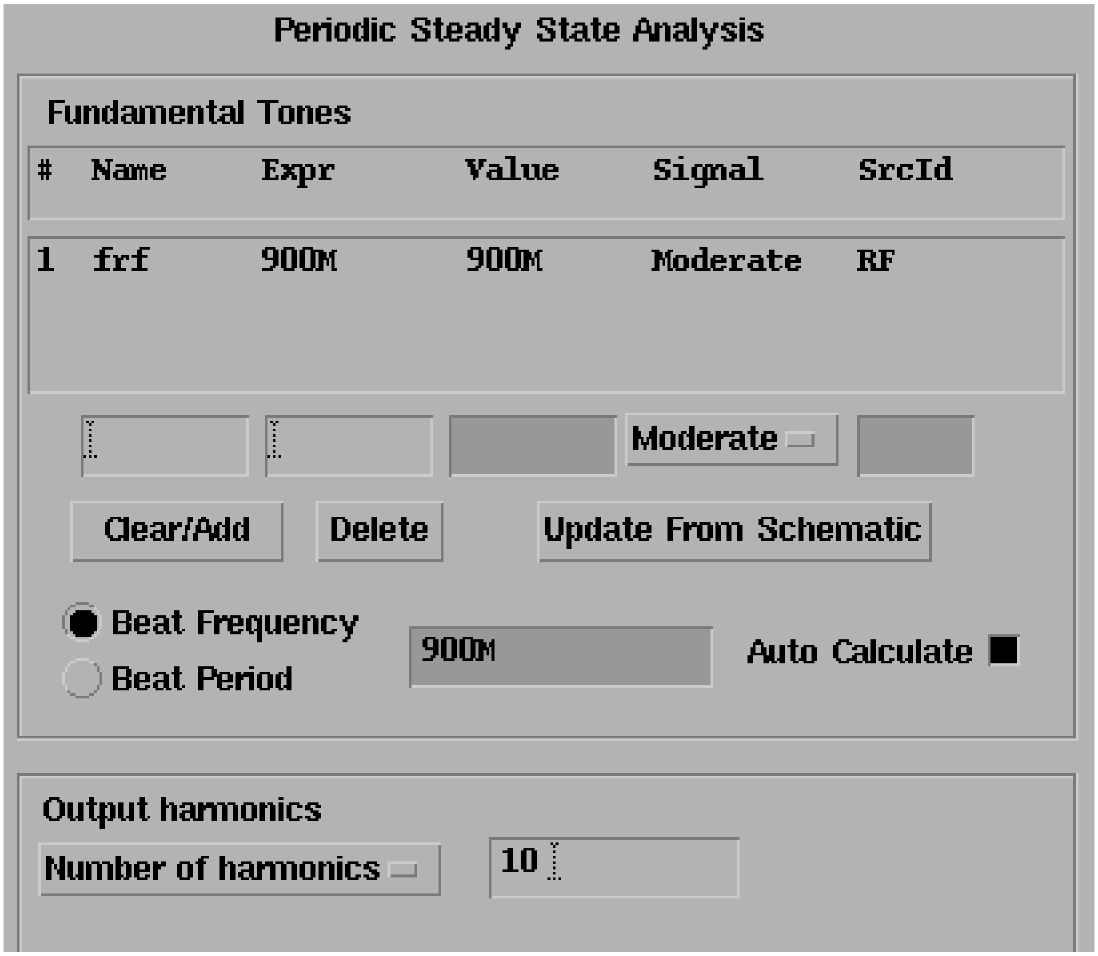
   The form changes to let you specify data for the PSS analysis.

   In the *Fundamental Tones* area, the *Beat Frequency* button is highlighted by default. Highlight the *Auto Calculate* button.

   The fundamental frequency for this analysis, which is `900M`, appears in the *Beat Frequency* field.

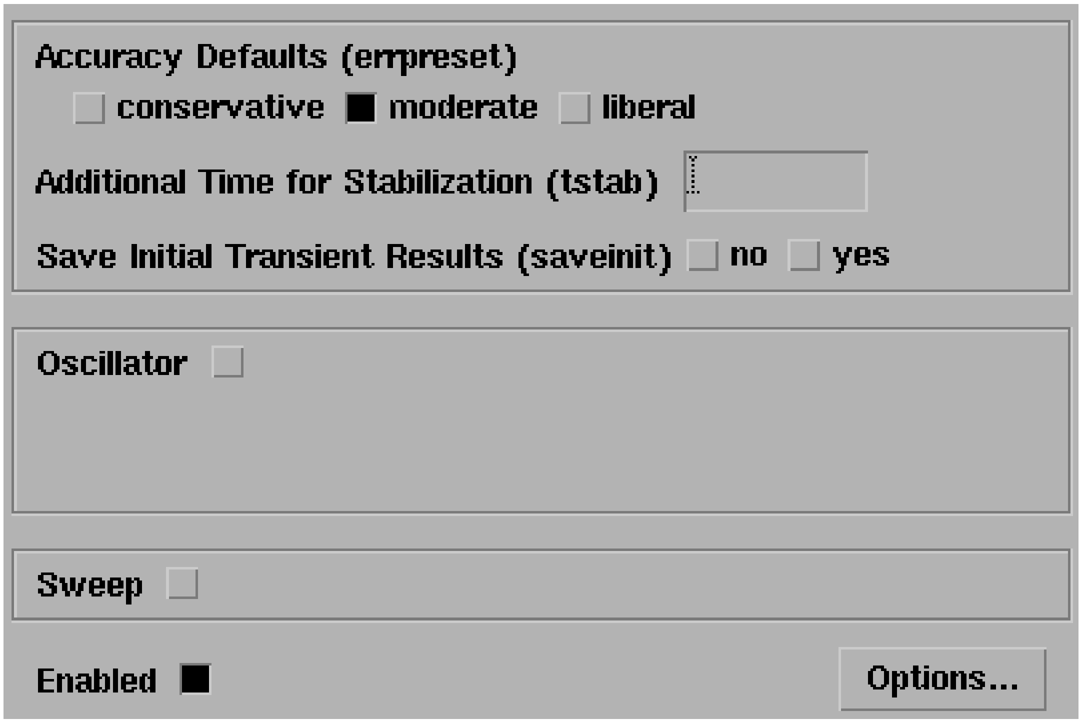
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type a reasonable value, such as 10, in the *Number of harmonics* field. (You can choose a different number.)

The top of the PSS Choosing Analysis form looks like this.



4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
5. Highlight *Enabled*, if necessary.

The bottom of the PSS Choosing Analysis form looks like this.



6. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the PSS analysis, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the output log file to be sure the simulation is completed successfully.

Plotting Voltage Gain

➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

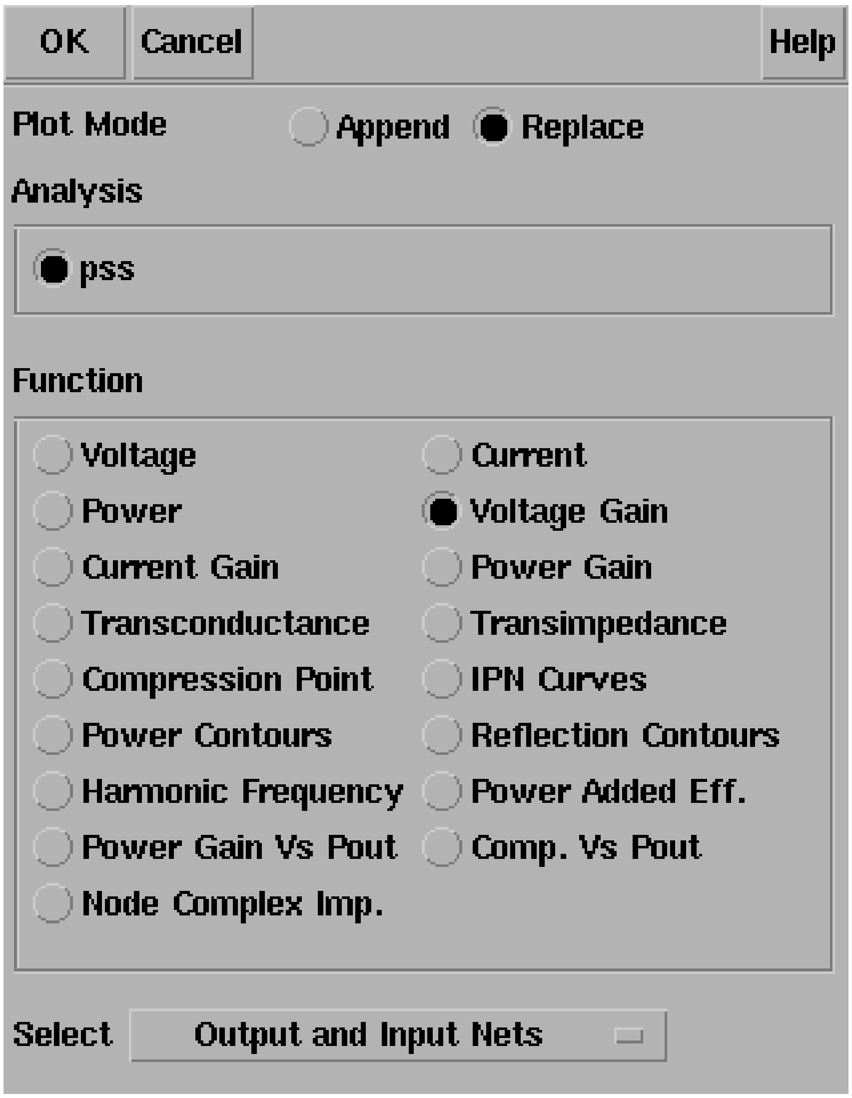
The Waveform window and the Direct Plot form appear.

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pss* for *Analysis Type*.

3. Highlight *Voltage Gain* for *Function*.

The top of the Direct Plot form appears below.



The Direct Plot form changes to reflect your choice.

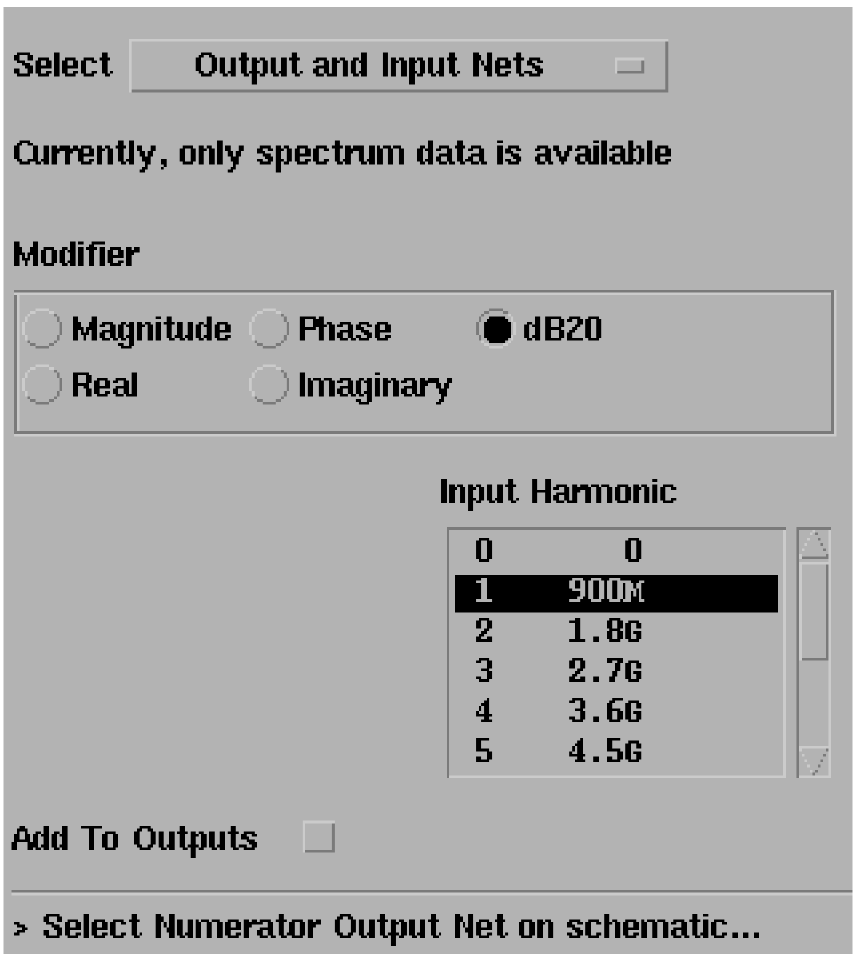
❑ Notice that the *Select* cyclic field changes to display *Output and Input nets*.

❑ Notice the message at the bottom of the form *Select Input Harmonic on this form….*

4. Highlight *dB20* for *Modifier*.

5. Highlight *900M* in the *Input Harmonic* list box.

The bottom of the Direct Plot form appears below.

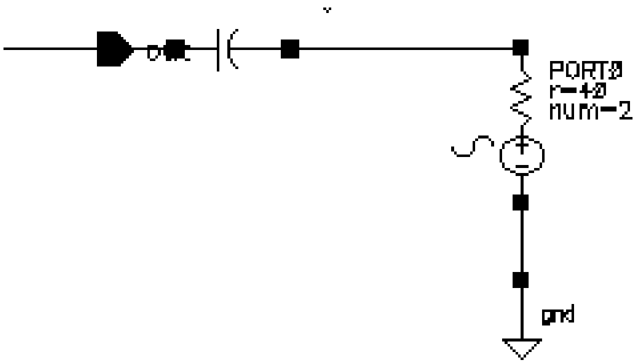


Calculate the voltage gain as *Vout/Vin* by following the instructions at the bottom of the Direct Plot form as illustrated in the next two steps—click first on the output net then click on the input net.

6. Following the message at the bottom of the form,

Select Numerator Output Net on schematic...

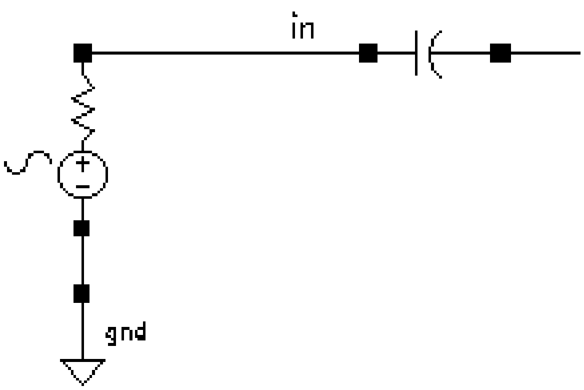
Select the `Vout` output net.



7. Following the message at the bottom of the form,

Select Denominator Input Net on schematic...

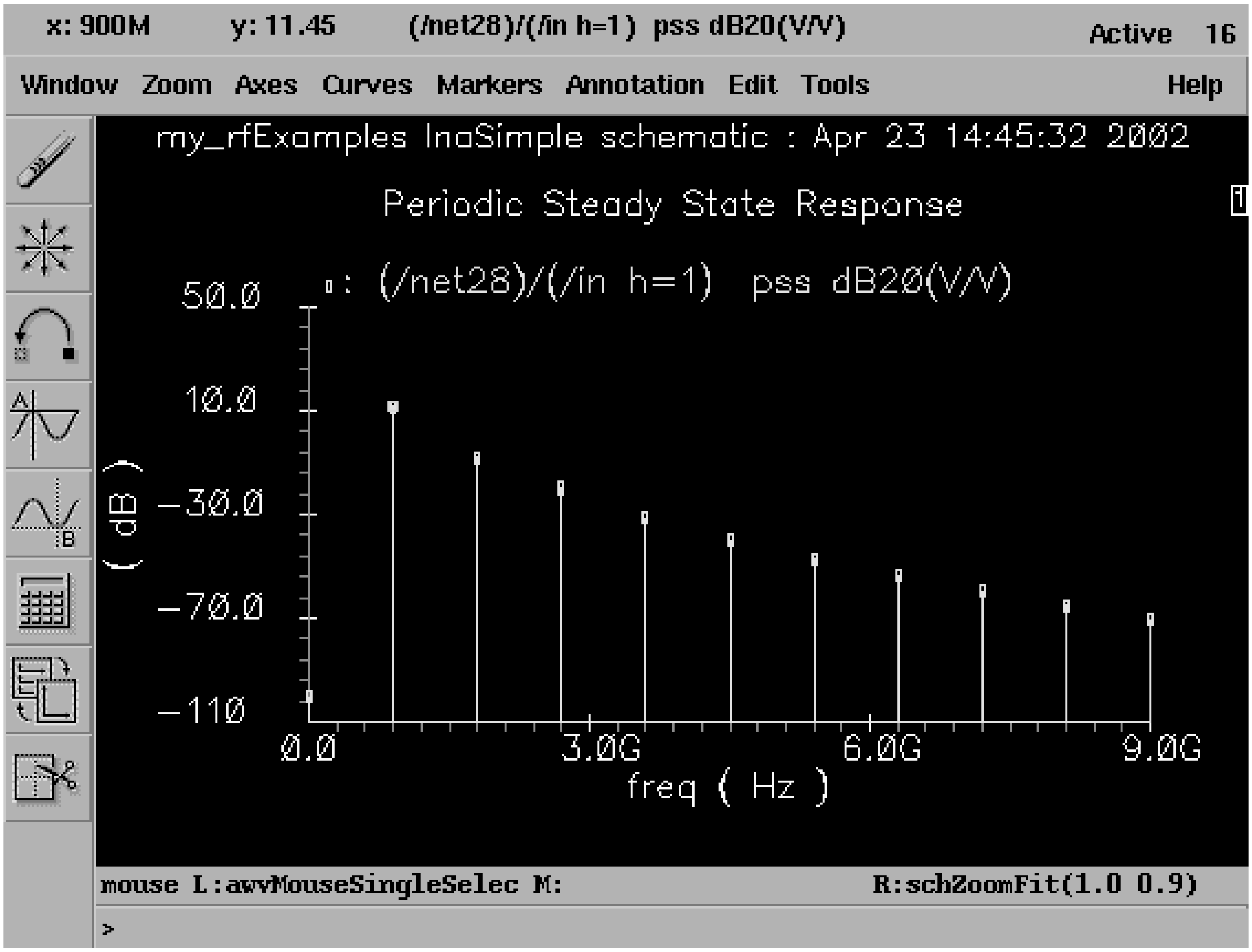
Select the `Vin` input net at the RF source.



8. The voltage gain plot appears in waveform window.

To display the voltage gain value, in the Waveform window, place your cursor at the top of the line representing the 900 MHz fundamental (the tallest, leftmost line).

The x and y axis values, which are displayed in the top left corner of the Waveform window, show that the voltage gain at 900 MHz is 11.45.



Calculating Output Voltage Distribution with PSS

The output voltage distribution of an amplifier is important in analyzing intermodulation distortion when the two RF input signals are close together. To model this situation, you modify a parameter to make the voltage source component generate two separate frequencies.

Setting Up the Simulation

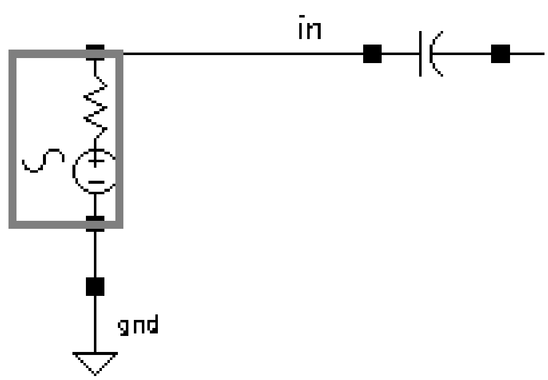
1. If necessary, open the lnaSimple circuit.

2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file
4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

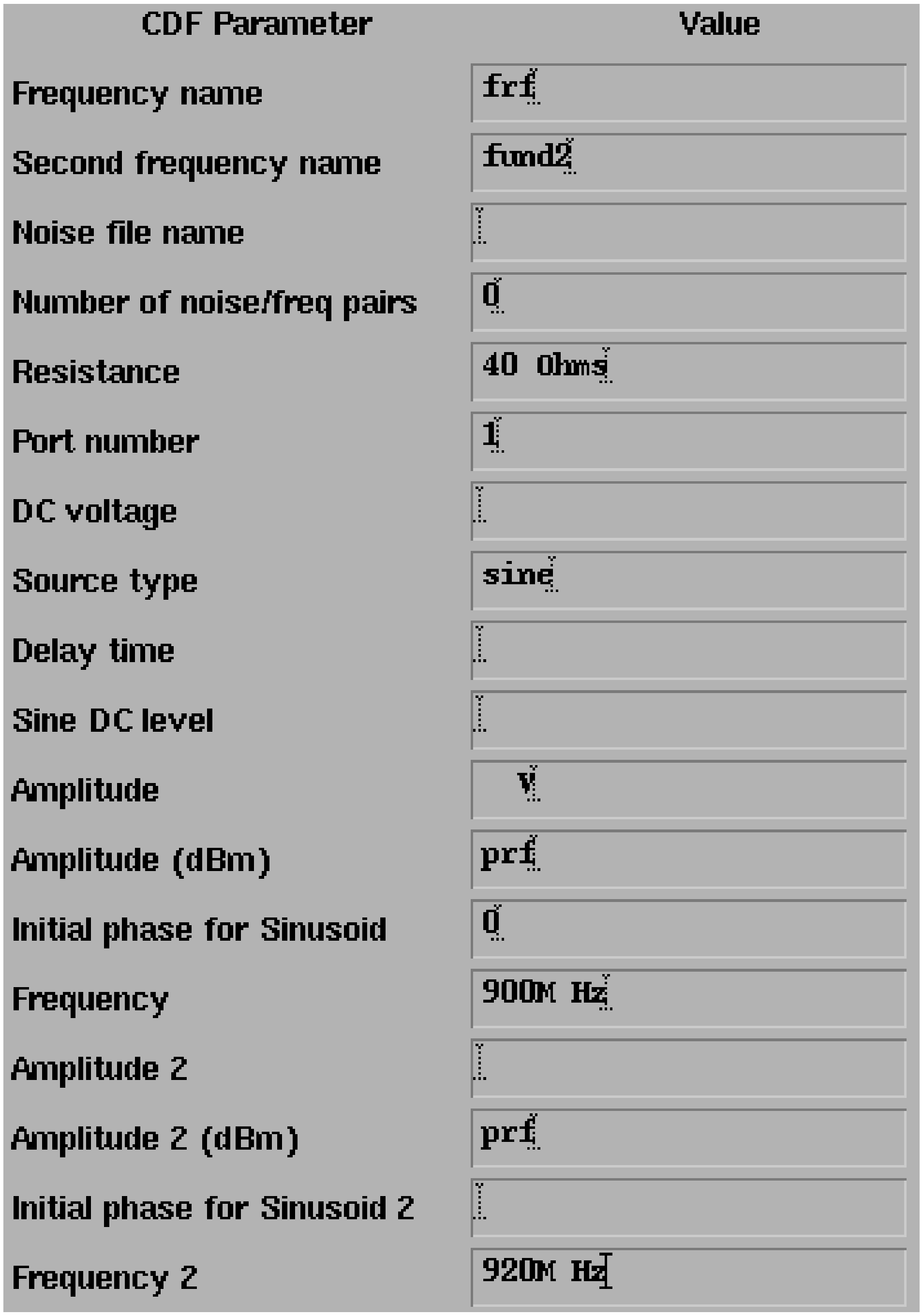
Editing the Schematic

1. In the Schematic window, click on the RF voltage source.



2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.



CDF Parameter	Value
Frequency name	frf
Second frequency name	fund2
Noise file name	
Number of noise/freq pairs	0
Resistance	40 Ohms
Port number	1
DC voltage	
Source type	sine
Delay time	
Sine DC level	
Amplitude	V
Amplitude (dBm)	prf
Initial phase for Sinusoid	0
Frequency	900M Hz
Amplitude 2	
Amplitude 2 (dBm)	prf
Initial phase for Sinusoid 2	
Frequency 2	920M Hz

3. Highlight *Display second sinusoid*.

   The form changes to let you specify data for a second sinusoid.

4. In the *Second frequency name* field, type `fund2,` or any name that you choose.

5. In the *Frequency 2* field, type `920M`.
6. In the *Amplitude 2 (dBm)* field, type `prf`.

   This sets the amplitude of the second tone equal to that of the first tone.
7. In the Edit Object Properties form, click *OK*.
8. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS Analysis

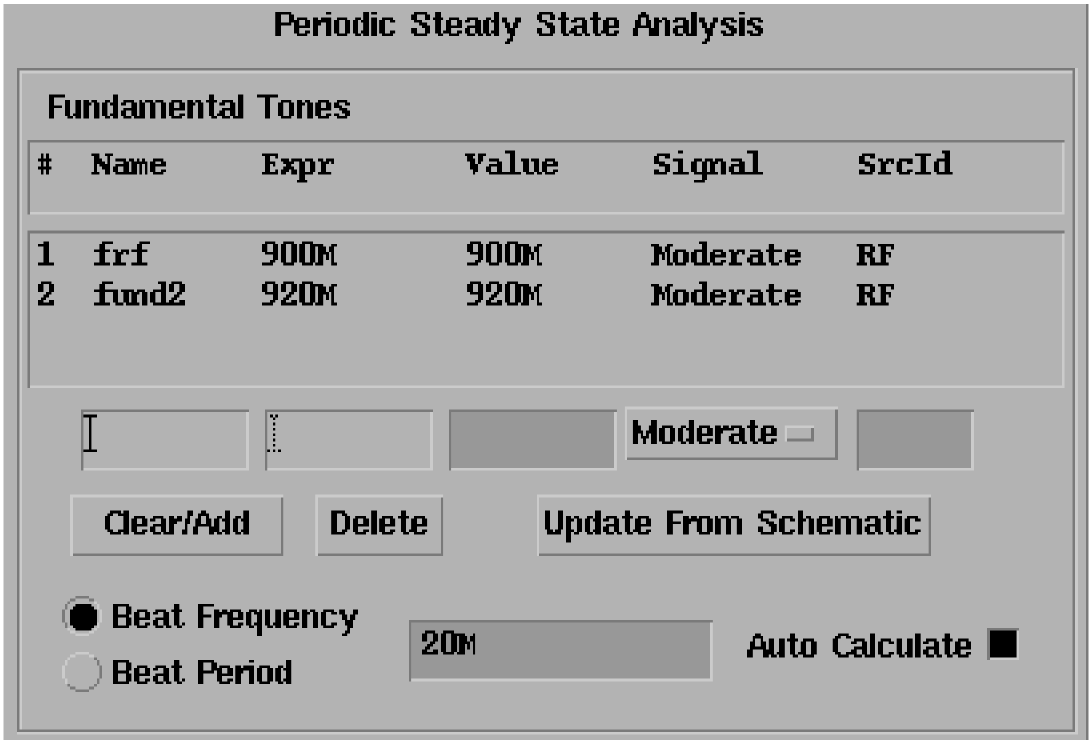
1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. Click on *pss* for the *Analysis* choice.

   The Choosing Analyses form changes to display fields for the PSS analysis.
3. In the *Fundamental Tones* area, highlight the *Auto Calculate* button. The *Beat Frequency* button is highlighted by default.

   The *Beat Frequency* field displays `20 M` because there are two RF tones—one at `900 M` and the second at 920 M.

The top of the Choosing Analysis form appears as follows.



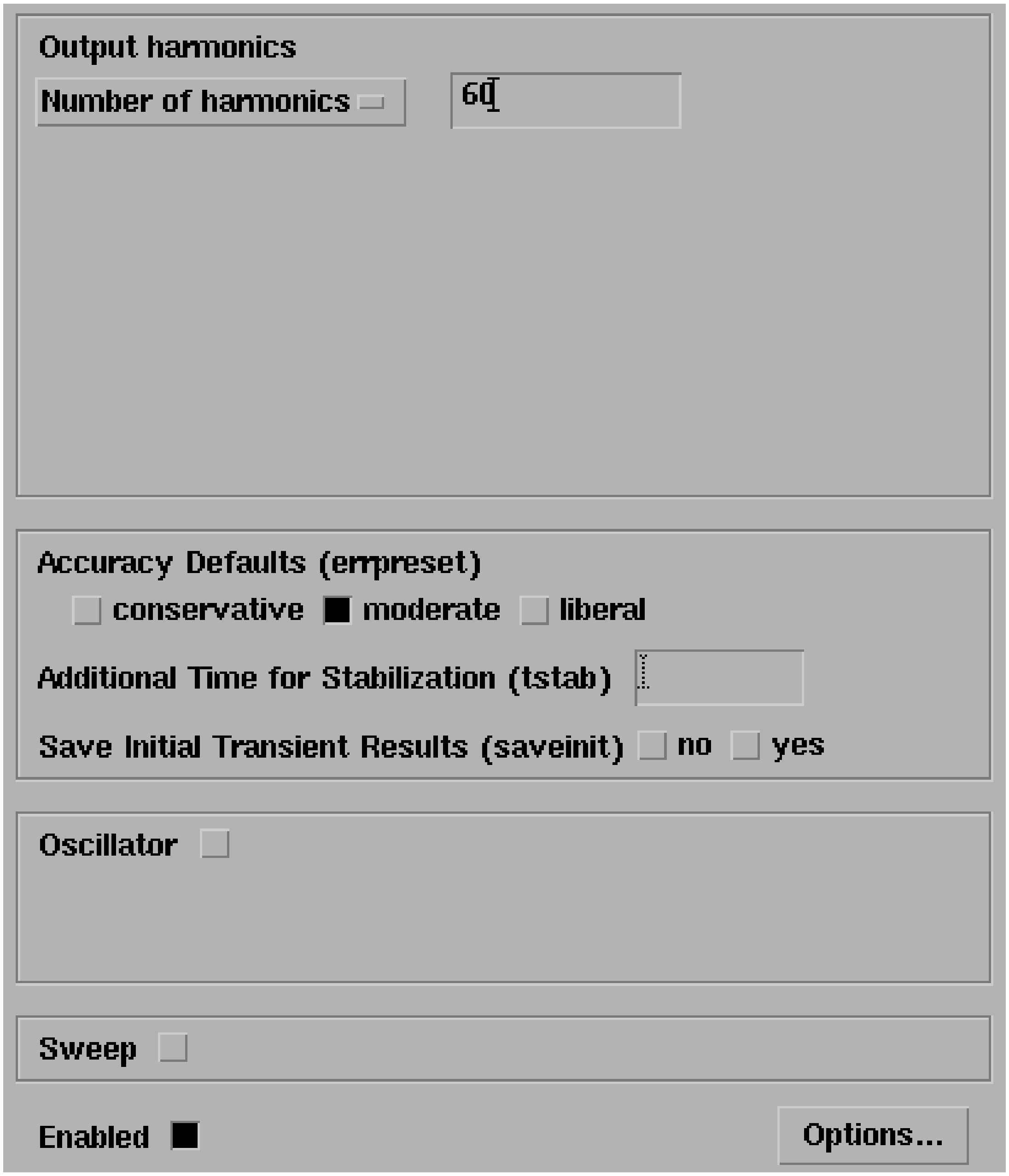
4. In the *Output harmonics* cyclic field, choose *Number of Harmonics* and type `60` in the *Number of Harmonics* field.

   The largest common multiple of 900 Mhz and 920 MHz is 20MHz. Using 60 harmonics extends the results out to 1.2 GHz, but you can might choose another number if you want.

5. Highlight *moderate* as the *Accuracy Defaults* value.

6. Make sure that *Enabled* is highlighted.

The top of the Choosing Analysis form appears as follows.



7. In the Choosing Analysis form, click *OK*.

Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the PSS analysis.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Output Voltage Distribution

➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

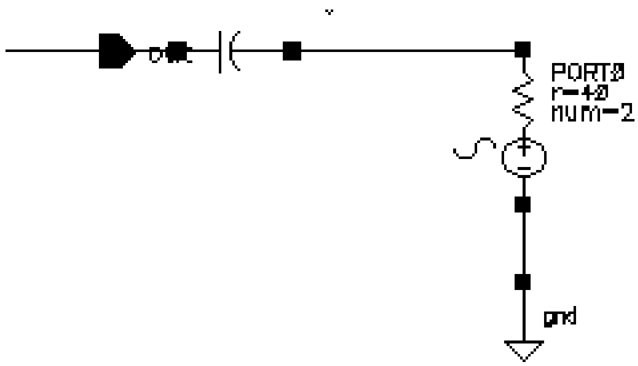
The Waveform window and the Direct Plot form appear.

In the Direct Plot form, do the following:

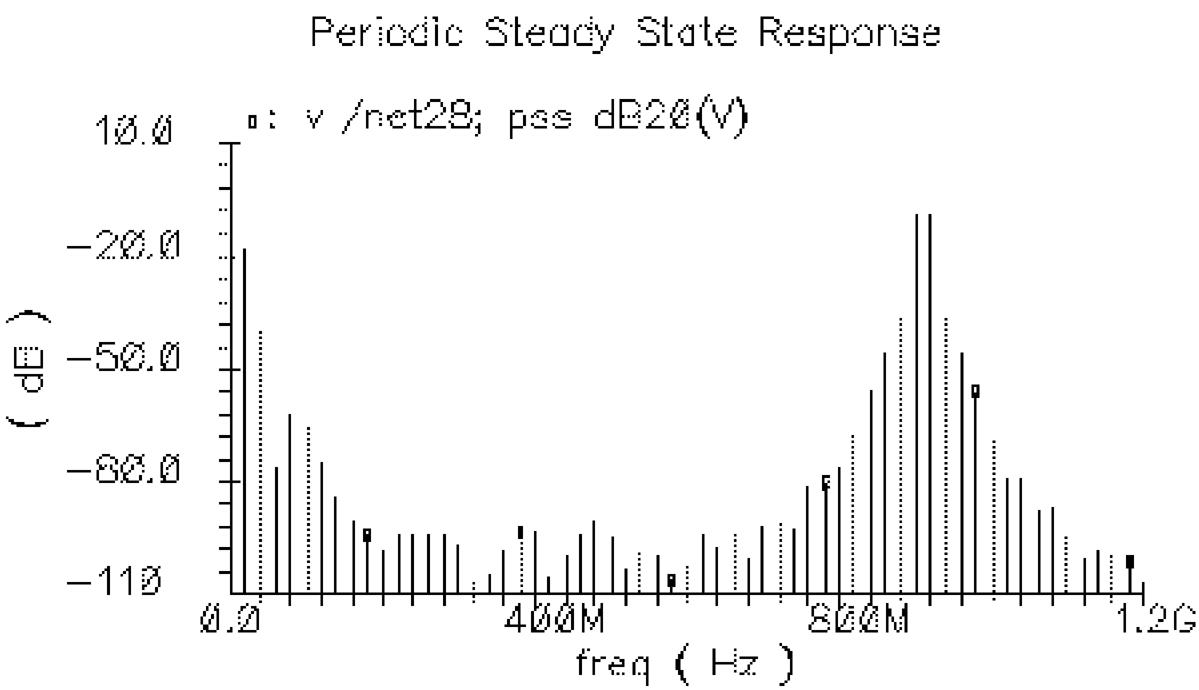
1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pss* for *Analysis*.
3. Highlight *Voltage* for *Function*.
4. Notice that the *Select* cyclic field displays *Net*.
5. Highlight *spectrum* for *Sweep* value.
6. Highlight *peak/rms* for *Signal Level*.
7. Highlight *dB20* for *Modifier*.
8. Following the message at the bottom of the form,

   Select Net on schematic...

Select the `Vout` amplifier output net.



In the Waveform window, you can see spikes at 900 MHz, 920 MHz (-9.3 dB) and a third spike at 20 Mhz. The next strongest harmonics are lower than -36 dB.



S-Parameter Analysis for Low Noise Amplifiers

Before you can perform S-parameter analyses of the low noise amplifier, you need to set up for the example.

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file
4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, select the RF voltage source.
2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.
3. In the Edit Object Properties form, be sure the *Source type* is set to `sine`, delete the *Frequency 2* value, if there is one, and click *OK*.

   The input voltage source now has only one frequency at `900MHz`. *Frequency 2* is zero.
4. In the Schematic window, choose *Design – Check and Save*.

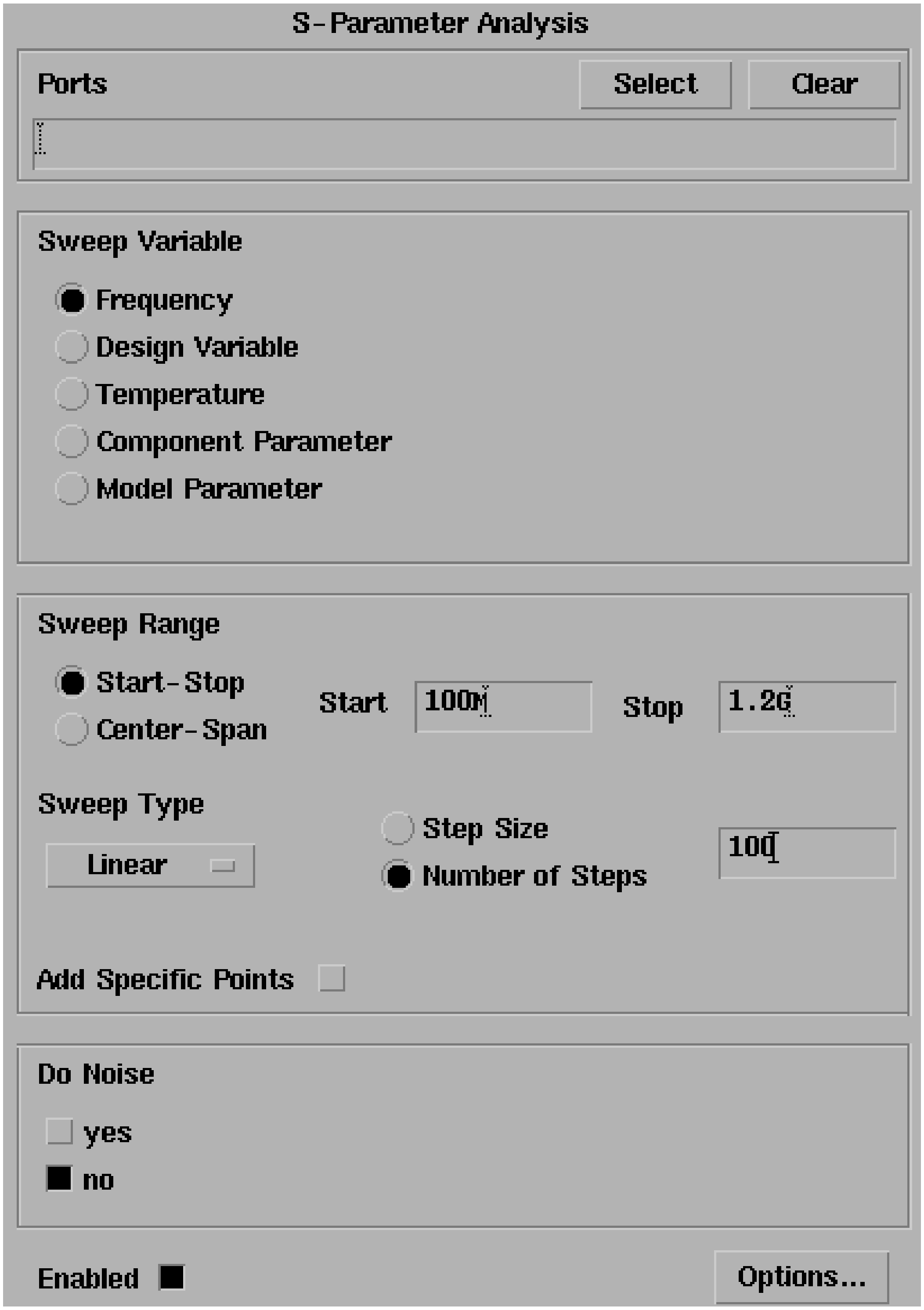
Setting up the S-Parameter Analysis

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click on *sp* for *Analysis*.

   The form changes to let you specify data for an S-parameter analysis.
3. Highlight *Frequency* for *Sweep Variable*.
4. Highlight *Start-Stop* for *Sweep Range* and type `100M` for *Start* and `1.2G` for *Stop*.
5. In the *Sweep Type* cyclic field, choose *Linear.*
6. Click on the *Number of Steps* button and type `100` in the *Number of Steps* field.
7. Highlight *Enabled*.

The filled-in form looks like this.



8. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

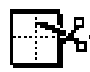
   The outputl log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting S-Parameters

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Waveform window and the Direct Plot form appear.

2. In the Waveform window, click on the add subwindow icon .

   A subwindow is added to the Waveform window, which is divided vertically into two windows.

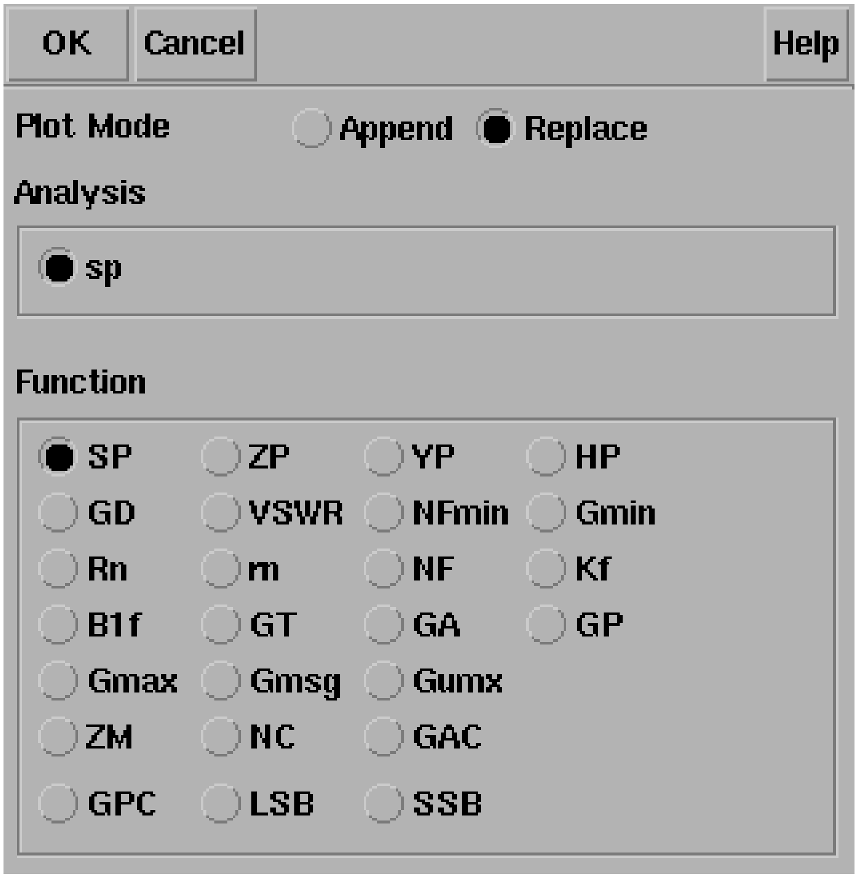
3. To choose a subwindow for a plot, click on the small number in a rectangle at the top right corner of each subwindow. 

   Note: You can also divide the Waveform window into horizontal strips by highlighting *Axes – To Strip* in the Waveform window.

To plot the S11 and S22 values, do the following in the Direct Plot form and the Waveform window.

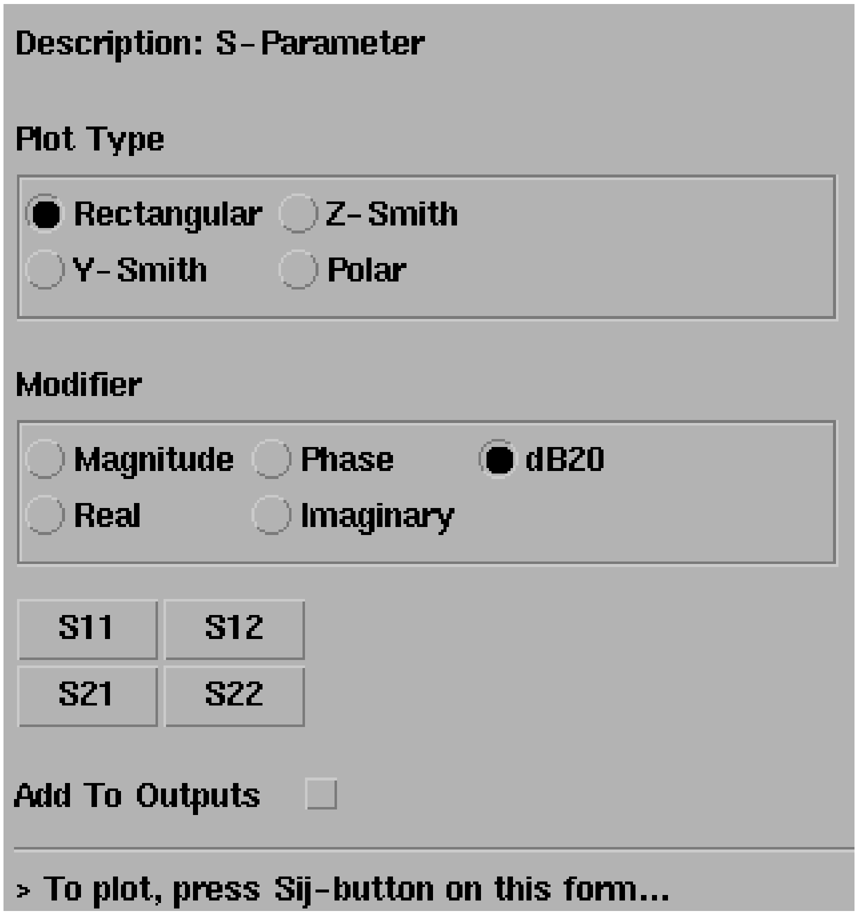
1. Highlight *Replace* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *SP* for *Function*.

The top of the Direct Plot form looks like this.



4. Highlight *Rectangular* for *Plot Type*.
5. Highlight *dB20* for *Modifier*.

The bottom of the SP Direct Plot form looks like this.



6. Following the message at the bottom of the form,

To plot, press Sij-button on this form...

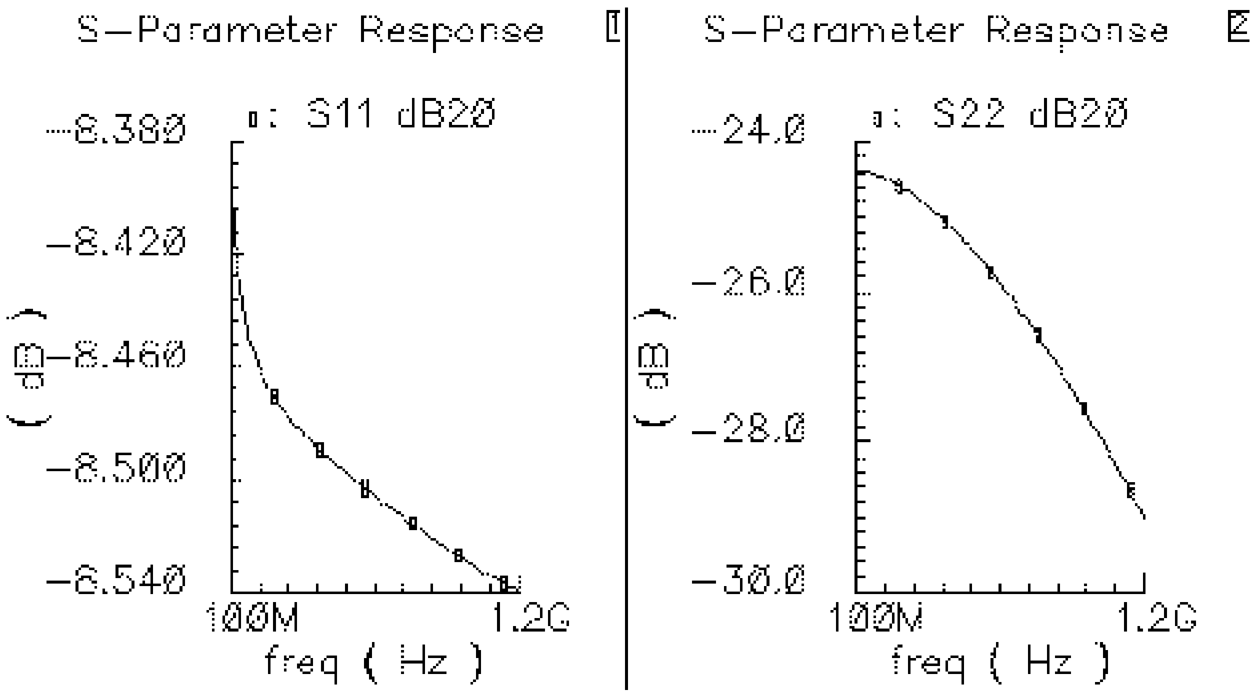
Select the S-parameters to plot.

7. In the Waveform window, click *[1]* to select the left window. In the Direct Plot form click *S11*.

S11 is plotted in the left half of the Waveform window.

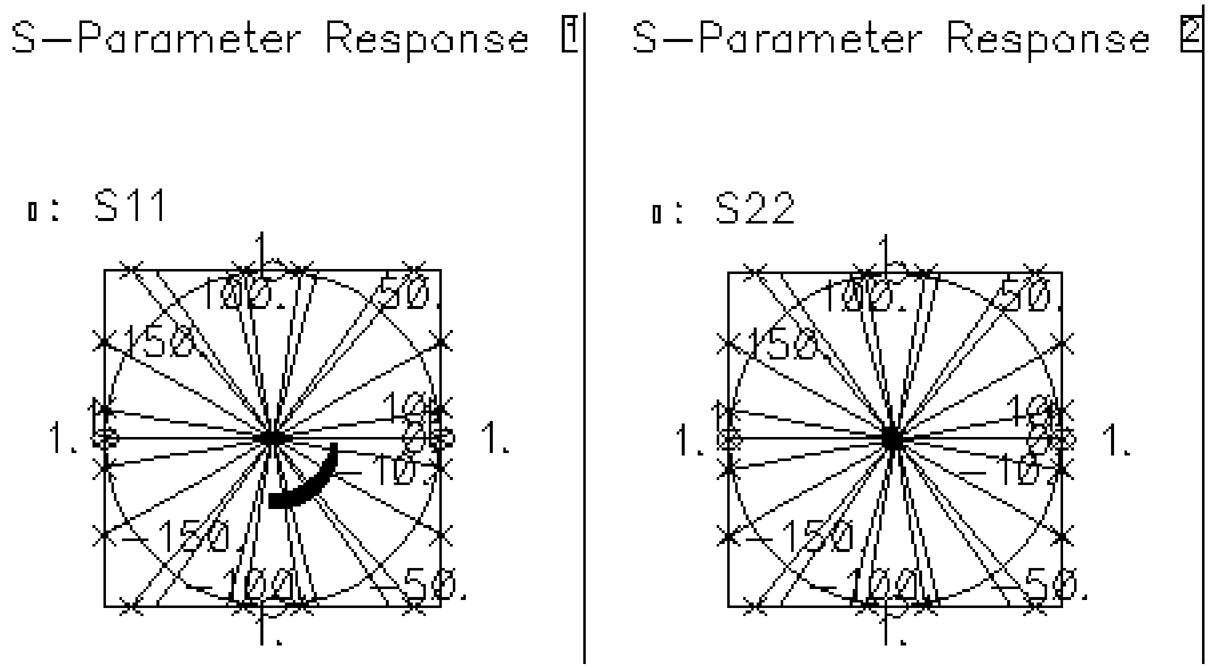
8. In the Waveform window, click *[2]* to select the right window. In the Direct Plot form click *S22*.

S22 is plotted in the right half of the Waveform window.

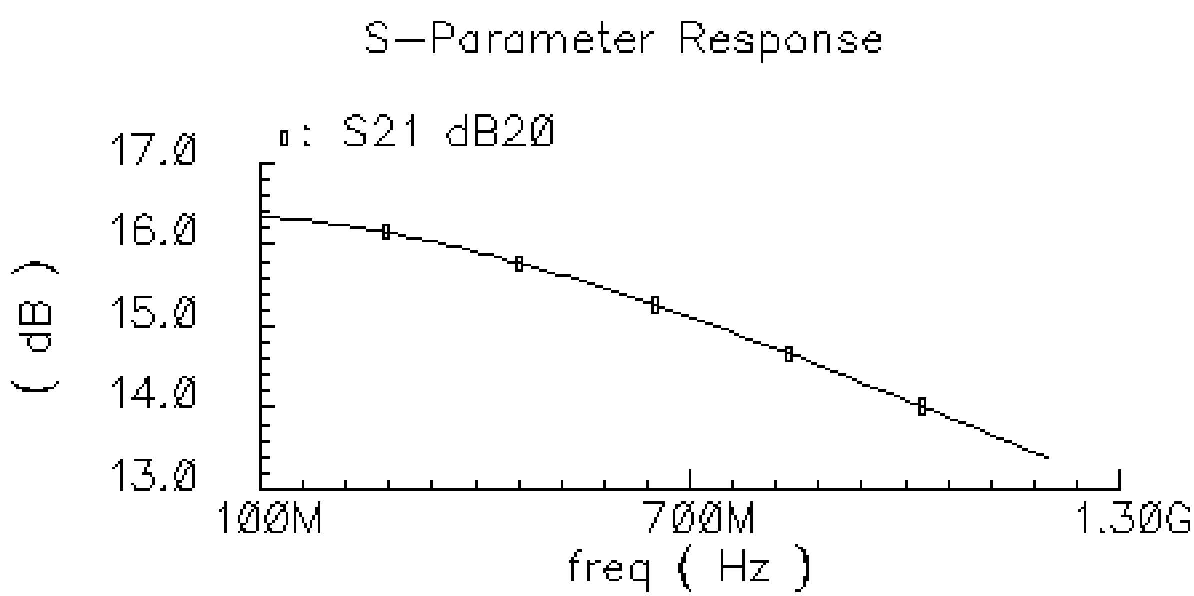


The values of S11 and S22 measure whether input and output are matched.

You can choose among several different plot types in the Direct Plot form. The following plots show S11 and S22 with the *Plot Type* set to *Polar*.

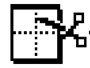


You can also choose S12 and S21 for plotting in the Direct Plot form. S12 is the reverse gain and a measure of isolation.



Plotting the Voltage Standing Wave Ratio

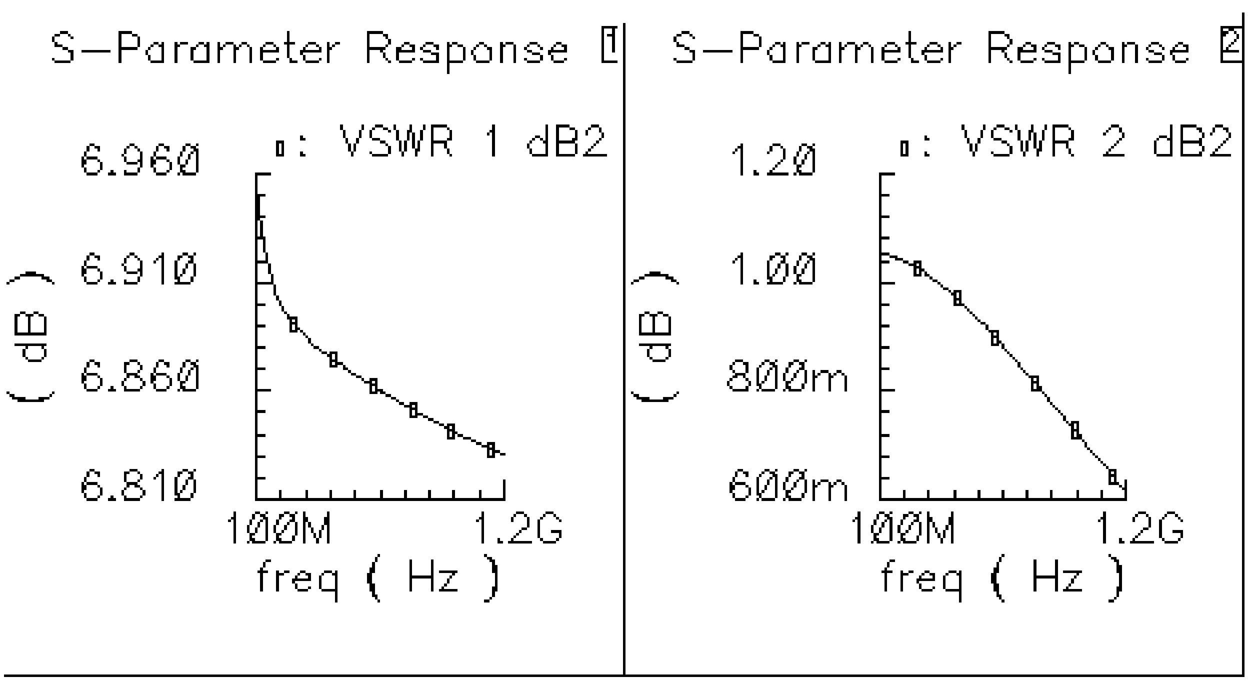
In the Direct Plot form and the Waveform window, do the following.

1. In the Waveform window, click on the add subwindow icon . to create 2 side-by-side panes.
2. In the Direct Plot form, highlight *Replace* for *Plot Mode*.
3. In the Direct Plot form, highlight *sp* for *Analysis*.
4. In the Direct Plot form, highlight *VSWR* for *Function*.
5. In the Direct Plot form, highlight *dB20* for *Modifier*.
6. In the Waveform window, click *[1]* to select the left window. In the Direct Plot form click *VSWR1*.

   VSWR1 is plotted in the left half of the Waveform window.
7. In the Waveform window, click *[2]* to select the right window. In the Direct Plot form click *VSWR2*.

   VSWR2 is plotted in the right half of the Waveform window.

   The waveform appears like the one below.



Linear Two-Port Noise Analysis with S-Parameters

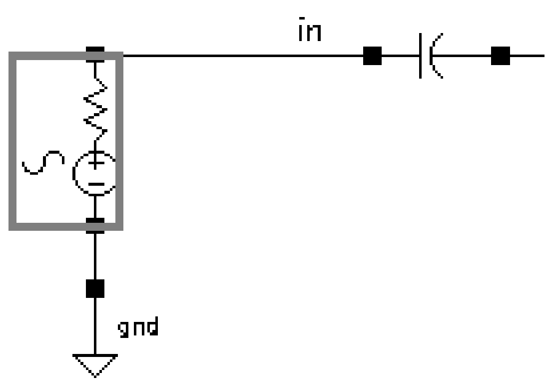
Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file
4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

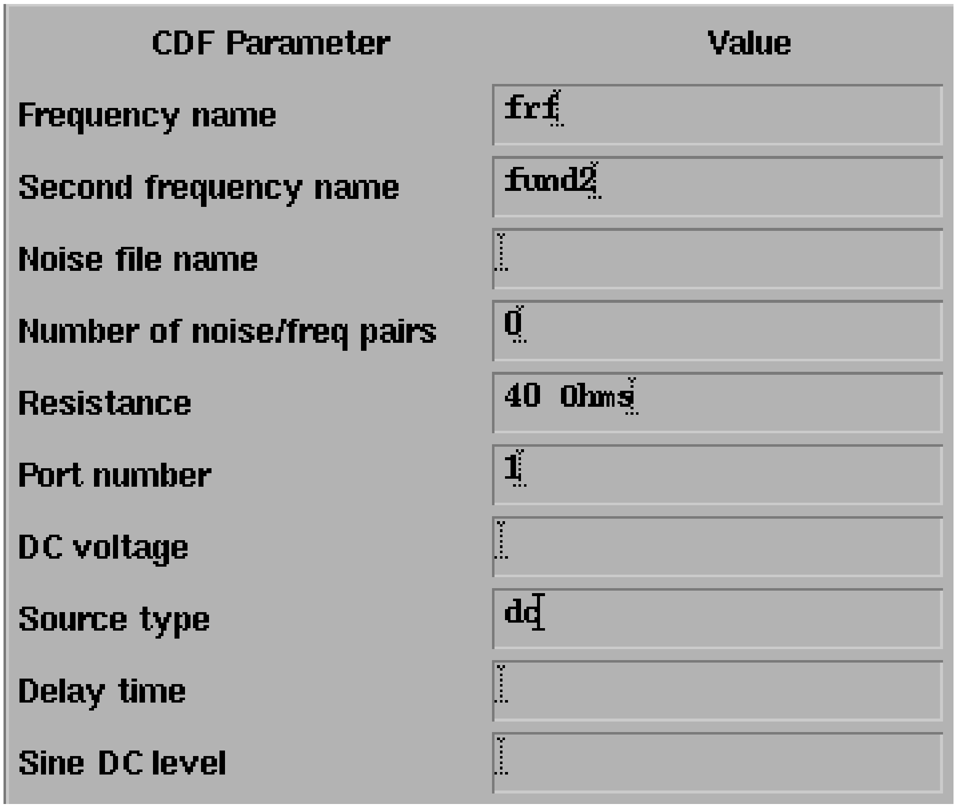
Editing the Schematic

1. In the Schematic window, click on the RF voltage source.



2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.



3. In the *Source type* field, type dc.
4. Click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Note on the isnoisy Parameter

The *isnoisy* parameter, which you can set on the Edit Object Properties form, specifies whether a resistor generates noise.

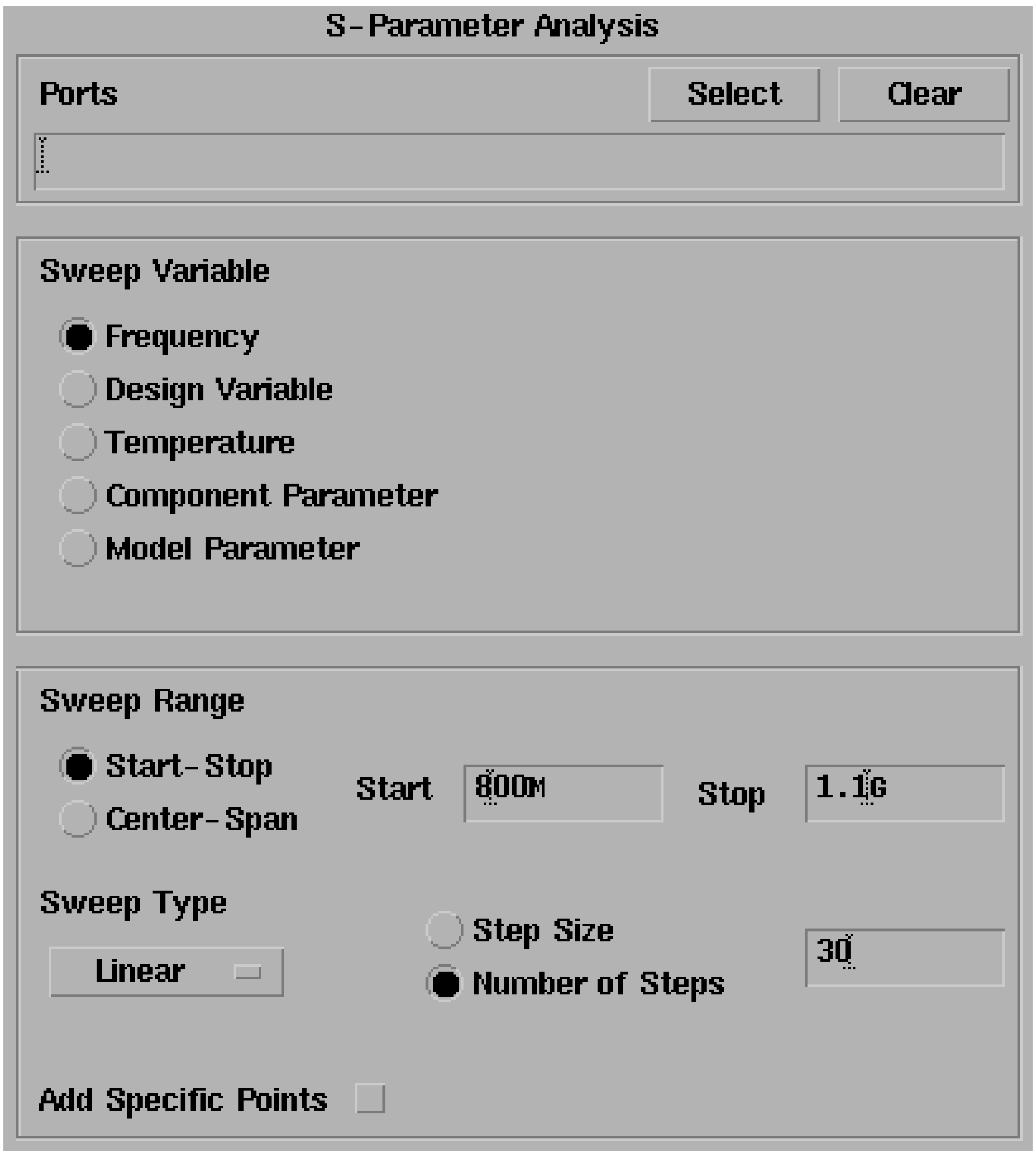
- If you set the parameter to *yes* or *no*, the parameter setting appears in the netlist.
- If you leave the parameter unspecified, the blank value in the cyclic field, the default setting is *yes* but the parameter does not appear in the netlist.

Setting up the S-Parameter Analysis

1. In the Simulation window, choose *Analyses – Choose*.

2. In the Choosing Analyses form, click on *sp* for the *Analysis* choice.
3. Highlight *Frequency* for the *Sweep Variable*.
4. Highlight *Start-Stop* for the *Sweep Range*. Type `800M` in the *Start* field and `1.1G` in the *Stop* field.
5. In the *Sweep Type* cyclic field, choose *Linear* for the sweep type and highlight *Number of Steps*. Type `30` in the *Number of Steps* field.

   The top of the SP Choosing Analyses form appears as follows.



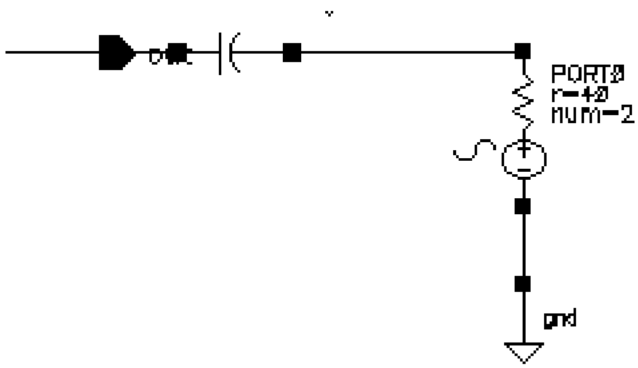
6. Highlight *yes* for *Do Noise*.

   The form changes to let you specify values for the *Output port* and *Input port* fields.

7. To select the Output port

   a. Click the *Select* button next to the *Output port* field.

   b. In the Schematic window, select the output port.

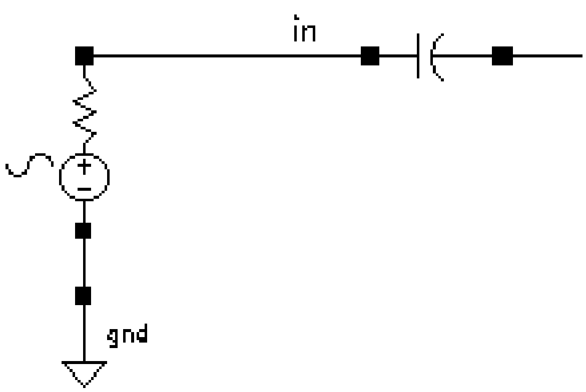


   `/PORT0` displays in the *Output port* field.

8. To select the Input port

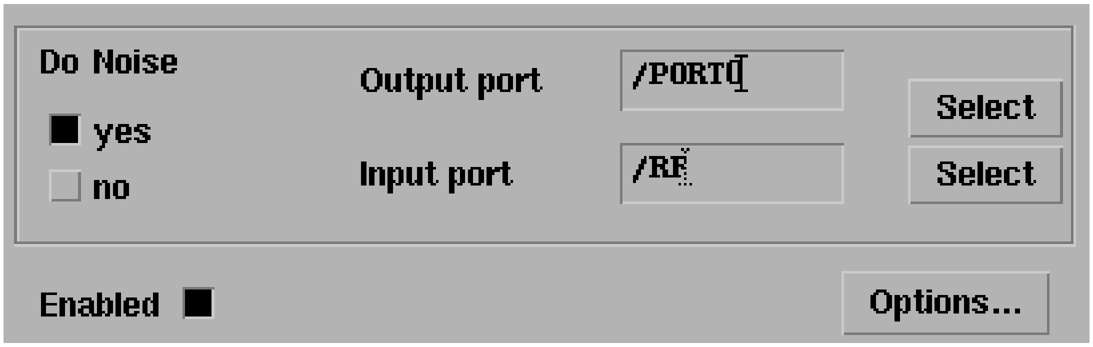
   a. Click the *Select* button next to the *Input port* field.

   b. In the Schematic window, select the RF port.



   `/RF` displays in the *Input port* field.

The bottom of the SP Choosing Analyses form appears as follows.



9. Make sure that *Enabled* is highlighted.
10. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.
2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting the Noise Figure and Minimum Noise Figure

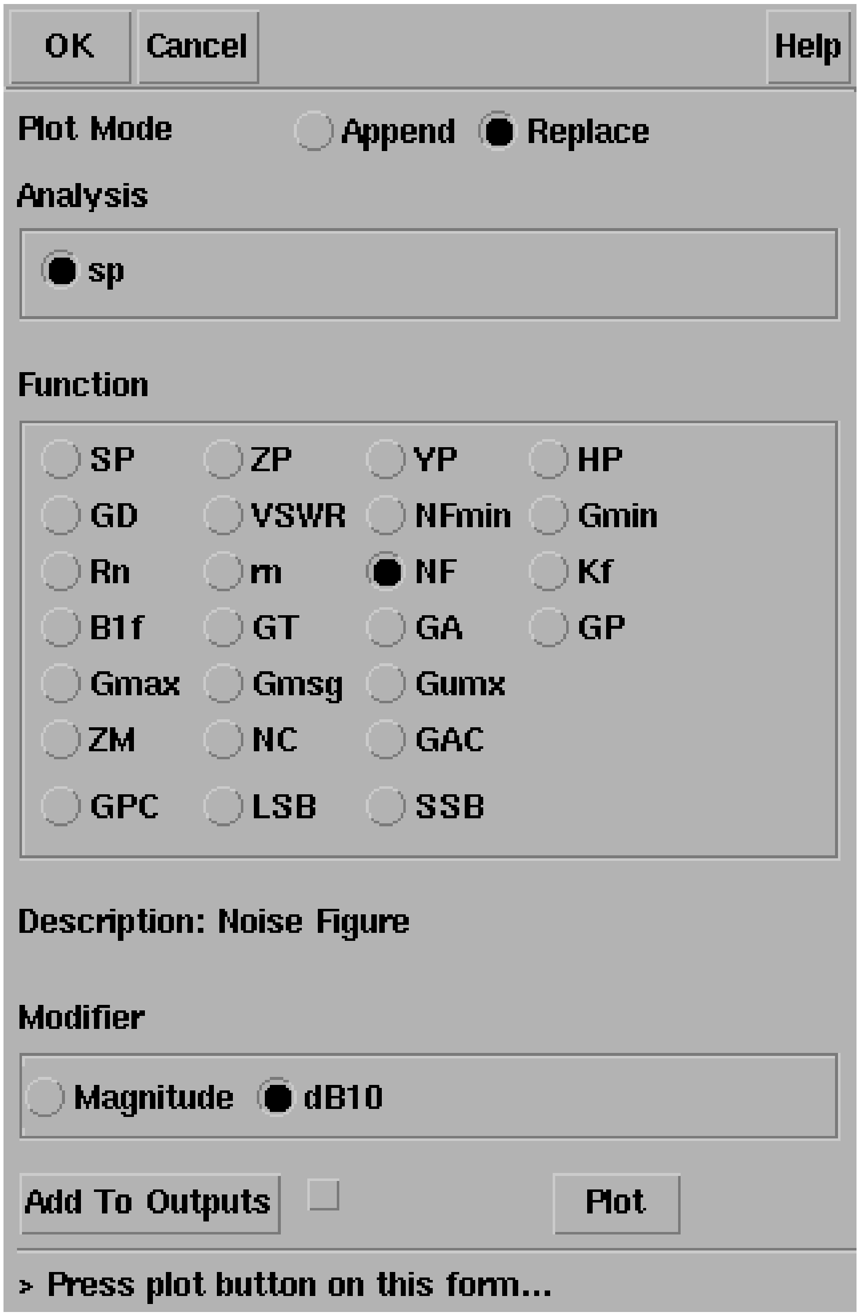
➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

The Waveform window and the Direct Plot form appear.

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *NF* for *Function*.
4. Highlight *dB10* for *Modifier*.

The completed form looks like this.

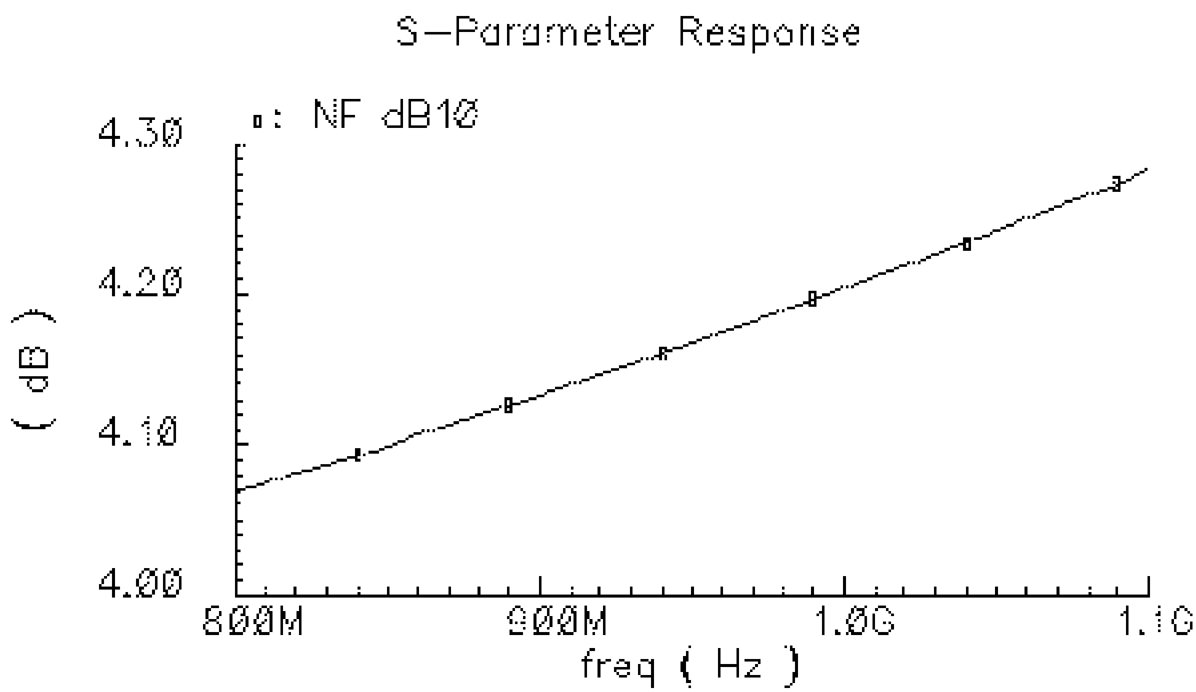


5. Following the message at the bottom of the form,

Press Plot button on this form...

Click on *Plot*.

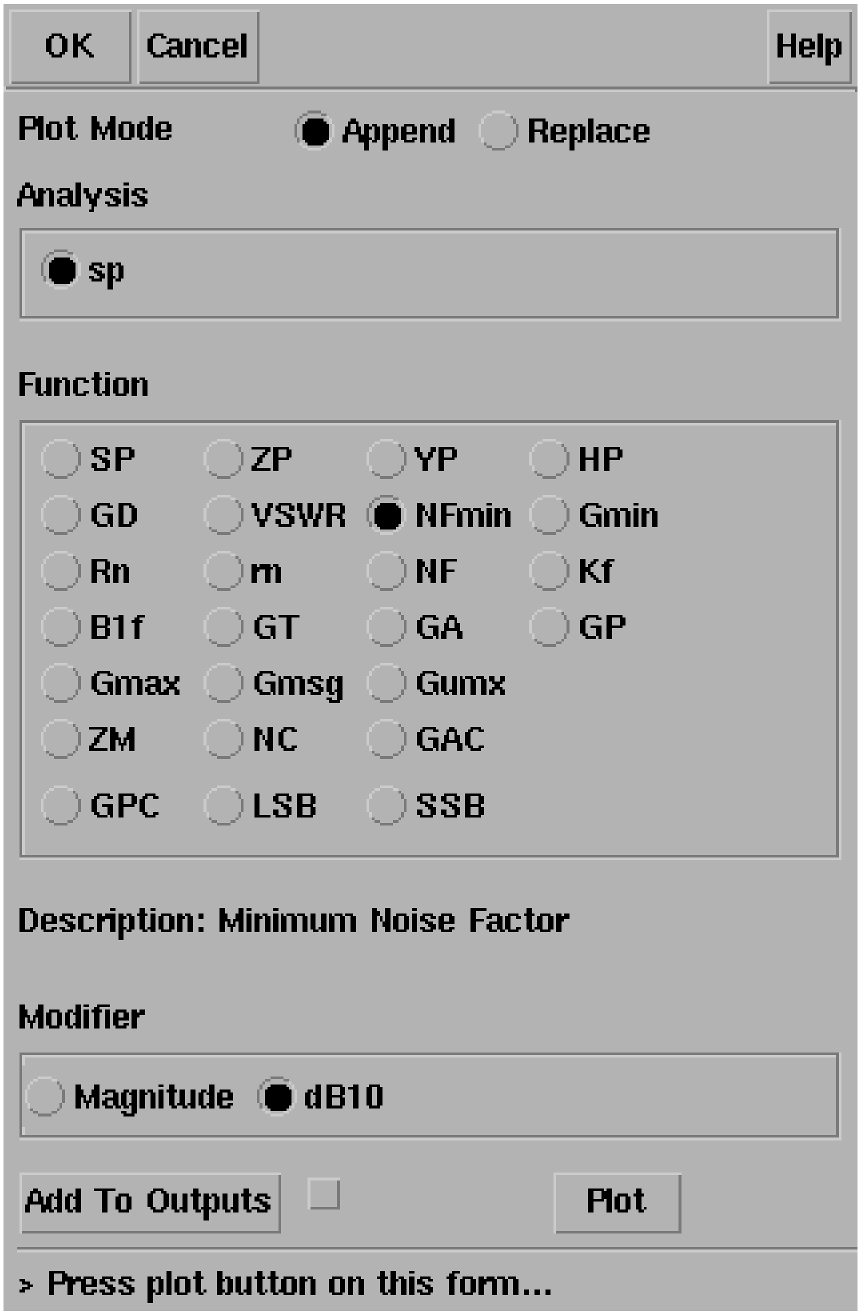
The plot for the noise figure appears in the Waveform window.



To add the minimum noise figure plot to the Waveform window, in the Direct Plot form, do the following:

1. Highlight *Append* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *NFmin* for *Function*.
4. Highlight *dB10* for *Modifier*.
5. Click on *Plot*.

The completed form looks like this.

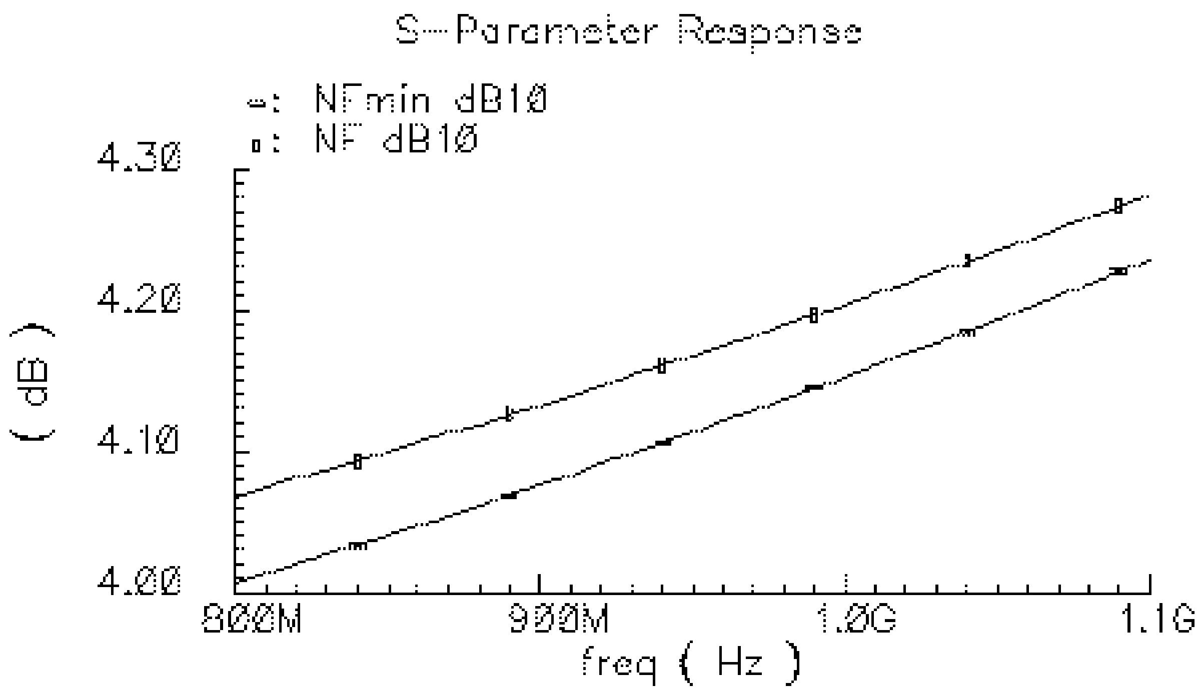


6. Following the message at the bottom of the form,

   Press Plot button on this form...

   Click on *Plot*.

The plot for the minimum noise figure is added to the noise figure plot in the Waveform window.

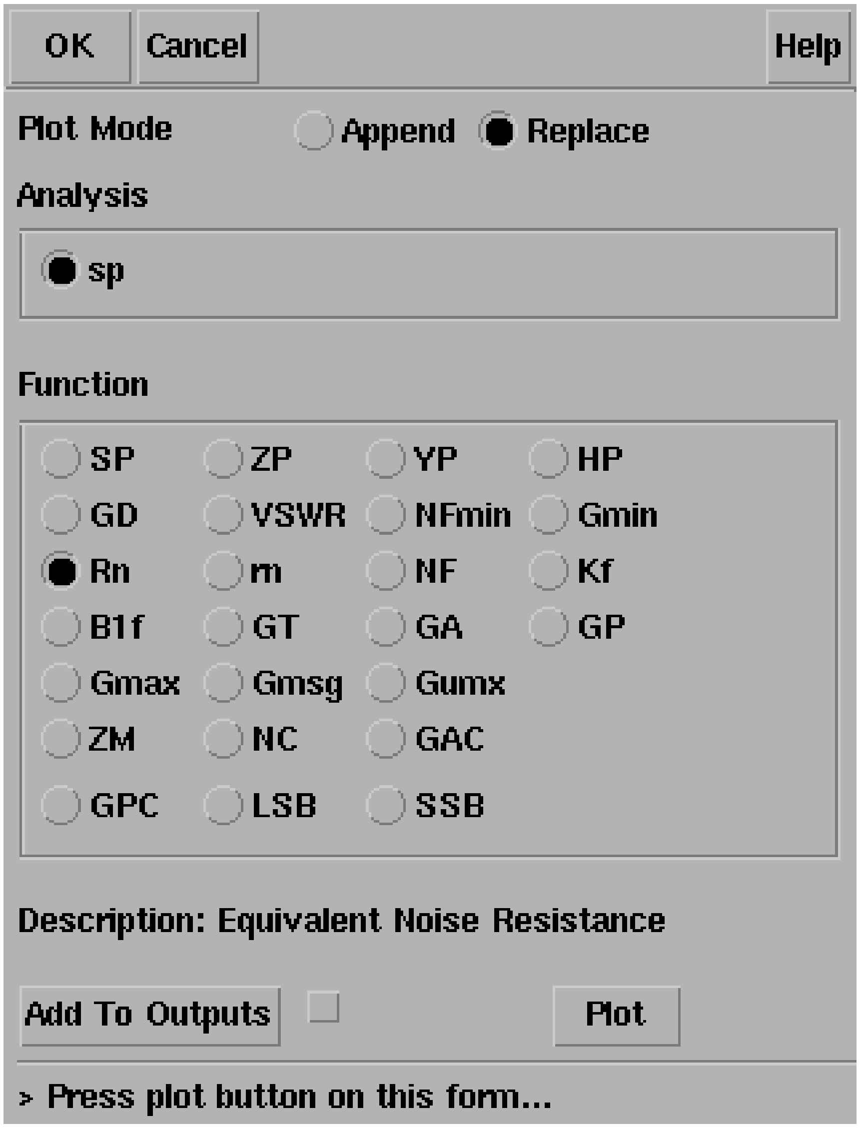


Plotting the Equivalent Noise Resistance

In the Direct Plot form, do the following:

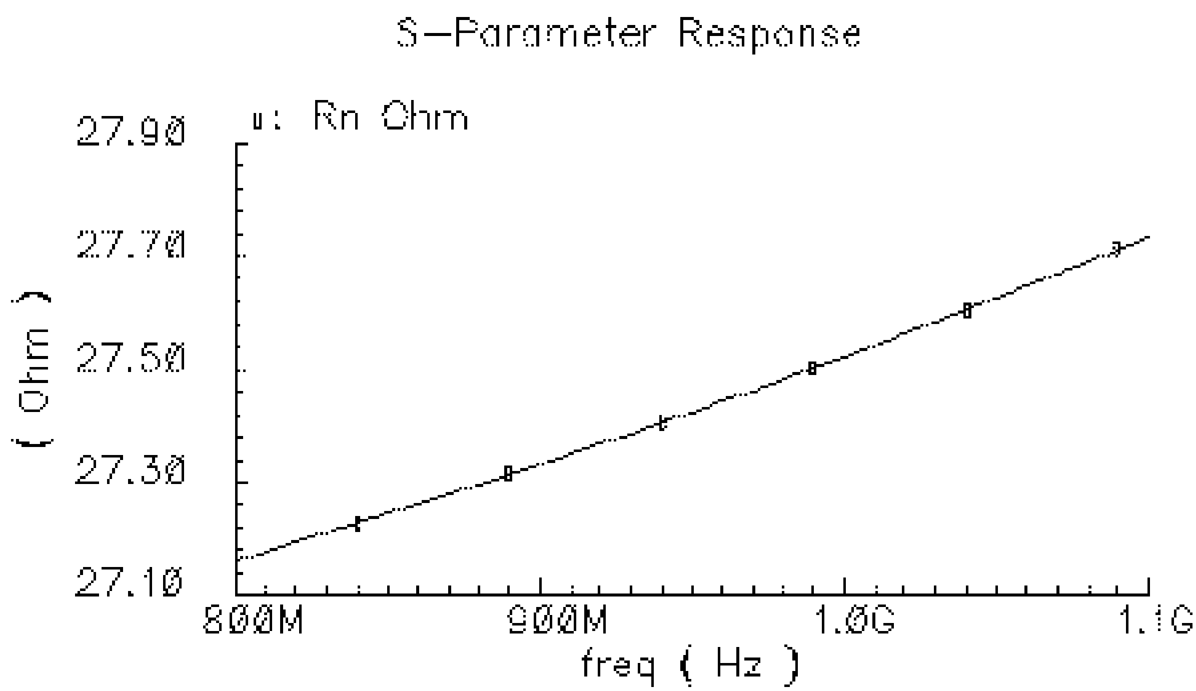
1. Highlight *Replace* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *Rn* for *Function*.

The completed form looks like this.



4. In the Direct Plot form, click on *Plot*.

The equivalent noise resistance is plotted in the Waveform window.



Plotting Load and Source Stability Circles

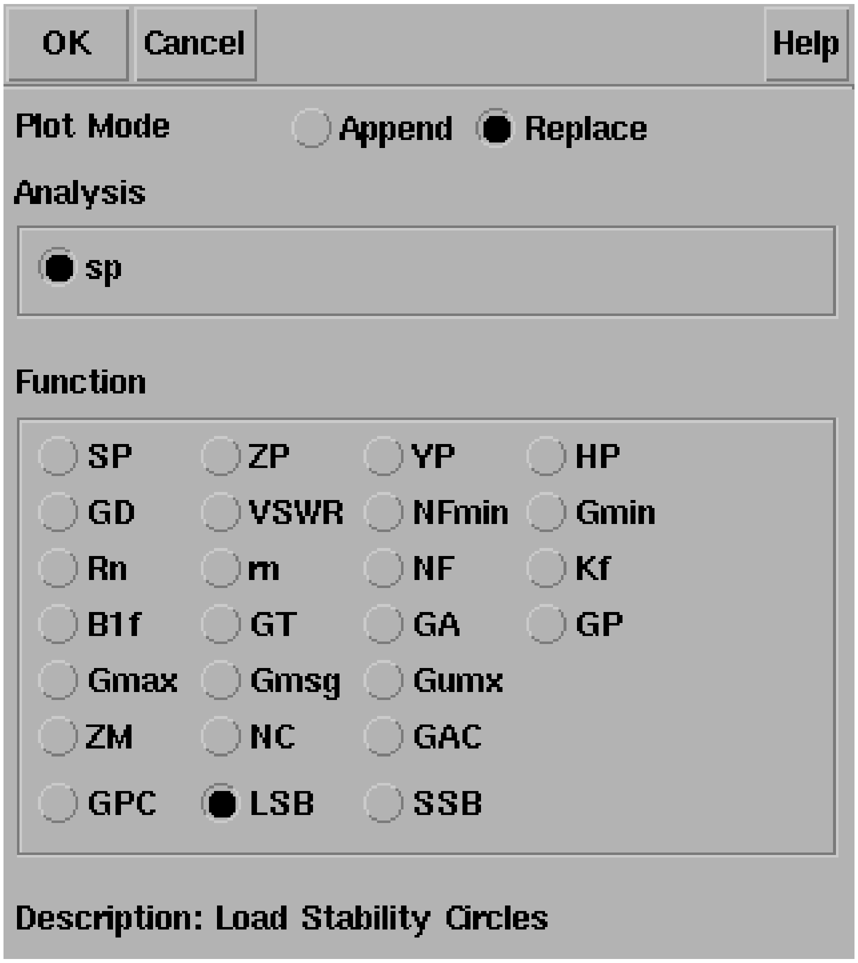
Load and source stability circles show the boundaries between values of load and source impedance that cause instability and values that do not. Either the inside or the outside of the circles might represent the stable region, depending on other values. If, as in the example below, the values of S-parameters S_{11} and S_{22} are less than one for a 50 Ohm impedance system, the center of the normalized Smith chart falls within the stable region.

To Plot the Load Stability Circles

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode.*
2. Highlight *sp* for *Analysis*.
3. Highlight *LSB* for *Function*.

The top of the Direct Plot form looks like this.



4. Highlight *Z-Smith* for *Plot Type*.

   The Direct Plot form changes to accept *Frequency Range (Hz)* values.

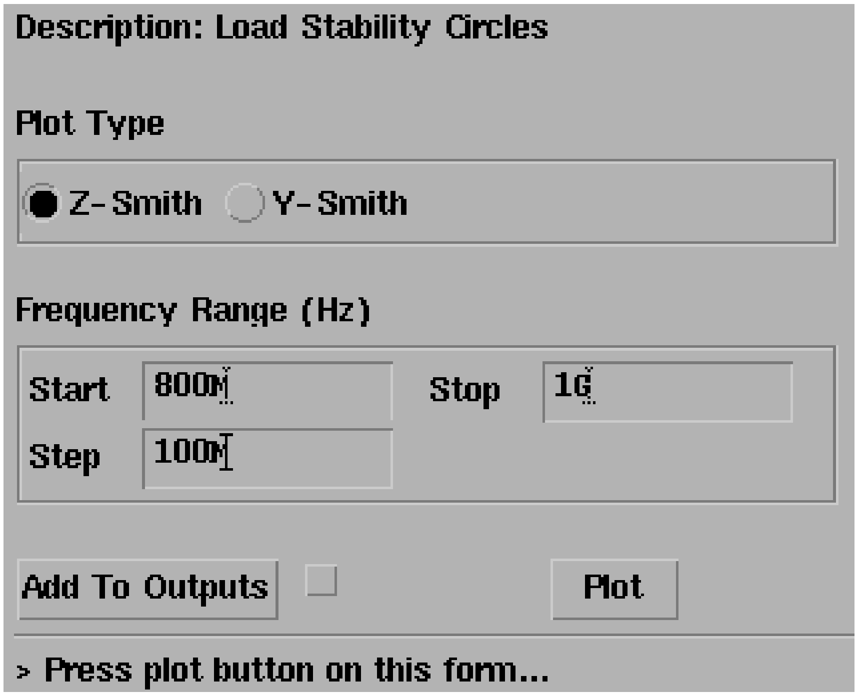
5. For the *Frequency Range (Hz)* values

   a. Type `800M` in the *Start* field.

   b. Type `1G` in the *Stop* field.

   c. Type `100M` in the *Step* field.

The bottom of the Direct Plot form looks like this.



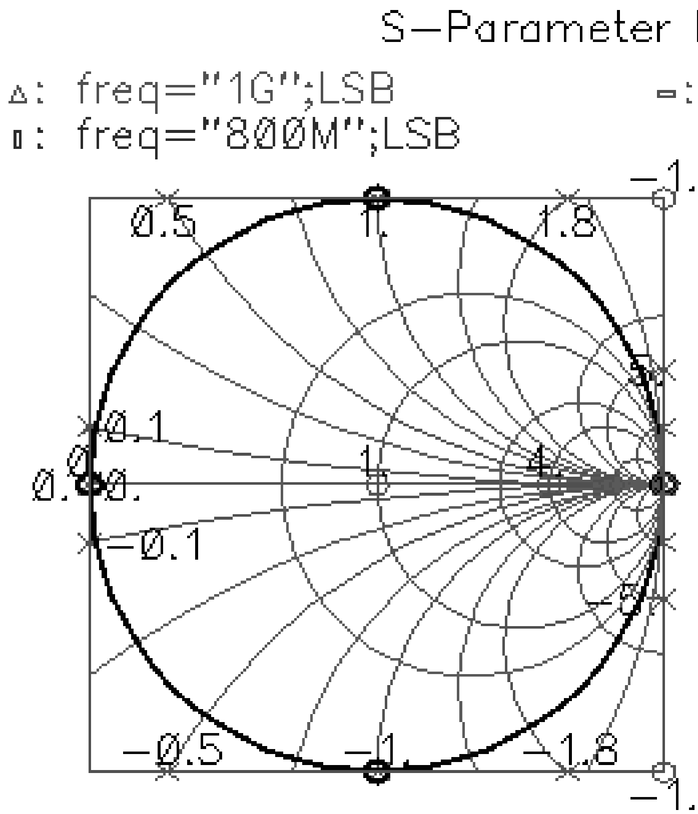
6. Following the message at the bottom of the form,

Press Plot button on this form...

Click on *Plot*.

The load stability circles are plotted, but they are outside the range of the graph.

To display the load stability circles, in the Waveform window, choose *Zoom – Fast Zoom Out* several times.

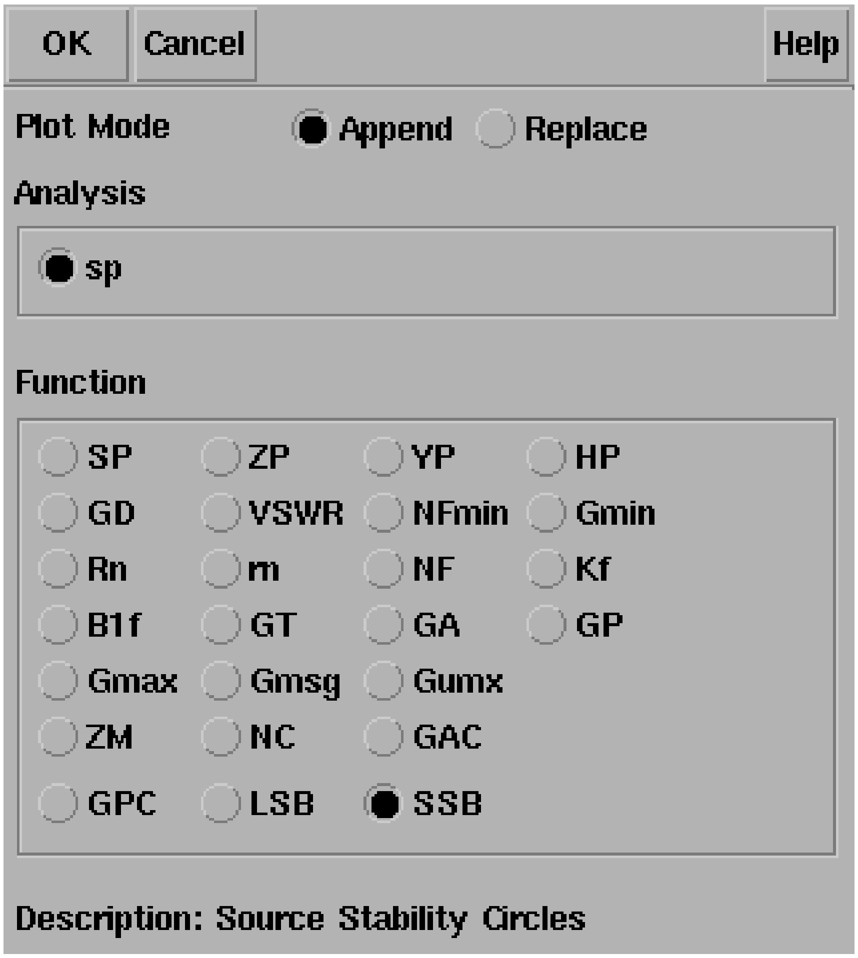


To Plot the Source Stability Circles

In the Direct Plot form, do the following:

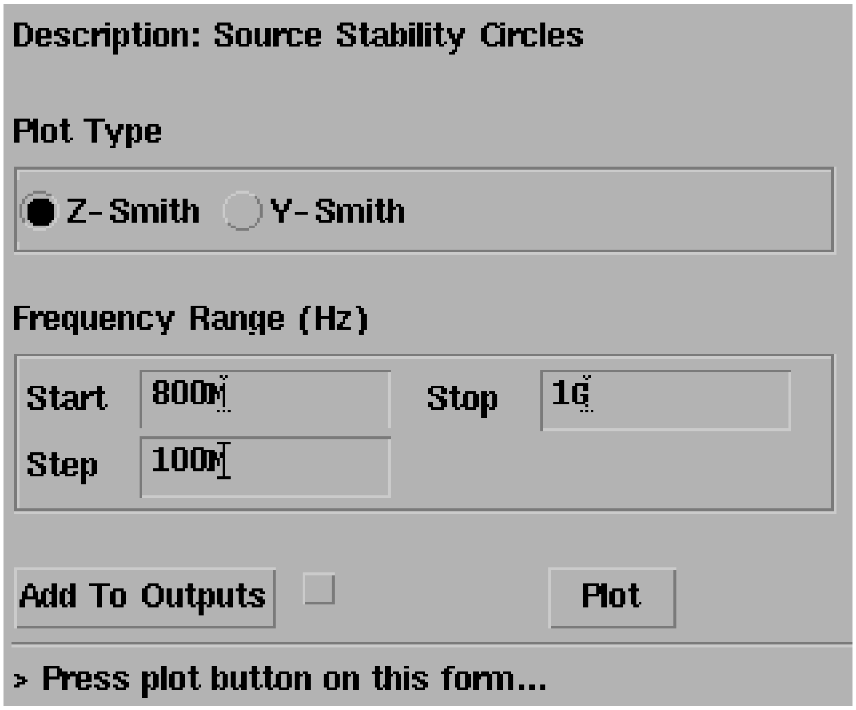
1. Highlight *Append* for *Plot Mode*.
2. Highlight *sp* for *Analysis*.
3. Highlight *SSB* for *Function*.

The top of the Direct Plot form looks like this.



4. Highlight *Z-Smith* for *Plot Type*.
5. For the *Frequency Range (Hz)* values
   a. Type `800M` in the *Start* field.
   b. Type `1G` in the *Stop* field.
   c. Type `100M` in the *Step* field.

The bottom of the Direct Plot form looks like this.



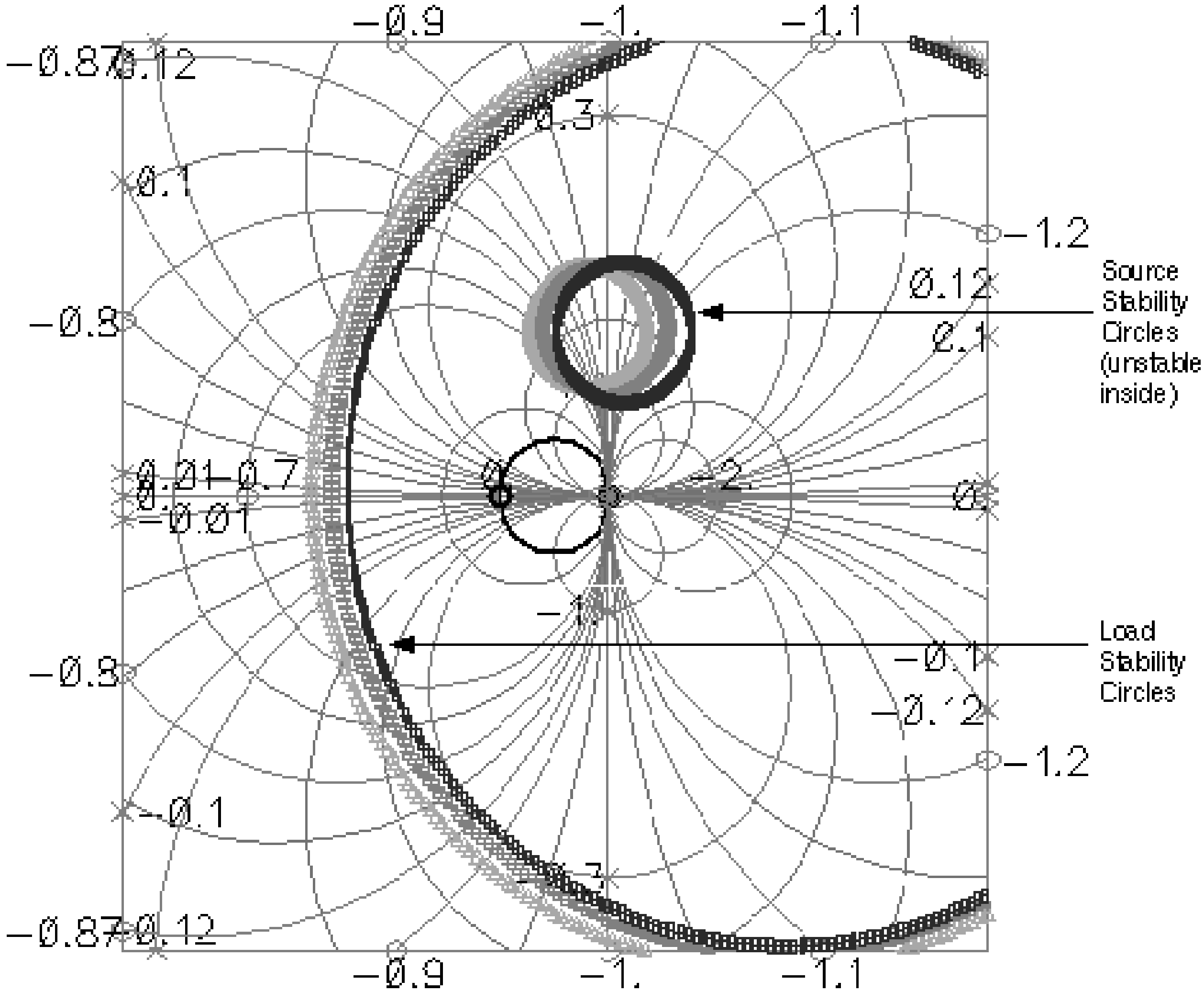
6. Following the message at the bottom of the form,

   Press Plot button on this form...

   Click on *Plot*.

   The source stability circles are plotted in the Waveform window along with the load stability circles.

To see the plot as it appears below, in the Waveform window, choose *Zoom – Fast Zoom Out* three times consecutively.



In the center of the Smith Chart, the values of S11 and S22 are greater than 1, so the center of the Smith chart is part of the unstable region. Therefore, all points inside the load stability circle and all points outside the source stability circle belong to the unstable region.

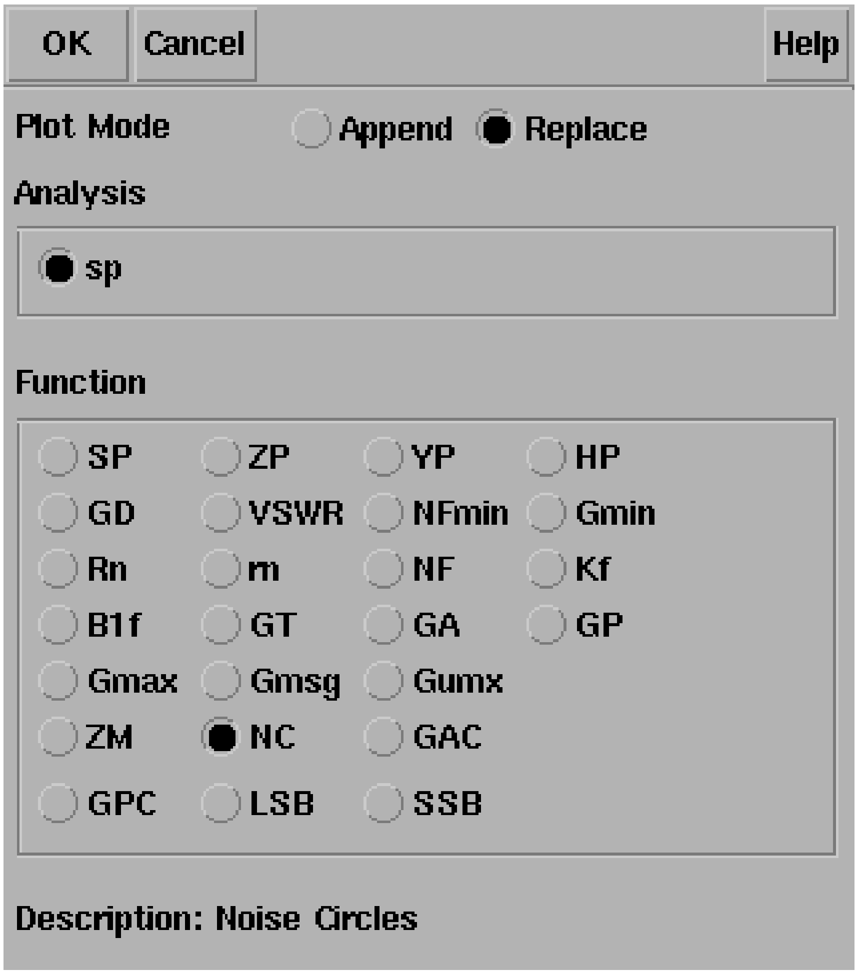
Plotting the Noise Circles

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.

2. Highlight *sp* for *Analysis*.
3. Highlight *NC* for *Function*.

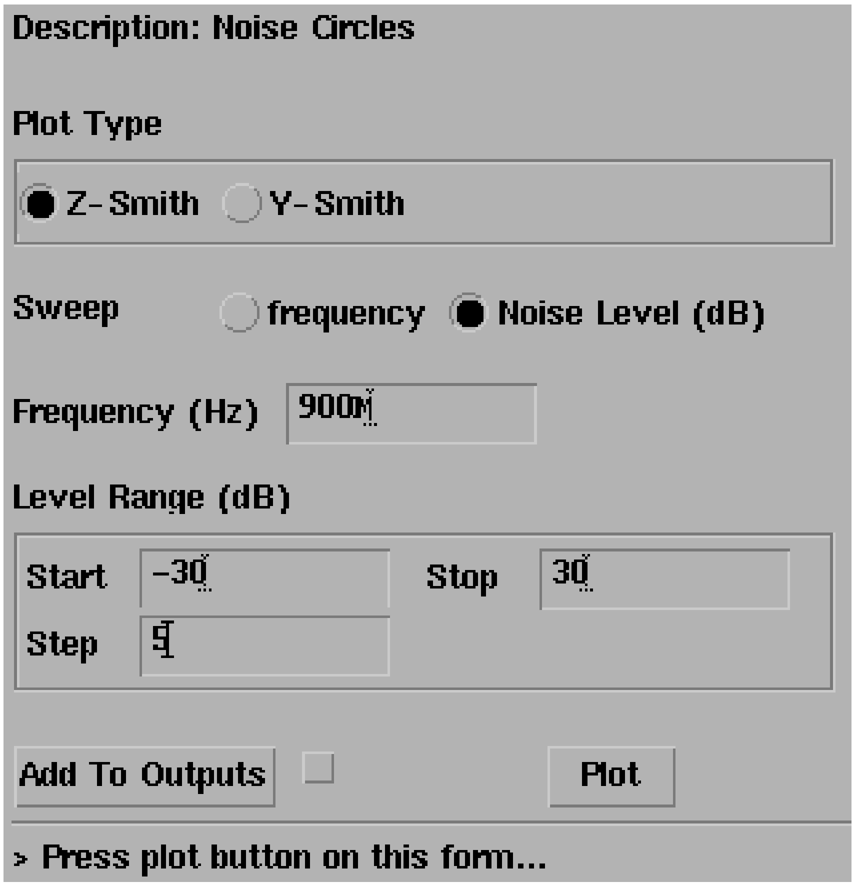
   The top of the Direct Plot form looks like this.



4. Highlight *Z-Smith* for *Plot Type*.
5. Highlight *Noise Level (dB)* for *Sweep*.
6. Type `900M` for *Frequency (Hz)*.
7. For the *Level Range (dB)* values
   a. Type `-30` in the *Start* field.
   b. Type `30` in the *Stop* field.

c. Type 5 in the *Step* field.

The bottom of the Direct Plot form looks like this.



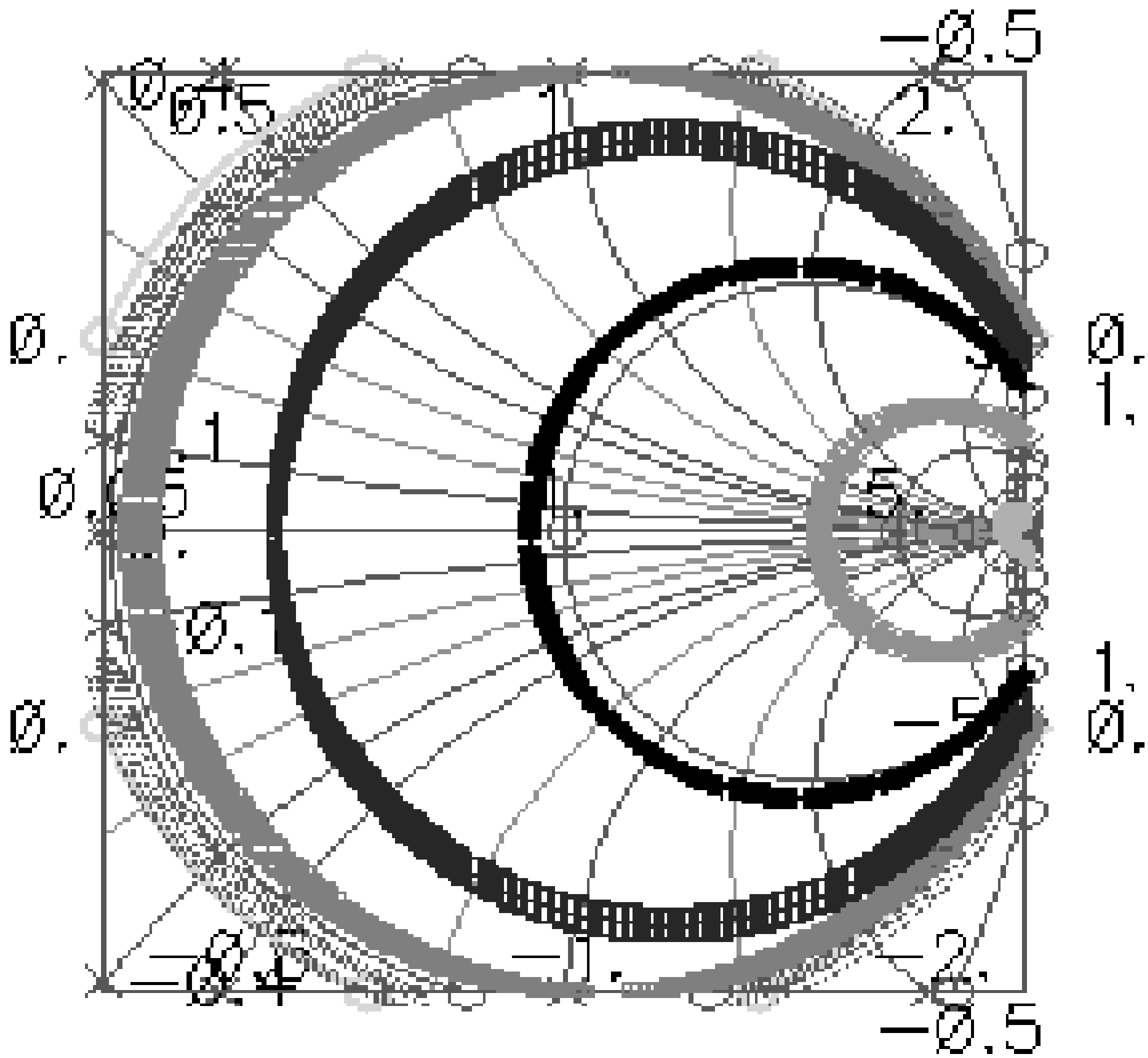
8. Following the message at the bottom of the form,

Press Plot button on this form...

Click on *Plot*.

The noise circles are plotted in the Waveform window.

By choosing *Zoom—Fast Zoom Out* in the Waveform window, you can see the plot as it appears below.



Noise Calculations with PSS and Pnoise

Setting Up the Simulation

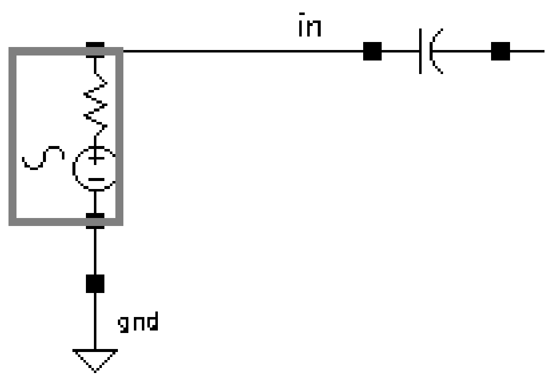
1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

For this simulation you need the `rfModels.scs` model file

4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

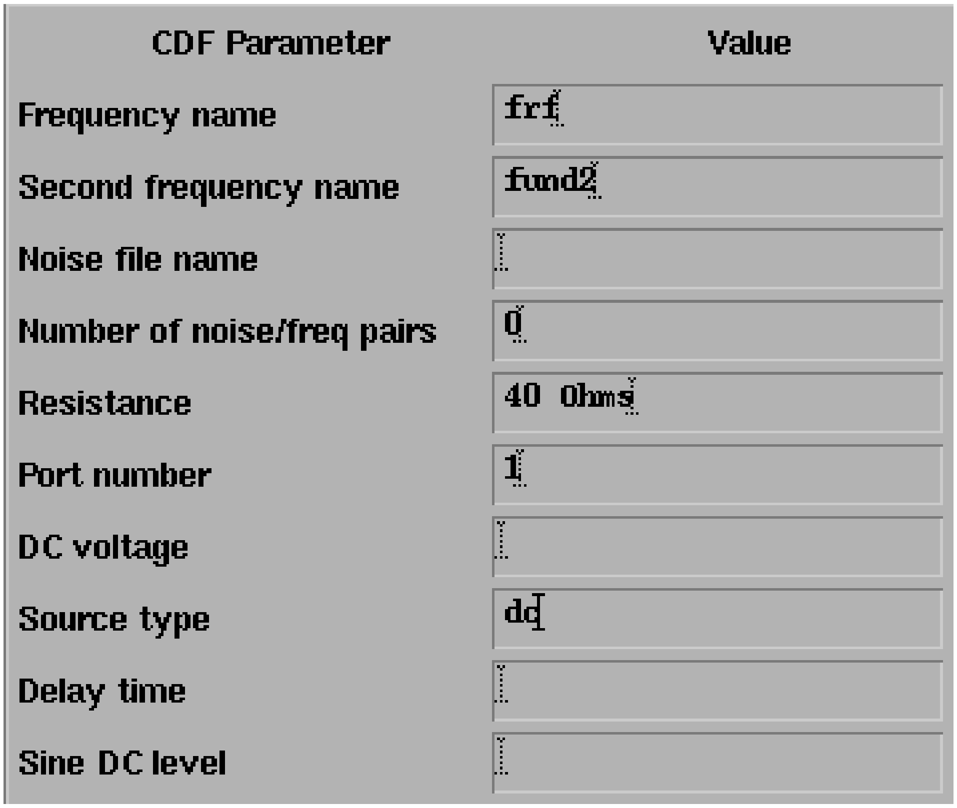
Editing the Schematic

1. In the Schematic window, click on the RF voltage source.



2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.



CDF Parameter	Value
Frequency name	frf
Second frequency name	fund2
Noise file name	
Number of noise/freq pairs	0
Resistance	40 Ohms
Port number	1
DC voltage	
Source type	dc
Delay time	
Sine DC level	

3. In the *Source type* field, type `dc`.
4. Click *OK*.
5. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS and Pnoise Analyses

➤ In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting up the PSS Analysis

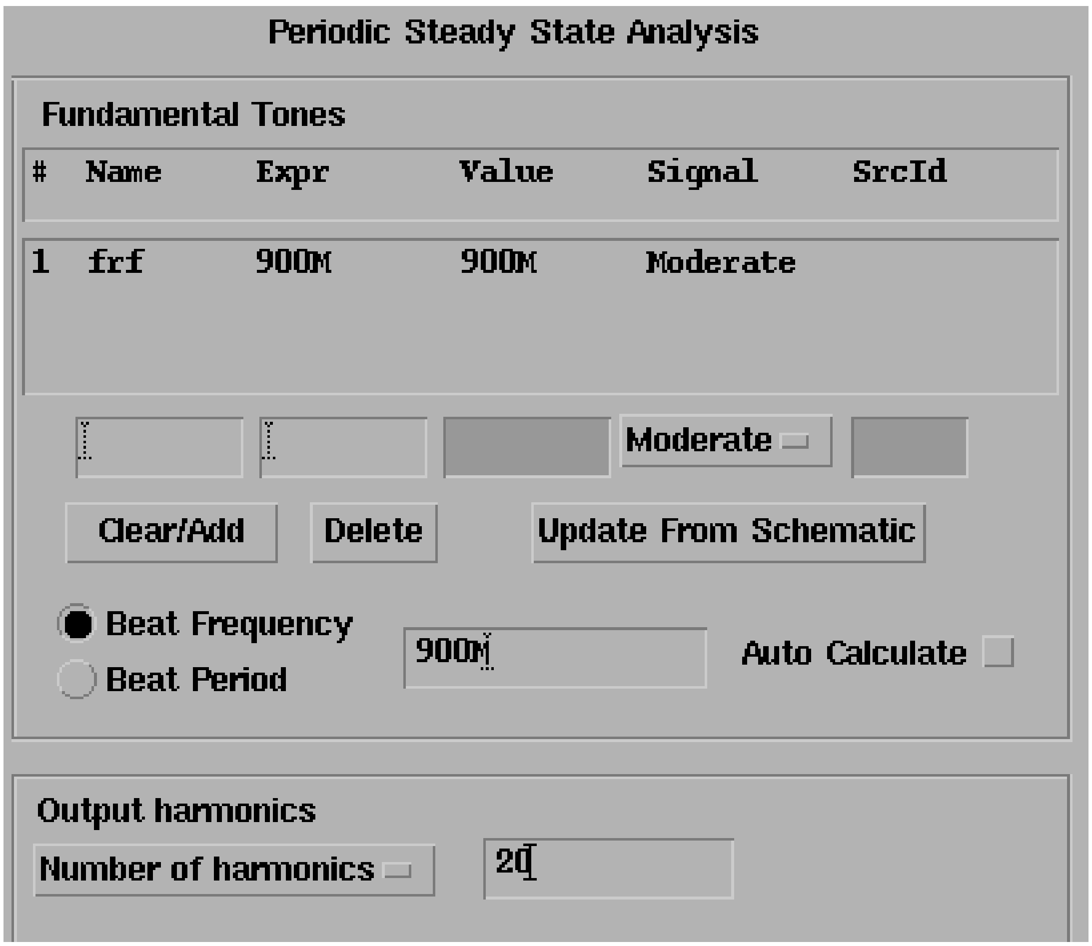
1. In the Choosing Analyses form, click on *pss*.
2. The *Beat Frequency* button is highlighted by default. Type `900M` in the *Beat Frequency* field.

When the *Source type* is dc, you must enter the fundamental frequency manually.

Be sure the *Auto Calculate* button is *not* highlighted.

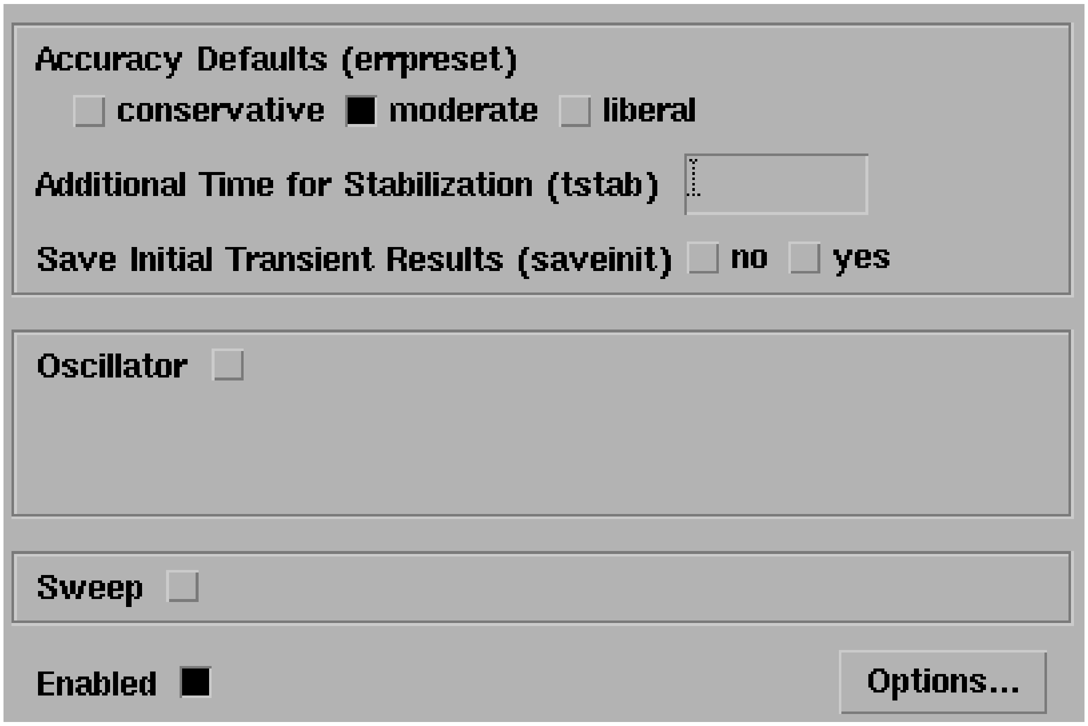
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as 20, in the *Number of harmonics* field.

   The top of the Choosing Analysis form for PSS appears as follows.



4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
5. Highlight *Enabled*.

The bottom of the Choosing Analysis form for PSS appears as follows.



Setting up the Pnoise Analysis

1. At the top of the Choosing Analyses form, click *pnoise*.

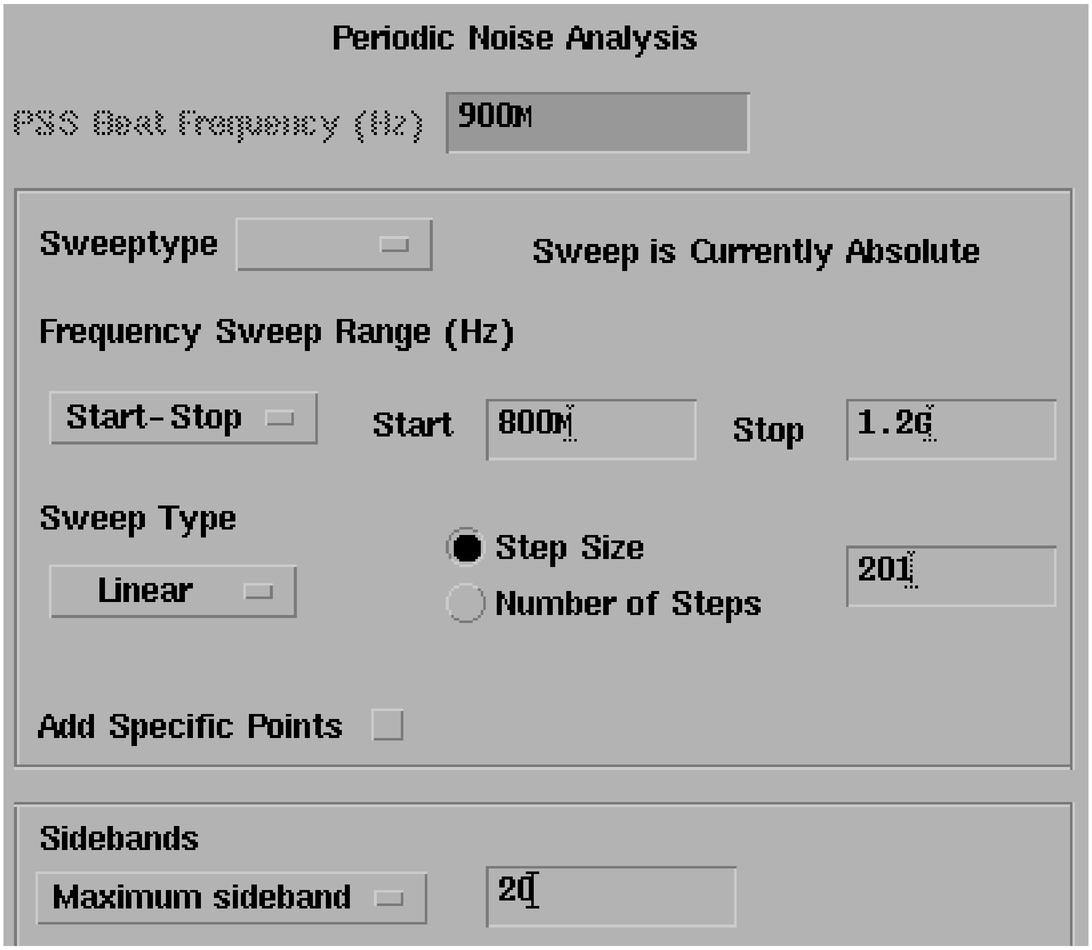
   The form changes to let you specify data for the periodic noise or, Pnoise, analysis.

2. In the *Frequency Sweep Range (Hz)* cyclic field, choose *Start – Stop*. Type `800M` in the *Start* field and `1.2G` in the *Stop* field.

3. In the *Sweep Type* cyclic field, choose *Linear* and click on the *Number of Steps* button. Type `201` in the *Number of Steps* field.

4. In the *Sidebands* cyclic field choose *Maximum sideband* and type `20` in the *Maximum sideband* field.

The top of the Choosing Analysis form for Pnoise appears as follows.



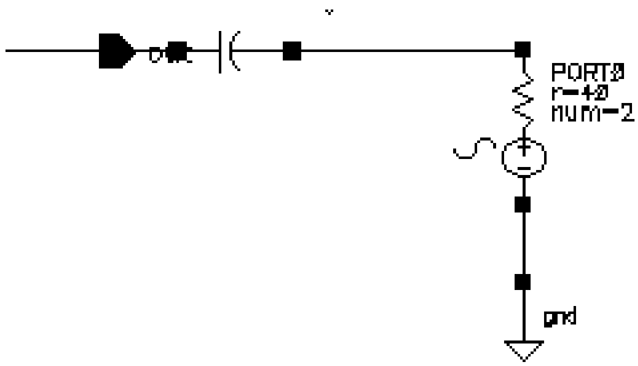
5. In the Output cyclic field, choose *voltage*.

The value is appropriate because the reference sideband is an LNA. The form changes to display positive and negative output node fields.

6. To select the *Positive Output Node*

a. Click the *Select* button next to the *Positive Output Node* field.

b. In the Schematic window, select the amplifier output net.



`/net28` displays in the *Positive Output Node* field.

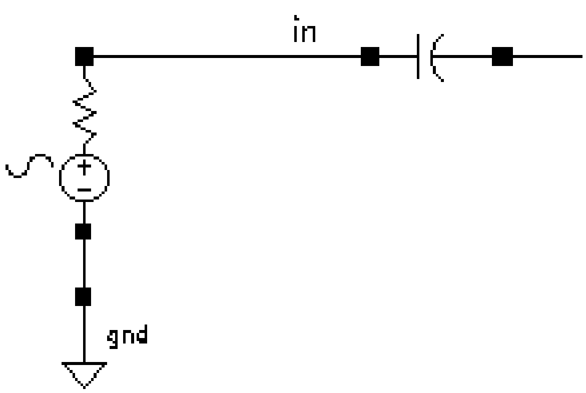
7. Leave the *Negative Output Node* field empty. It defaults to `/gnd!`.

You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then clicking on the output node in the schematic.

8. To select the *Input Voltage Source*

a. Click the *Select* button next to the *Input Voltage Source* field.

b. In the Schematic window, select the RF port.



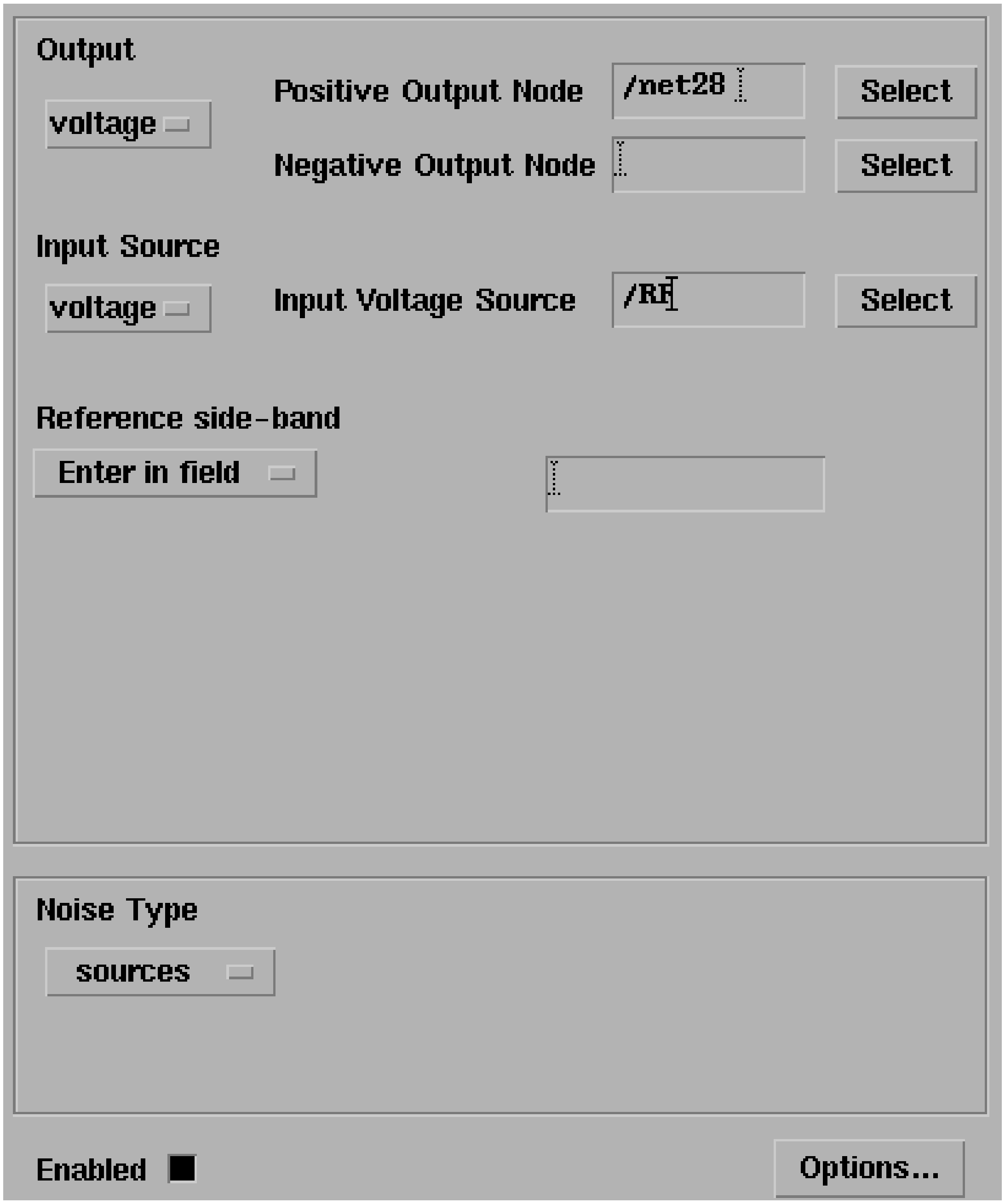
`/RF` displays in the *Input Voltage Source* field.

9. In the *Reference side-band* field, type `0`.

You specify the reference sideband as `0` for an LNA because an LNA has no frequency conversion from input to output.

10. Highlight *Enabled*.

The bottom of the Choosing Analysis form for Pnoise appears as follows.



11. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting the Noise Calculations

➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Waveform window and the Direct Plot form appear.

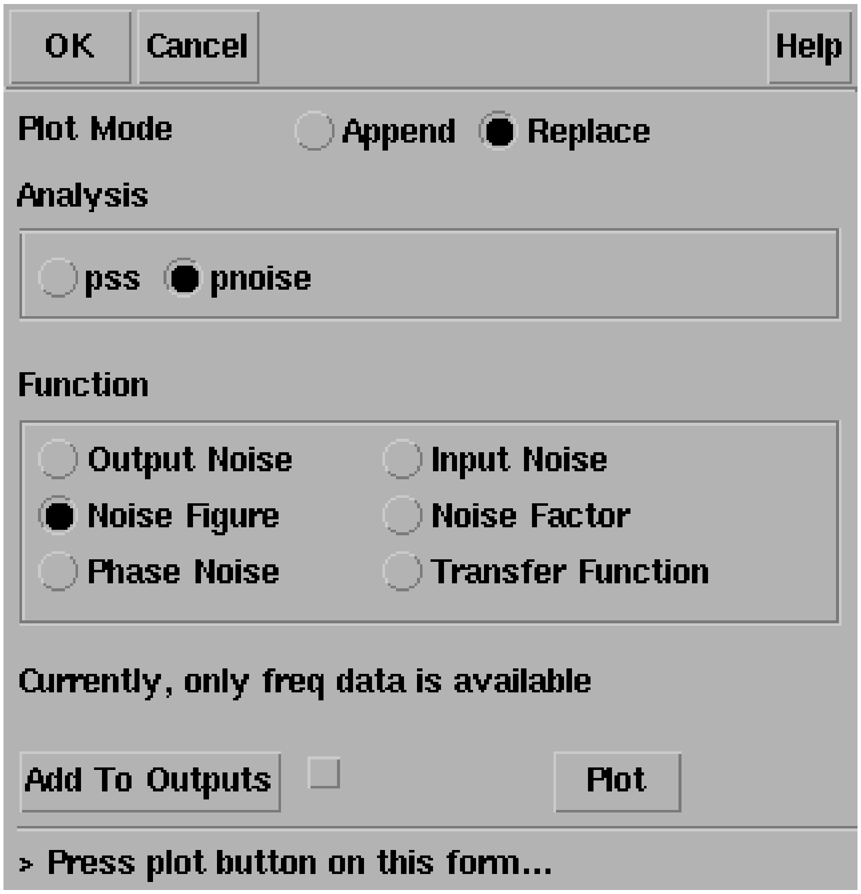
In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pnoise* for *Analysis*.

   The form changes to display fields for the Pnoise analysis.

3. Highlight *Noise Figure* for *Function*.

The completed form looks like this.

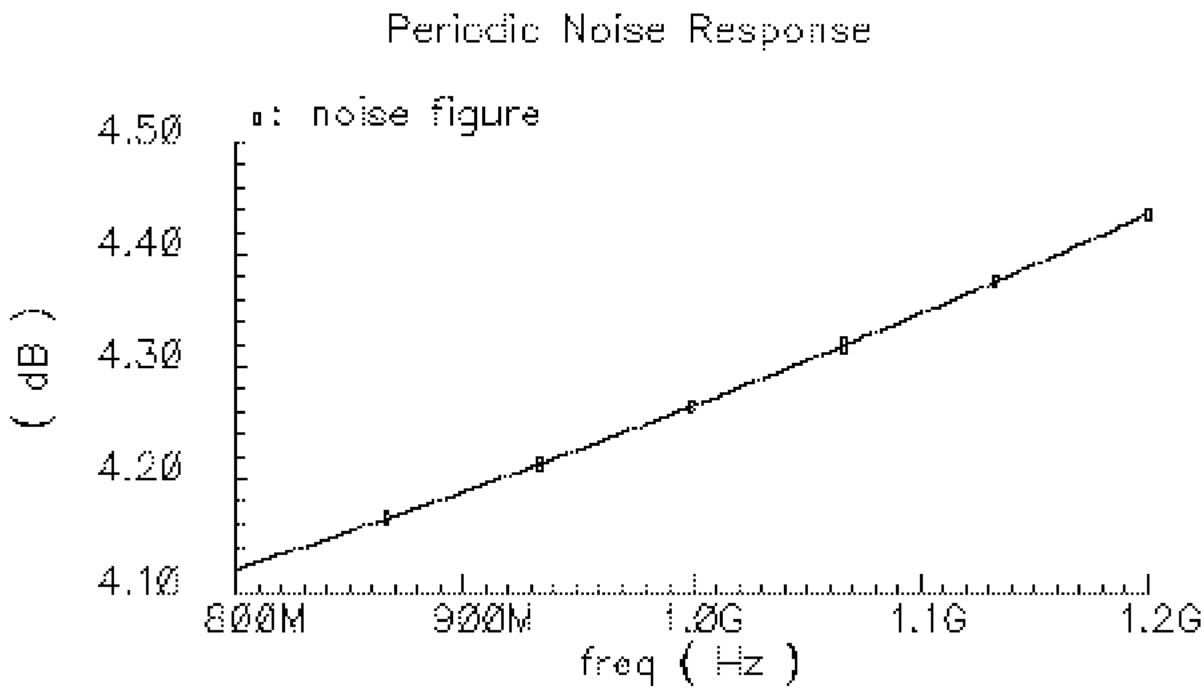


4. Follow the message at the bottom of the form

Press plot button on this form...

Click on *Plot*.

The plot appears in the Waveform window. The Noise Figure at 900MHz is about 4.19 dB.



5. Close the Direct Plot form and the Waveform window.

Printing the Noise Summary

1. In the Simulation window, choose *Results – Print – Noise Summary*.

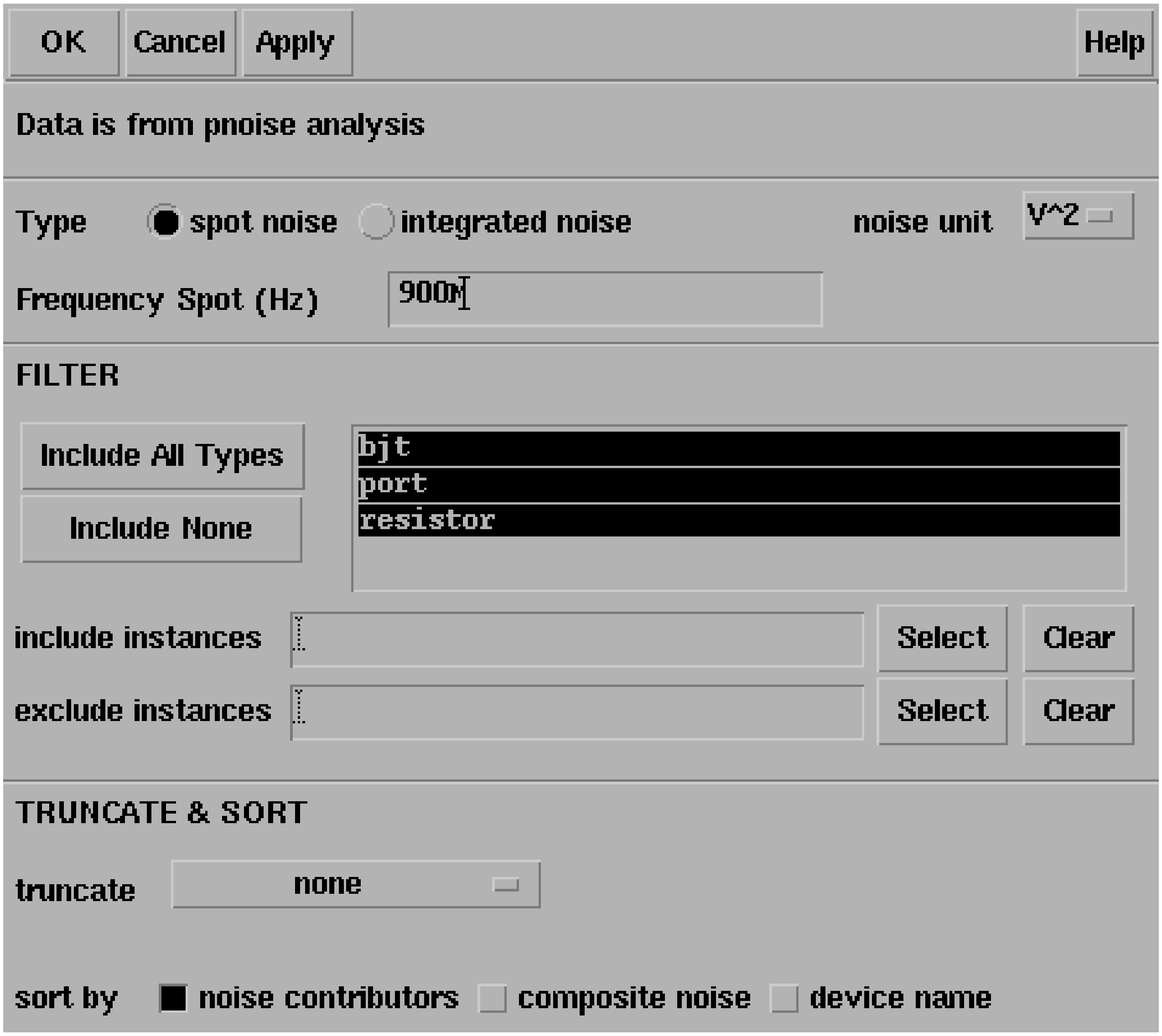
   The Results Display window and the Noise Summary form appear.

In the Noise Summary form, do the following:

1. Highlight *spot noise* for *Type*.
2. Type `900M` in the *Frequency Spot (Hz)* field.
3. Click on *Include All Types* in the *FILTER* list box.

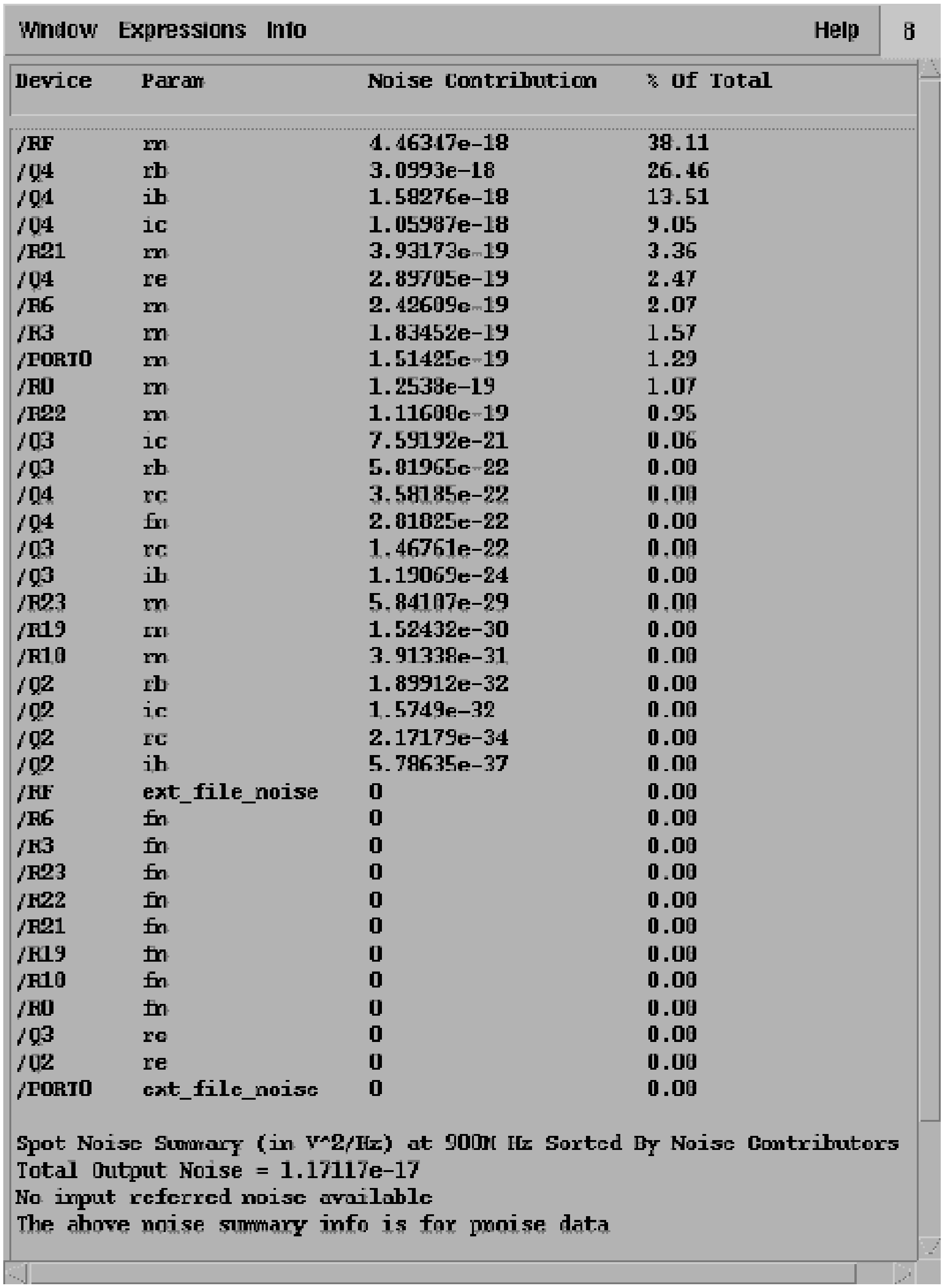
   The *Include All Types* is the *FILTER* list box default, however, you must click on the button to activate the choice.
4. In the *truncate* cyclic field, select *none* in the *TRUNCATE & SORT* section.

The filled-in Noise Summary form looks like this.



5. In the Noise Summary form, click *OK*.

The noise contributors now appear in the Results Display window.



Window Expressions Info Help 8

Device	Param	Noise Contribution	% Of Total
/RF	rn	4.46347e-18	38.11
/Q4	rb	3.0993e-18	26.46
/Q4	ib	1.58276e-18	13.51
/Q4	ic	1.05987e-18	9.05
/R21	rn	3.93173e-19	3.36
/Q4	re	2.89705e-19	2.47
/R6	rn	2.42609e-19	2.07
/R3	rn	1.83452e-19	1.57
/PORT0	rn	1.51425e-19	1.29
/R0	rn	1.2538e-19	1.07
/R22	rn	1.11600e-19	0.95
/Q3	ic	7.59192e-21	0.06
/Q3	rb	5.81965e-22	0.00
/Q4	rc	3.58185e-22	0.00
/Q4	fn	2.81825e-22	0.00
/Q3	rc	1.46761e-22	0.00
/Q3	ib	1.19069e-24	0.00
/R23	rn	5.84107e-29	0.00
/R19	rn	1.52432e-30	0.00
/R10	rn	3.91338e-31	0.00
/Q2	rb	1.89912e-32	0.00
/Q2	ic	1.5749e-32	0.00
/Q2	rc	2.17179e-34	0.00
/Q2	ib	5.78635e-37	0.00
/RF	ext_file_noise	0	0.00
/R6	fn	0	0.00
/R3	fn	0	0.00
/R23	fn	0	0.00
/R22	fn	0	0.00
/R21	fn	0	0.00
/R19	fn	0	0.00
/R10	fn	0	0.00
/R0	fn	0	0.00
/Q3	re	0	0.00
/Q2	re	0	0.00
/PORT0	ext_file_noise	0	0.00

Spot Noise Summary (in V^2/Hz) at 900M Hz Sorted By Noise Contributors
Total Output Noise = 1.17117e-17
No input referred noise available
The above noise summary info is for pnoise data

This summary helps you determine the percentage of noise contributed by the different devices in the circuit.

Plotting the 1dB Compression Point

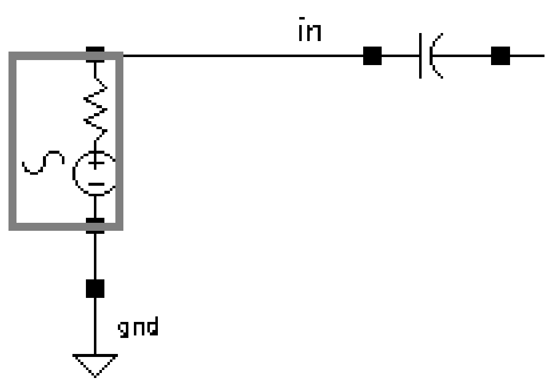
Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file.
4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

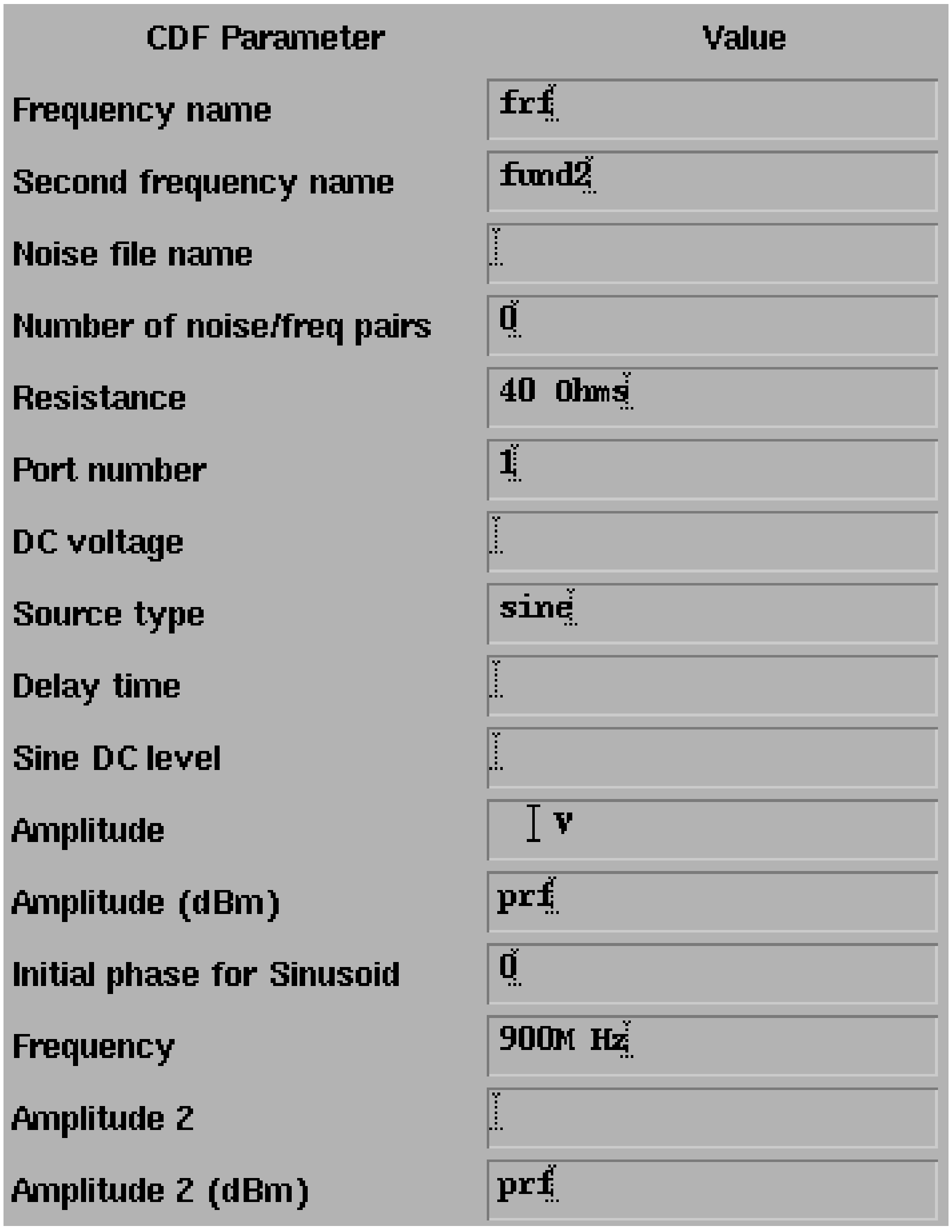
Editing the Schematic

1. In the Schematic window, click on the RF voltage source.



2. In the Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.



CDF Parameter	Value
Frequency name	frf
Second frequency name	fund2
Noise file name	
Number of noise/freq pairs	0
Resistance	40 Ohms
Port number	1
DC voltage	
Source type	sine
Delay time	
Sine DC level	
Amplitude	V
Amplitude (dBm)	prf
Initial phase for Sinusoid	0
Frequency	900M Hz
Amplitude 2	
Amplitude 2 (dBm)	prf

3. In the *Source type* field, type `sine`.
4. Type `900M` for the *Frequency* value.
5. Type `prf` for the *Amplitude (dBm)* value.
6. Be sure the *Amplitude* field is empty.

7. In the Edit Objects Properties form, click *OK.*
8. In the Schematic window, choose *Design – Check and Save*.

Setting up the PSS Analysis

1. In the Simulation window, choose *Analyses – Choose*.

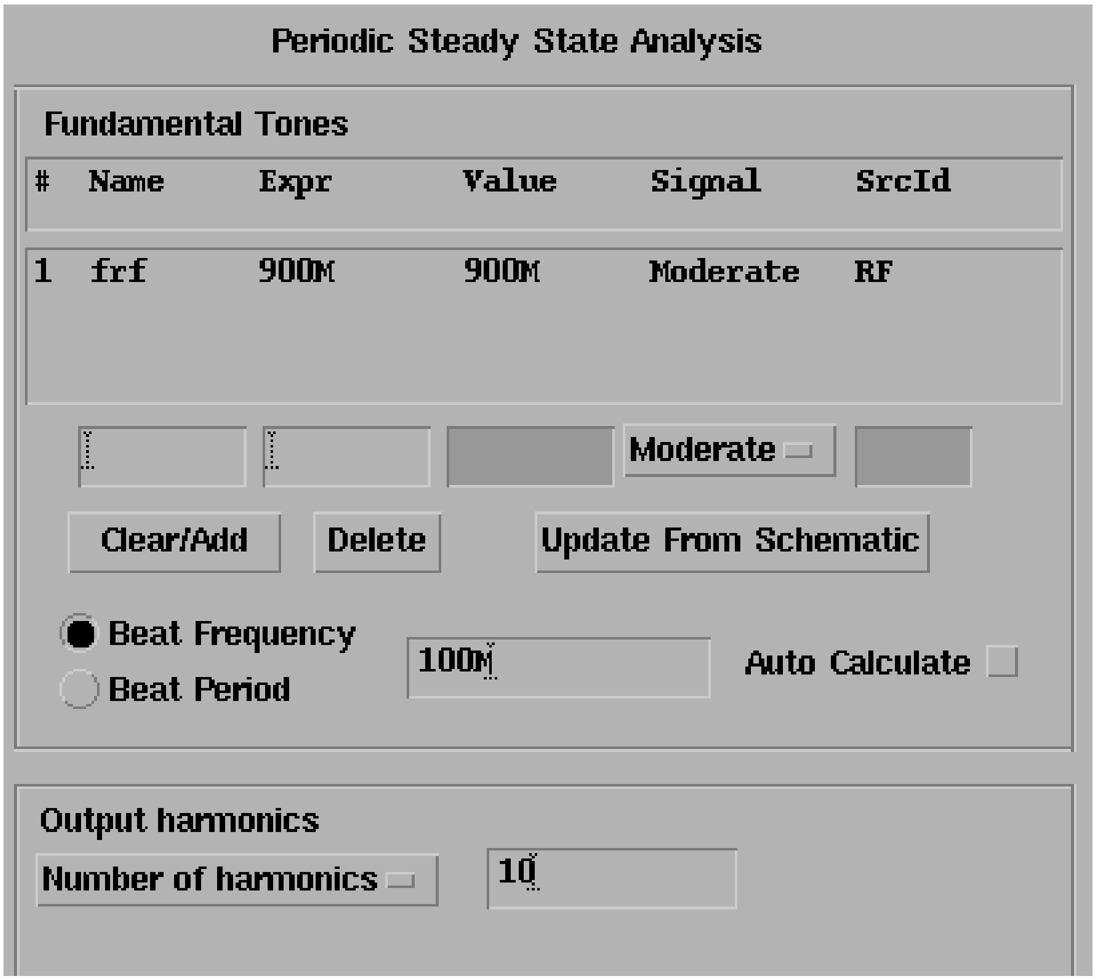
   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click on *pss*.

   The form changes to let you specify data for the PSS analysis.
3. Type `100M` for the *Fundamental Frequency*.

   The *Beat Frequency* button is highlighted by default.

   Be sure the *Auto Calculate* button is *not* highlighted.
4. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `10`, in the *Number of harmonics* field.

The top of the PSS form appears like the one below.



5. Highlight *moderate* for the *Accuracy Default (errpreset).*

6. Highlight the *Sweep* button.

The form changes to let you specify data for the sweep.

7. In the *Sweep* cyclic field, choose *Variable*.

8. Click the *Select Design Variable* button.

The Select Design Variable form appears.

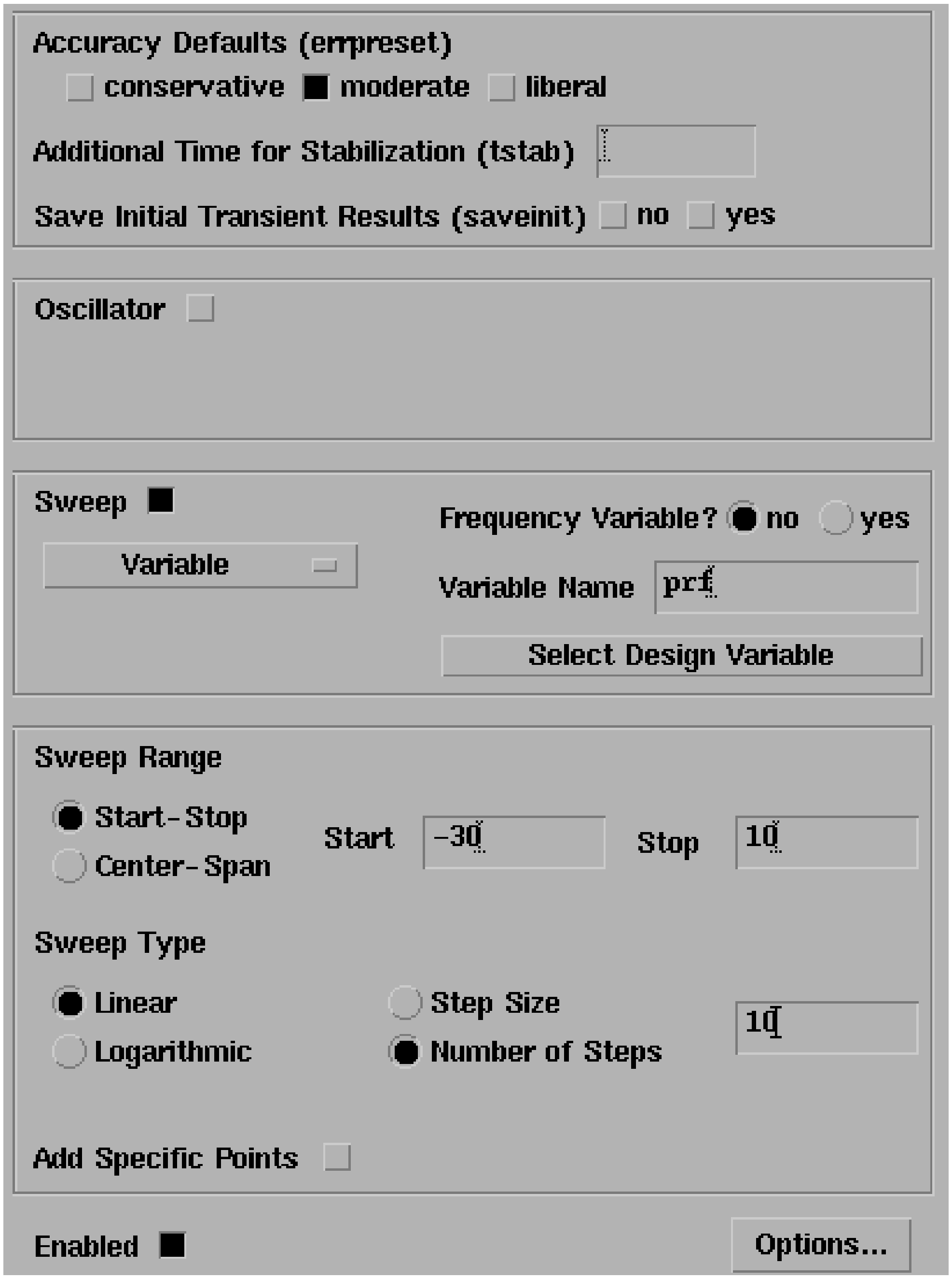
9. In the Select Design variable form, highlight *prf* and click *OK*.

The *prf* variable is the amplitude, in dBm, of the RF input source. If you are not sure of this, use the Edit Object Properties form to examine the *Amplitude (dBm)* field for the RF voltage source.

10. Choose *Start-Stop* for the *Sweep range* and type `-30` in the *Start* field and `10` in the *Stop* field.

11. Choose *Linear* for the *Sweep Type* and click on the *Number of Steps* button. Type `10` as the number of steps in the field.

The bottom of the PSS form appears like the one below.



12. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting the 1dB Compression Point

➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

The Waveform window and the Direct Plot form appear.

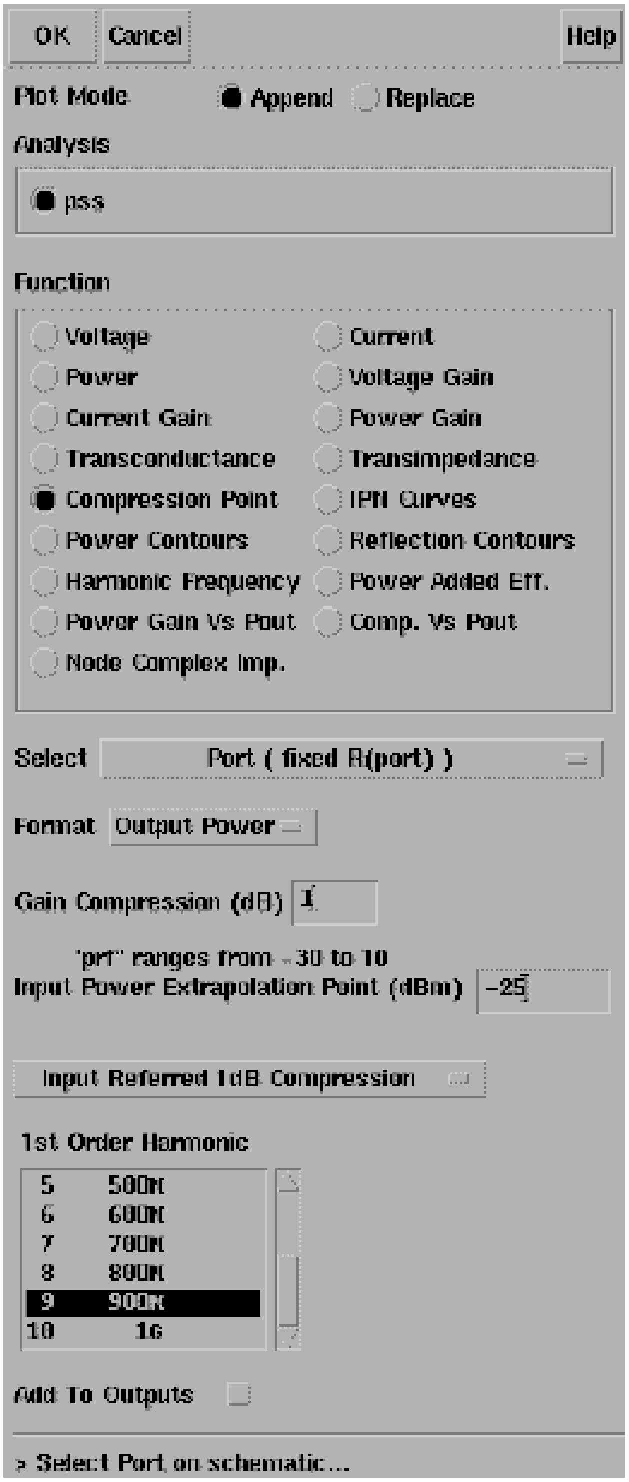
In the Direct Plot form, do the following:

1. Highlight *pss* for *Analysis*.
2. Highlight *Compression Point* for *Function*.
3. Type `-25` for the *Input Power Extrapolation Point (dBm).*

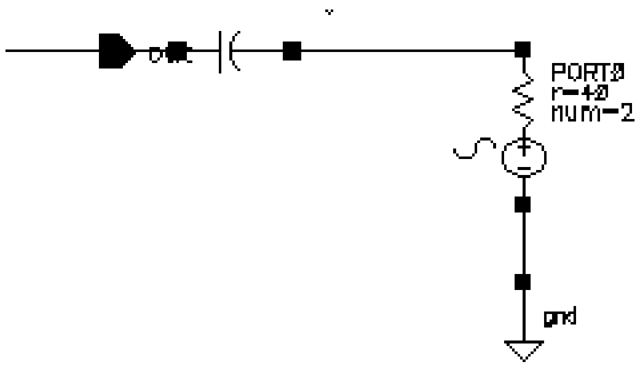
   You learn what value to use for the extrapolation point through experience. If you do not specify a value, the plot defaults to the minimum variable value.

4. Choose *Input Referred 1dB Compression*.
5. Click on *900M* in the *1st Order Harmonic* list box.

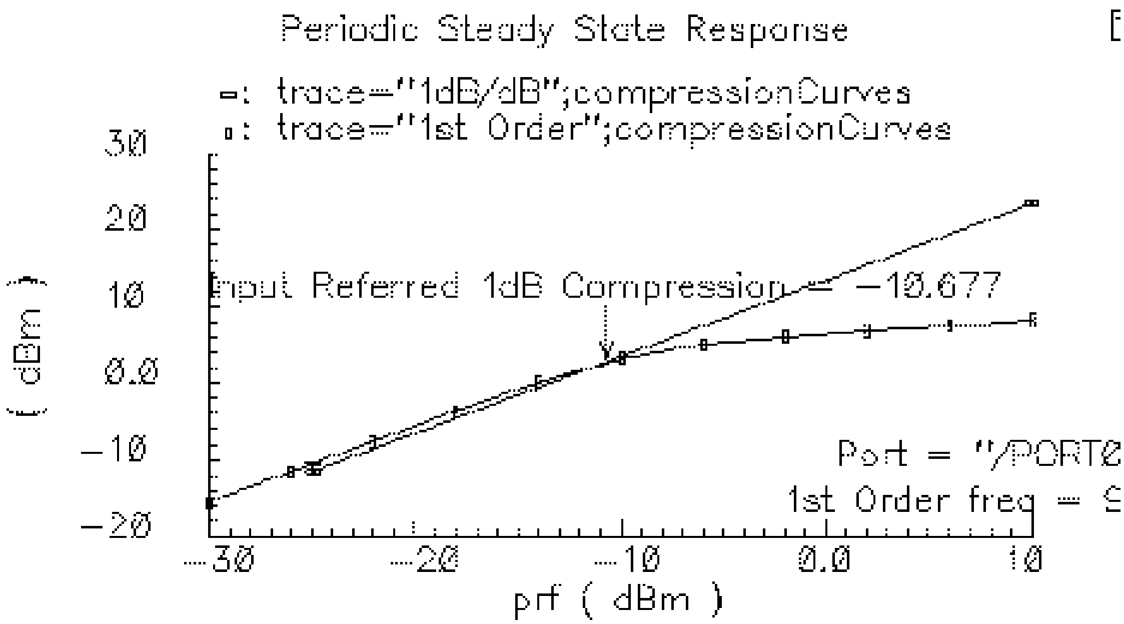
The filled-in form looks like this.



6. In the Schematic Window, click on the `PORT0` output port.



The plot appears in the Waveform window.



The SpectreRF simulator calculates the 1dBCompressionPoint as -10.6337.

Calculating the Third-Order Intercept Point with Swept PSS

The method described here uses only swept PSS analysis. An alternative method, which runs more quickly if the input signals are very close together, uses both swept PSS and PAC analyses and is described in "Third-Order Intercept Measurement with Swept PSS and PAC" on page 296.

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file
4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

Editing the Schematic

1. In the Schematic window, click on the input voltage source and then choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.
2. In the Edit Object Properties form, do the following:
   a. Choose *sine* for the *Source Type*.
   b. Highlight *Display second sinusoid*.
   c. Type `fund2`, or any name you choose, in the *Second frequency name* field.
   d. Type `prf` for the *Amplitude (dBm)* and *Amplitude 2 (dBm)* values.
   e. Type `900M` for the *Frequency* value.
   f. Type `920M` for the *Frequency 2* value.
   g. Be sure the *Amplitude* and *Amplitude2* fields are empty.
3. In the Edit Object Properties form, click *OK*.
4. In the Schematic window, choose *Design – Check and Save*.

Setting up the Swept PSS Analysis

➤ In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

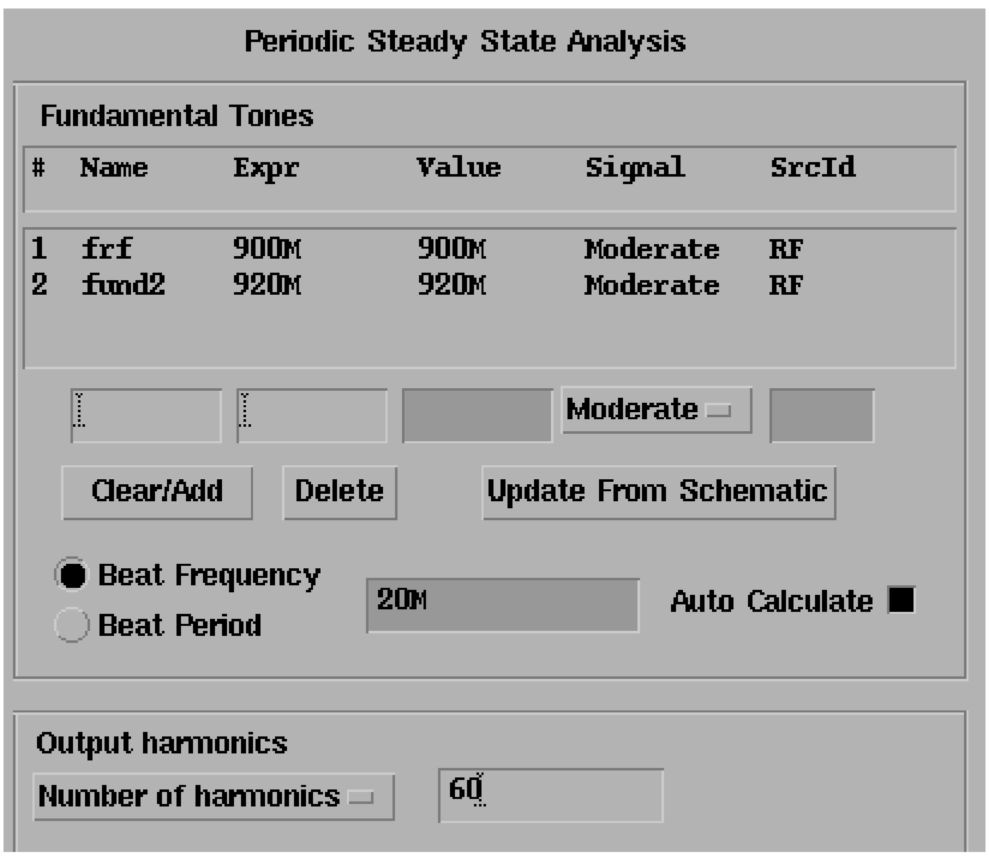
In the Choosing Analyses form, do the following and then click *OK*:

1. In the Choosing Analyses form, click on *pss*.
2. In the *Fundamental Tones* list box, be sure the *Auto calculate* button is highlighted.

   The fundamental frequency, 20M, is displayed.

   The *Beat Frequency* button is highlighted by default.
3. In the *Output harmonics* cyclic field, choose *Number of harmonics* for the *Output harmonics* choice and type in a reasonable value, such as `60`, in the field.

   The top of the PSS form looks like this.

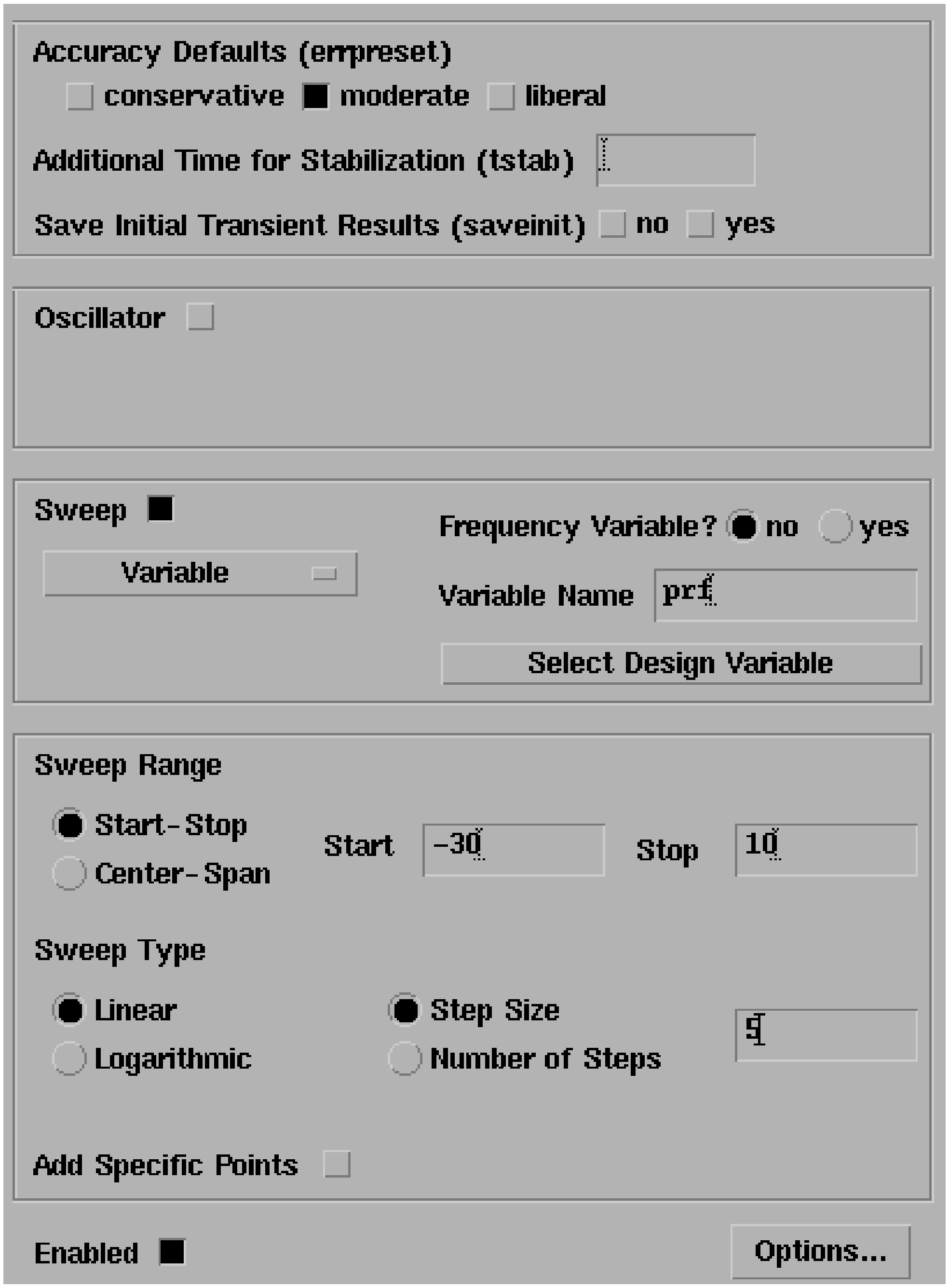


4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
5. Highlight the *Sweep* button.

   The form changes to let you specify data for the sweep.
6. In the *Sweep* cyclic field, choose *Variable*.
7. Click on the *Select Design Variable* button.

   The Select Design Variable form appears.
8. In the Select Design Variable form, highlight *prf* and click *OK*.
9. Choose *Start-Stop* for the *Sweep Range*, and type `-30` and `10` as the *Start* and *Stop* values, respectively, in the fields.
10. Choose *Linear* for the *Sweep Type* and click on the *Step Size* button. Type 5 as the step size.

The bottom of the PSS form looks like this.



Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting the Third-Order Intercept Point

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Waveform window and the Direct Plot form appear.

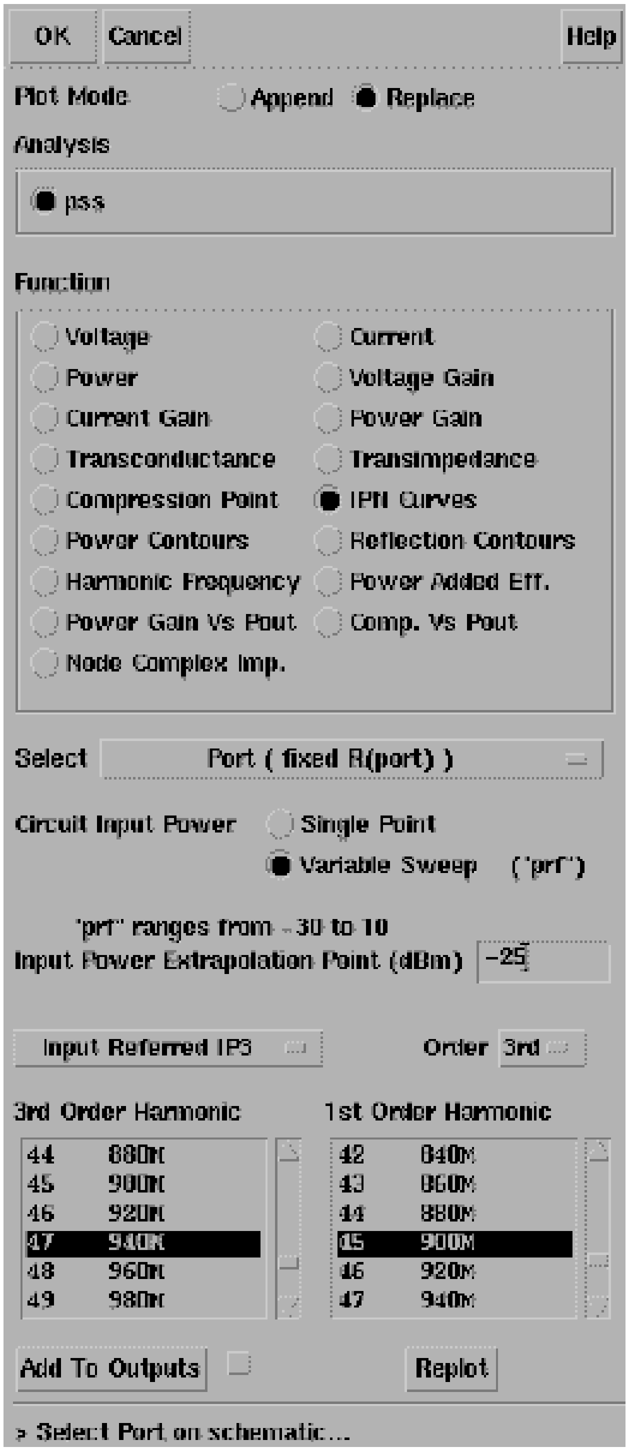
2. Highlight *Replace* for *Plot Mode*.
3. Highlight *pss* for *Analysis*.
4. Highlight *IPN Curves* for *Function*.
5. Highlight *Variable Sweep (“prf”)* for *Circuit Input Power*.
6. Choose *Input Referred IP3*.
7. Type `-25` for the *Input Power Extrapolation Point (dBm)*.

   You learn what value to use for the extrapolation point through experience. If you do not specify a value, the plot defaults to the minimum variable value.

8. In the *order* cyclic field, choose *order 3rd*.
9. In the *3rd Order Harmonic* list box, click on *940M*.
10. In the *1st Order Harmonic* list box, click on *900M*.

    Because the two input frequencies are 900MHz and 920MHz, the two-tone, third-order harmonics are 880MHz and 940MHz. These correspond to the 45th and 47th harmonics.

The filled-in form looks like this.

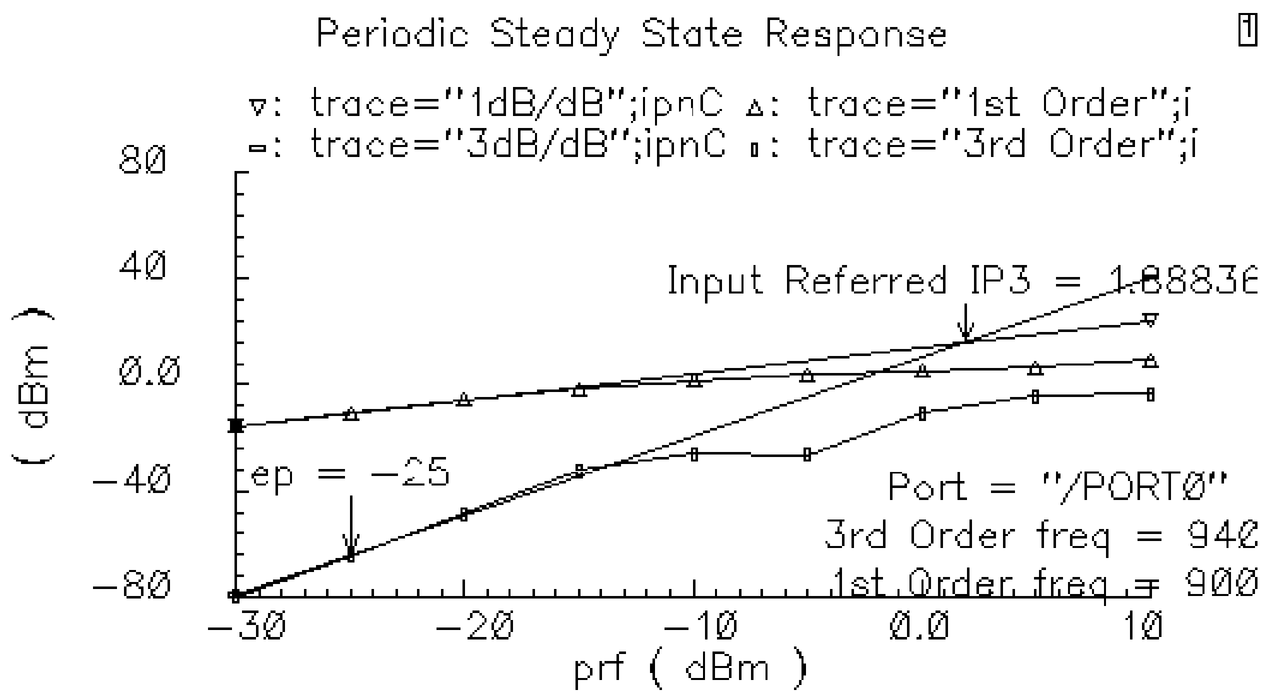


11. In the Schematic window, click on the PORT 0 output port.

The IPN point is plotted in the Waveform window.

12. Click on *Replot.*

The SpectreRF simulator plots the first-order and third-order curves and identifies the intersection of their slopes.



The third-order interpolation point for this example is 1.88836 dB.

Calculating Conversion Gain and Power Supply Rejection with PSS and PXF

Setting Up the Simulation

1. If necessary, open the lnaSimple circuit.
2. If necessary, set up the simulator options.
3. If necessary, specify the full path to the model files in the Model File Set-up form.

   For this simulation you need the `rfModels.scs` model file

4. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

Editing the Schematic

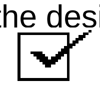
1. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears.

2. In the Schematic window, click on the RFinput voltage source.

   The Edit Object Properties form changes to let you specify data for the input voltage source.

In the Edit Object Properties form, do the following, if necessary:

1. Type `dc` for the *Source Type*.
2. In the Edit Object Properties form, click *OK*.
3. To check and save the design, click on the check mark icon in the top left corner of the Schematic window. 

Setting up the PSS and PXF Analyses

➤ In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

Setting up the PSS Analysis

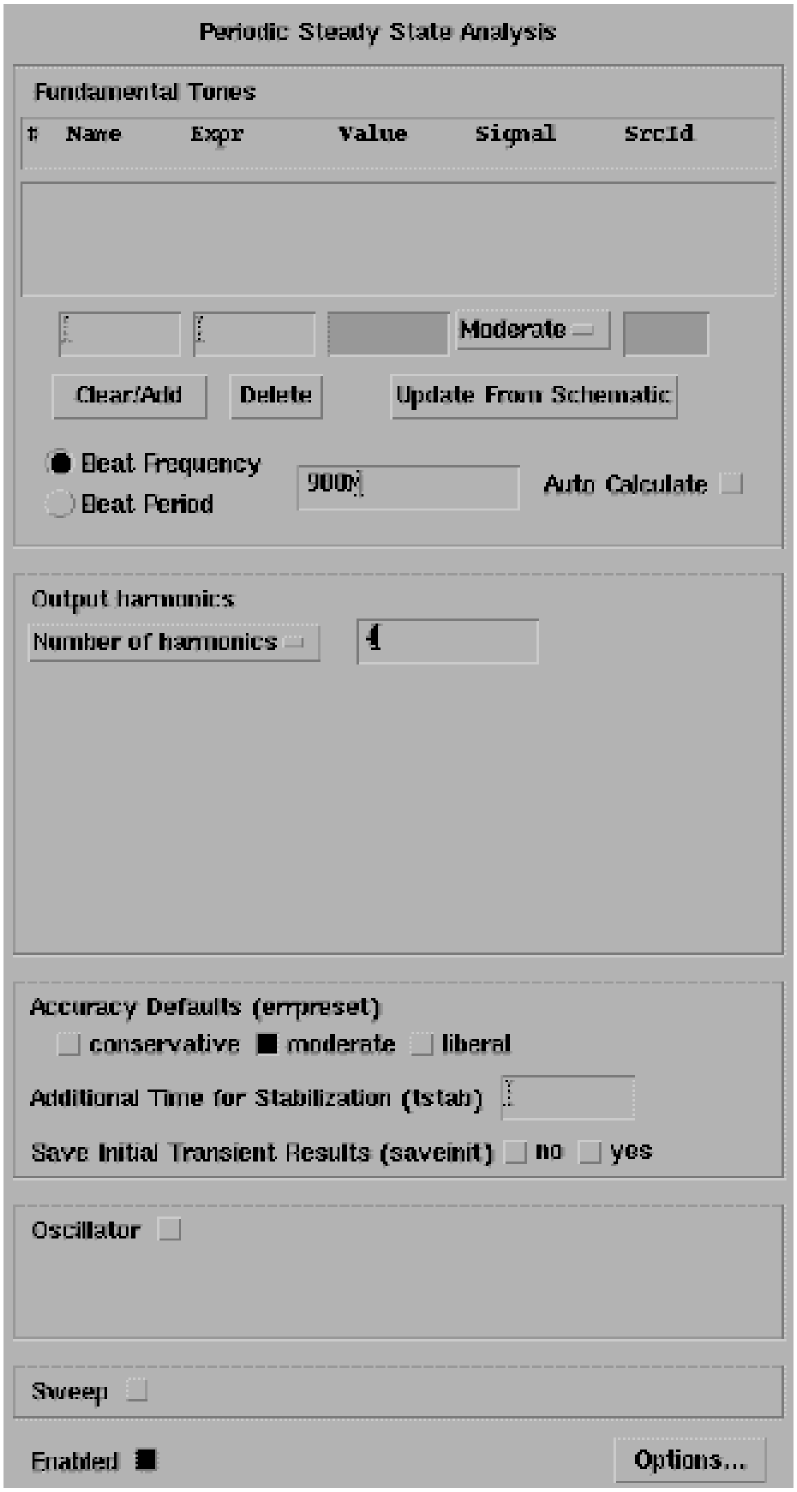
1. In the Choosing Analyses form, highlight *pss*.
2. The *Beat Frequency* button is highlighted by default. Type `900M` in the *Beat Frequency* field.

   Because the source type is dc, you must manually enter the fundamental frequency.

   Be sure the *Auto Calculate* button is not highlighted.

3. In the *Output harmonics* cyclic field, choose *Number of harmonics* and type in a reasonable value, such as `4`, in the field.
4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

The Choosing Analyses form looks like this.



Setting up the PXF Analysis

1. At the top of the Choosing Analyses form, click on *pxf*.

   The form changes to let you specify data for the PXF analysis.

2. In the *Frequency Sweep Range(Hz)* cyclic field, choose *Start-Stop*, and type `10M` and `1.2G` for the *Start* and *Stop* values, respectively in the fields.

3. In the *Sweep Type* cyclic field, choose *Linear* and highlight the *Number of Steps* button.

4. Type `101` for *Number of Steps* in the field.

5. In the *Sidebands cyclic field*, choose *Maximum sideband* for the *Sidebands* choice. Type `0` in the field.

6. Choose *voltage* for the *Output* choice.

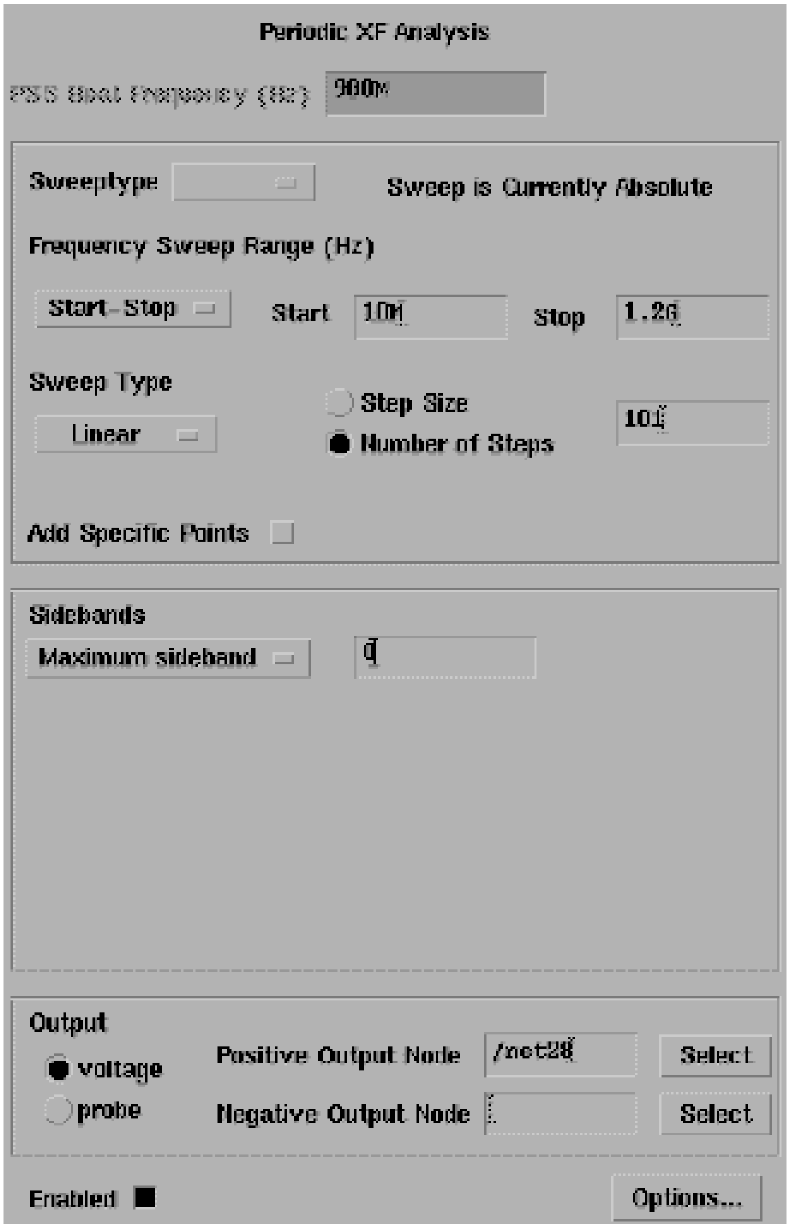
7. Click on the *Select* button for the *Positive Output Node*, and then click on the output signal in the Schematic window.

8. Highlight the *Enabled* button for the small-signal analysis, if necessary.

9. Leave the *Negative Output Node* field empty.

   This field defaults to */gnd!* You can set the *Negative Output Node* to a different value by clicking on the *Negative Output Node Select* button and then clicking on the output node in the schematic.

The pxf section of the form looks like this.



10. In the Choosing Analyses form, click *OK*.

Running the Simulation

1. To run the simulation, choose *Simulation – Netlist and Run* in the Simulation window.

The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation is completed successfully.

Plotting Conversion Gain and Power Supply Rejection

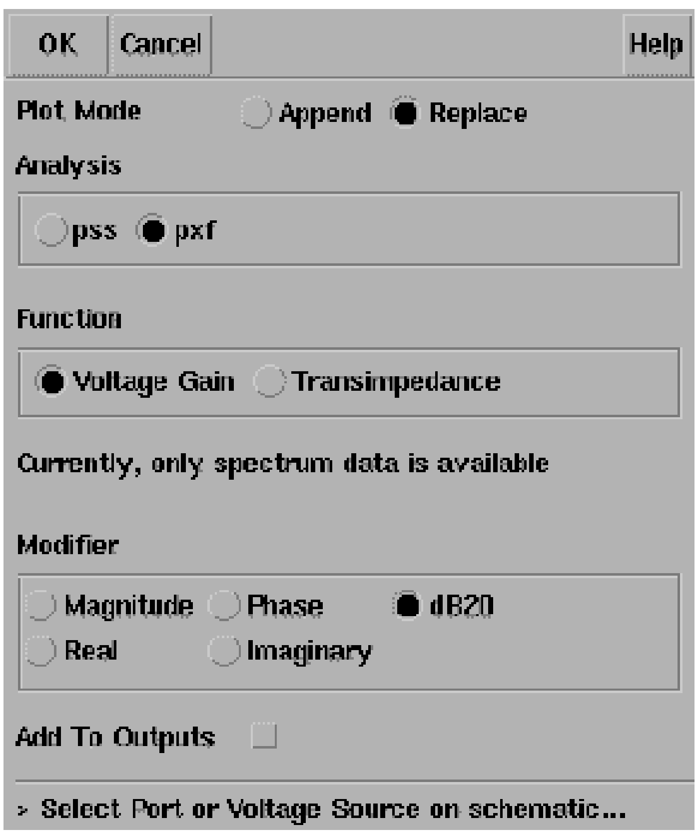
➤ In the Simulation window, choose *Results – Direct Plot – Main Form*.

The Waveform window and the Direct Plot form appear.

In the Direct Plot form, do the following:

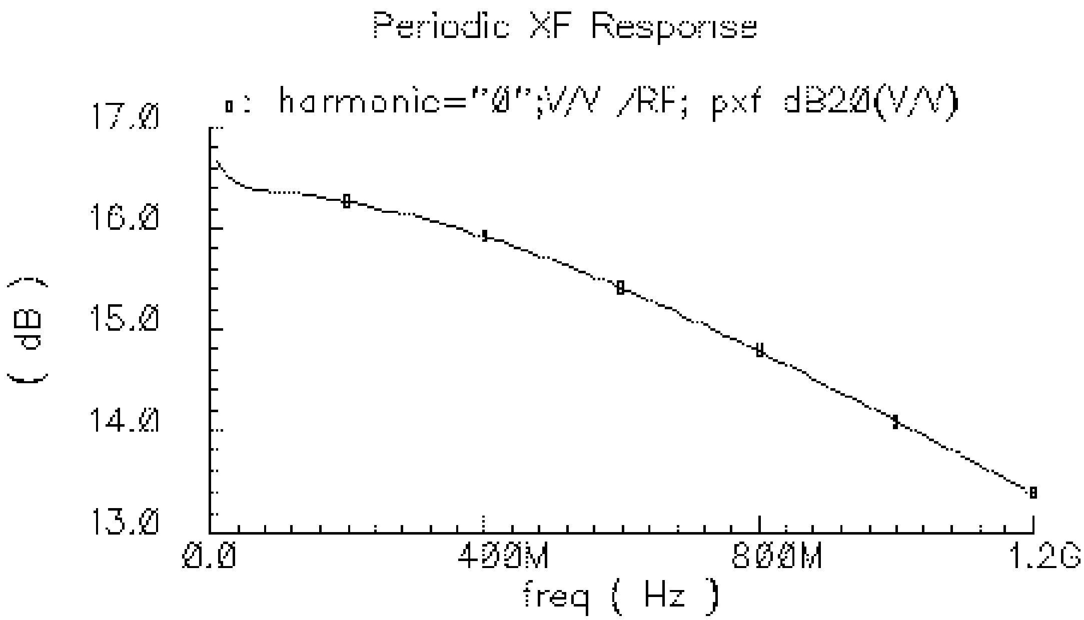
1. *Highlight Replace* for *Plot mode*.
2. *Highlight pxf* for *Analysis*.
3. *Highlight Voltage Gain* for *Function*.
4. *Highlight dB20* for *Modifier*.

   The filled-in form looks like this.



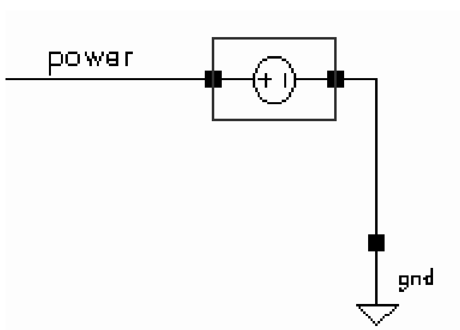
5. In the Schematic window, click on the RF source component.

The transfer function for the LNA is plotted in the Waveform window.

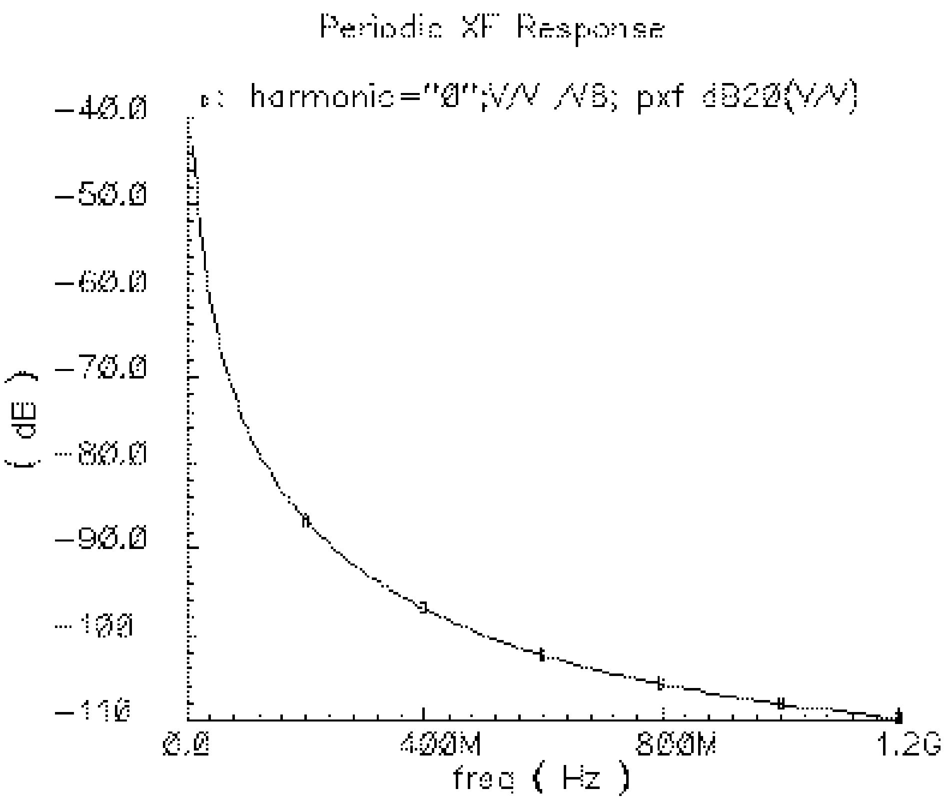


The conversion gain at 900M is approximately 14.4 dBV.

6. In the Direct Plot form, choose *Replace* for *Plot Mode*.
7. To see the power supply rejection, click on the vdb DC voltage source in the schematic.



Be sure to click on the components, not on the wires.



The power supply rejection at 900M is approximately -106 dBV.

7

Modeling Transmission Lines

The Transmission Line Model Generator (LMG) with the SpectreRF simulator models transmission lines in two ways, by both

- Loading quadrature model subcircuits generated by LMG
- Using the LRCG matrixes, also produced by LMG, in *mtline* multi-conductor transmission line model files

LMG supports microstrip, stripline, multi-layer and multi-transmission line systems that include substrate loss. When you use LMG with the new *mtline* external transmission line model, LMG supports the various interconnects used in IC and PCB designs.

The Transmission Line Models: tline3, mline, and mtline

There are several ways to use the *tline3*, *mline*, and *mtline* external transmission line model files in the analog design environment.

The tline3 Model

The *tline3* model is a 3-terminal, single-conductor transmission line model. The *tline3* model can be either a stripline or a microstrip line. The *tline3* model can either use an existing subcircuit produced by LMG or it can invoke LMG in the background during netlisting.

For details on the *tline3* model, reference the following examples.

"Creating a tline3 Macromodel in the LMG GUI" on page 497.

"Using an Existing tline3 Macromodel in the Schematic Flow" on page 507.

"Creating a tline3 Macromodel in the Schematic Flow" on page 515.

The mline Model

The *mline* model is a multi-terminal, multi-conductor transmission line model. The *mline* model can be either a stripline or a microstrip line. It can also handle multi-conductor systems. For n signal lines, the *mline* is a 2n+1 terminal device. It can use an existing subcircuit produced by LMG or it can invoke LMG in the background during netlisting.

For details on the *mline* model, reference the following examples.

“Creating an mline Transmission Line Model Starting LMG From UNIX” on page 520.

“Creating an mline Transmission Line in the Schematic Flow” on page 532.

The mtline Model

The *mtline* model is a multi-layer, multi-conductor transmission line model. The *mtline* model can be a stripline, a microstrip line, a substrate loss interconnect, or a coplanar waveguide. The *mtline* model handles both multi-layer and multi-conductor transmission line systems. For n signal lines, the *mtline* is a 2n+2 terminal device; the ground plane takes 2 terminals. While the *mtline* can use the existing subcircuit produced by LMG, it usually uses an LRCG file which is also produced by LMG. From the analog design environment, an *mtline* model also can invoke the LMG GUI from its CDF.

For details on the *mtline* model, reference the following examples.

“Using LMG and mtline Together in the Analog Design Environment” on page 537.

“Coplanar Waveguide Modeling and Analysis” on page 549

LMG Use Models

The Transmission Line Model Generator (LMG) has two use models. This chapter illustrates by example procedures for both use models with various transmission line instances. You can

- Use the LMG GUI, independent of the analog design environment
- Model transmission lines from within the analog design environment by entering parameter values

Modeling Transmission Lines Using the LMG GUI

To use the LMG GUI standalone, you first open the LMG GUI in any one of several ways

- From the Unix command line, by entering `LMG &` at the Unix prompt.
- By choosing *Tools—RF—Transmission Line Generator* in the Simulation window.
- For the *mtline* model only, you can open LMG from the *mtline* model's CDF.

From the LMG GUI, you manually enter data for each unique transmission line configuration in your design. You manually create and save separate and unique model files for each transmission line segment where there is a unique combination of line type, geometry, parameters, or operating frequency.

This use model builds a 2-D cross-section of the transmission line as you enter parameter values in the graphical window at the top of the LMG GUI as you create the transmission line. Whenever you modify a parameter value, the 2-D cross-section updates to reflect your change.

Begin by entering line-parameter information, the length of the transmission line, and the operating frequencies for the transmission line segment in the LMG GUI. LMG eventually creates a lumped-circuit macromodel for the transmission line segment. You continue by creating lumped-circuit macromodels for each unique transmission line segment in the circuit.

You can share and reuse models that you create and save manually. When you run LMG manually, you create a model file that you can apply to multiple, transmission line instances with identical characteristics. If you have a circuit with a number of identical transmission line instances, you can reuse the same model for each of these instances.You might also want to run LMG manually so you can display a cross-section of the transmission line.

You also enter unique names for the per-unit length LRCG matrix file and the Spectre netlist file that describe the subcircuit. It is helpful to use names that describe the configuration of the particular transmission line. For example, for a stripline with length `1000` and with a maximum frequency of `20GHz` you might use the name `tline_s1000_20G`.

Using your input specifications, LMG automatically extracts parameters and generates a quadrature model subcircuit for each transmission line segment. The quadrature model subcircuits are only accurate for the length and frequency that you specify.

If you modify a transmission line, rerun LMG for the new configuration and update the appropriate instances in the schematic to reflect the new LMG output.

Before you simulate the circuit, you incorporate each model file into the design by specifying its filename as a property on a corresponding *tline3, mline, and mtline* instance. For *mtline* models, you can also specify an LRCG file.

For examples, see the following sections.

“Creating a tline3 Macromodel in the LMG GUI” on page 497.

“Creating an mline Transmission Line Model Starting LMG From UNIX” on page 520.

“Using LMG and mtline Together in the Analog Design Environment” on page 537.

Modeling Transmission Lines Without Using the Visual Interface

This use model is easier to use than the first use model. In this use model, LMG runs within the analog design environment. The software creates every transmission line model instance separately.

This use model includes the following general steps:

1. Open a schematic.
2. Place a *tline3*, *mline*, or *mtline* transmission line instance in the Schematic window.
3. Choose *Edit—Properties—Objects* and select the transmission line instance you placed in the schematic.
4. Describe the transmission line instance by entering parameter values in the Edit Properties form fields. (Units are meters and Hertz only.)
5. For each *tline3*, *mline*, or *mtline* transmission line instance in the design, repeat steps 2, 3, and 4.

For examples, see the following sections.

“Using an Existing tline3 Macromodel in the Schematic Flow” on page 507.

“Creating a tline3 Macromodel in the Schematic Flow” on page 515.

“Using an mline Macromodel in the Schematic Flow” on page 525.

“Creating an mline Transmission Line in the Schematic Flow” on page 532.

Default Values and the initlmg File

When you start LMG, the ./`.initlmg` file is checked for default values. If the ./`.initlmg` file does not exist, then LMG uses the following default values:

```
lmgLineType = Microstrip
lmgLengthUnit = um
lmgFreqUnit = GHz
lmgNumberofConductors = 1
lmgDielectricPermittivity = 10
```

```
lmgDielectricThickness = 400
lmgConductorWidth = 300
lmgConductorDistances = For multiconductors, the width of the first conductor
lmgConductorThickness = 100
lmgConductorHeight = For multiconductors, the height of the first conductor
lmgConductorLength = 5000
lmgConductivity = 5.6e7
lmgMaxFreq = 10
lmgSubCircuitName = tline
```

Tline3 Transmission Line Modeling Examples

This section presents a series of examples showing how to create a *tline3* transmission line model using the LMG GUI. Because the procedures for creating microstrip and stripline transmission lines are very similar, only a stripline example is shown here.

Creating a tline3 Macromodel in the LMG GUI

This example illustrates how to create a *tline3* macromodel from the LMG GUI while you are using the analog design environment.

This example uses the circuit `tline3oscRFlmg` from your editable copy of the `rfExamples` library (*my_rfExamples* here). You need to open this circuit to set up the analog design environment. The *my_rfExamples* library also includes other sample circuits used in this manual. If you need assistance setting up or accessing your editable copy of the `rfExamples` library, refer to Chapter 3, “Setting Up for the Examples.”

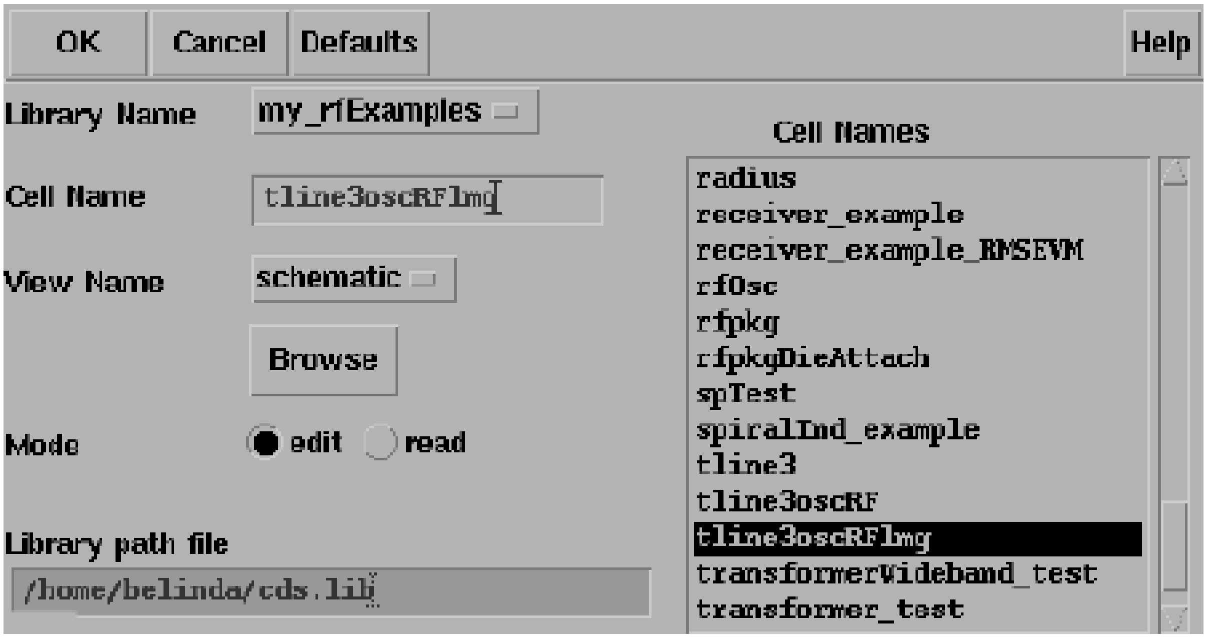
Opening the tline3oscRFlmg Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

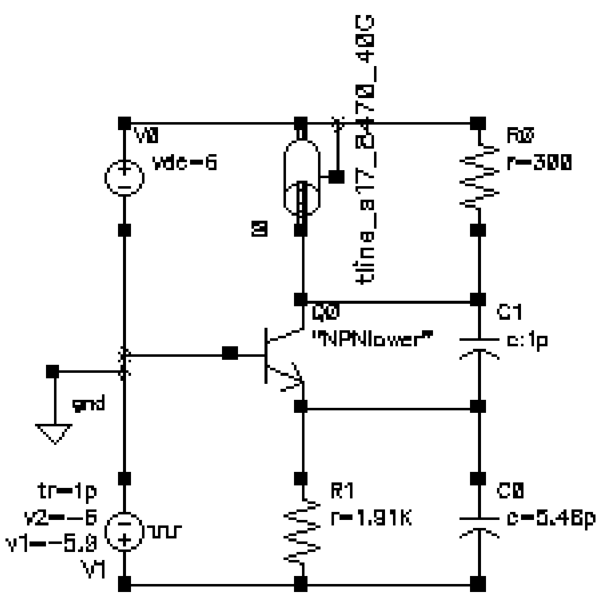
2. In the Open File form, choose *my_rfExamples* in the *Library Name* cyclic field and choose *tline3oscRFlmg* in the *Cell Names* list box.

The completed Open File form appears like the one below.



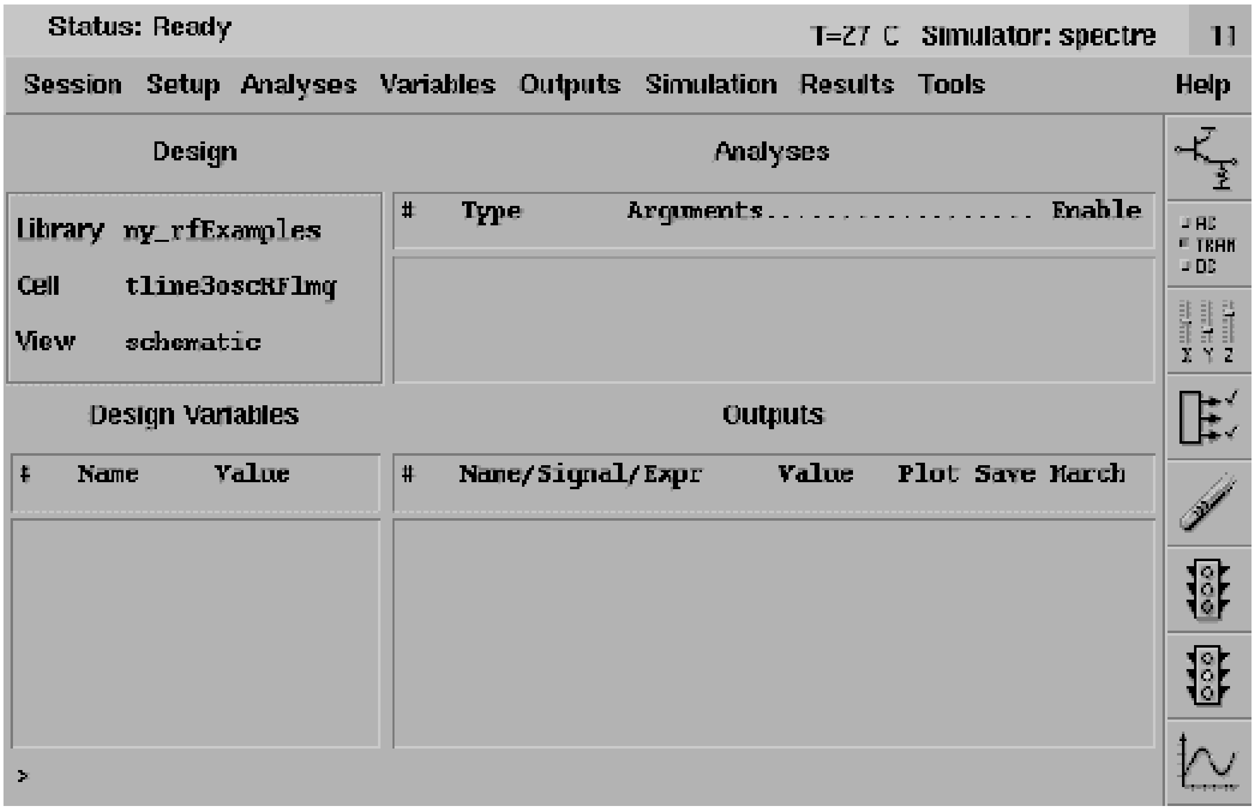
3. Click *OK*.

The Schematic window for the `tline3oscRFlmg` circuit appears.



4. In the Schematic window, choose *Tools—Analog Environment*.

The Simulation window opens.



Opening the LMG GUI

This example assumes that this is the first run of LMG and that the file `./.initlmg` does not yet exist. Consequently, LMG uses internal default values.

Note: When you start LMG by typing `lmg &` at the UNIX prompt, the LMG GUI displays directly without first displaying the Transmission Line Modeler form and the path to the model directory as shown here in this example.

To model a *tline3* transmission line using the LMG GUI, perform the following steps:

1. Choose *Tools–RF–Transmission Line Modeler* in the Simulation window.

The Transmission Line Modeler form appears displaying the path to the directory where the model will be created.

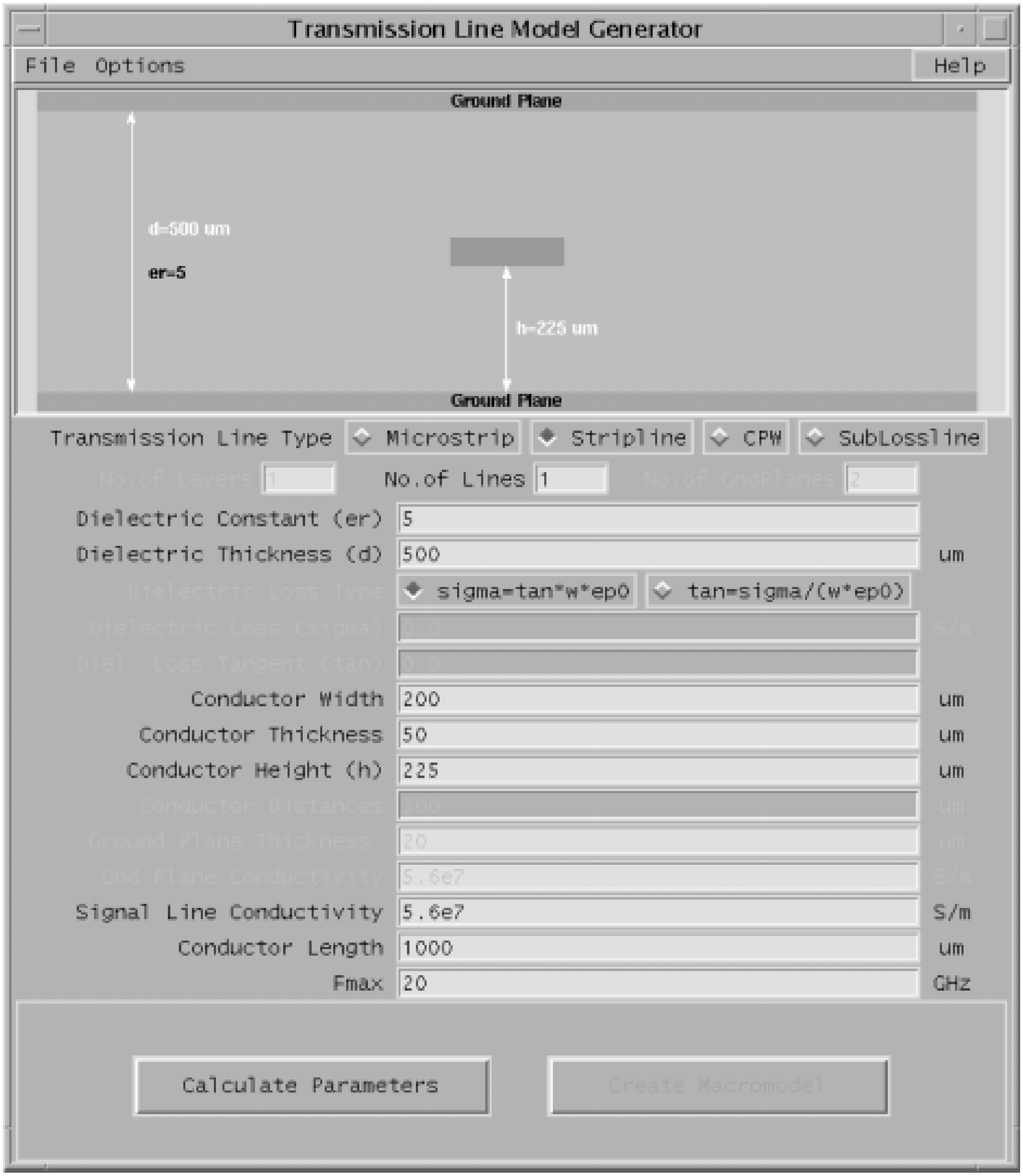
Note: Make a note of the location of this model as you will need to enter the absolute path to this macromodel in another example, “Using an Existing tline3 Macromodel in the Schematic Flow” on page 507.

The path is similar to the following:

`/home/belinda/simulation/tline3oscRFlmg/spectre/schematic/netlist`

2. Click *OK* to display the LMG GUI.

The LMG GUI appears similar to the one below.



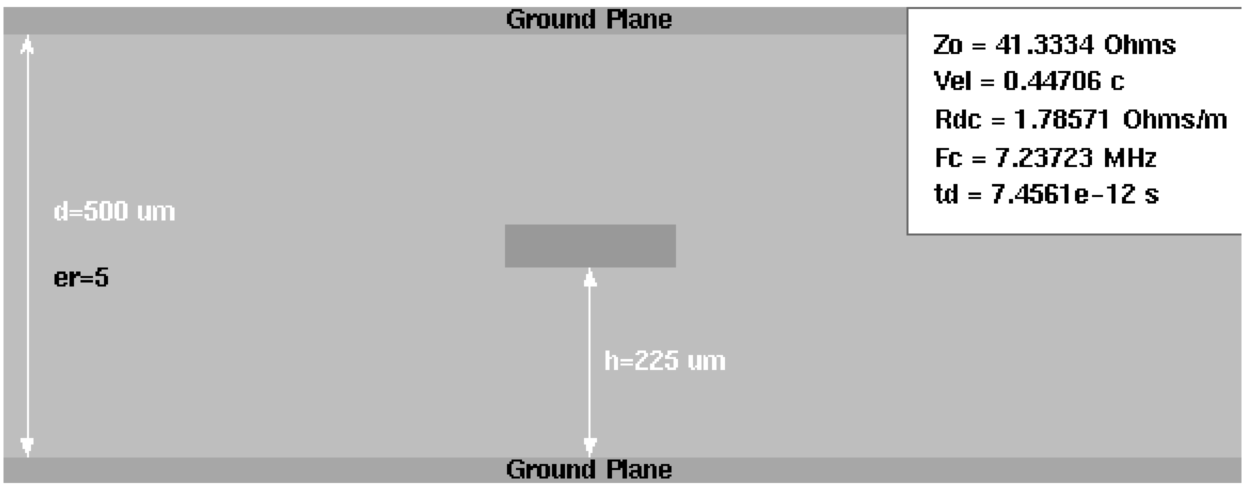
3. Choose *Options—Model Type—Lossless* to model a lossless transmission line.

4. Choose *Options—Subckt Format—Spectre*.

   This choice assumes that you are running your simulation in the analog design environment.

5. Choose *Options—Length Unit—um* to specify *um* as the length unit.
6. Choose *Options—Freq. Unit—GHz* to specify *GHz* as the frequency unit.
7. Click *Stripline* to create a stripline model.

   The display section of the form now shows a stripline transmission line similar to the one below.



8. Specify the following values in the data entry fields:

   No. of Lines: `1`

   Dielectric Constant (er): `5`

   Dielectric Thickness (d): `500` *um*

   *sigma=tan*w*ep0*

   Conductor Width: `200` *um*

   Conductor Thickness: `50` *um*

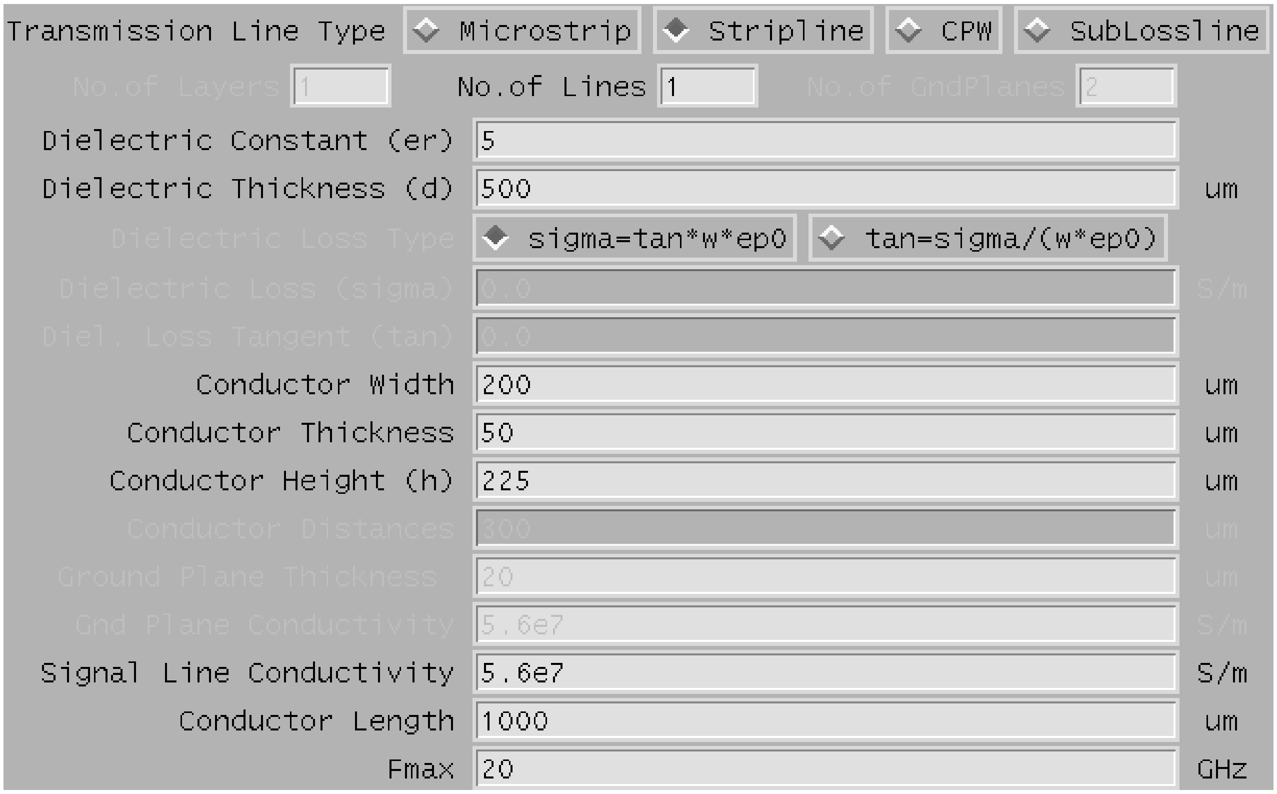
   Conductor Height (h): `225` *um*

   Signal Line Conductivity: `5.6e7` *S/m*

   Conductor Length: `1000` *um*

   Fmax: `20` *GHz*

The data entry fields appear like the ones below.



9. Click the *Calculate Parameters* button.

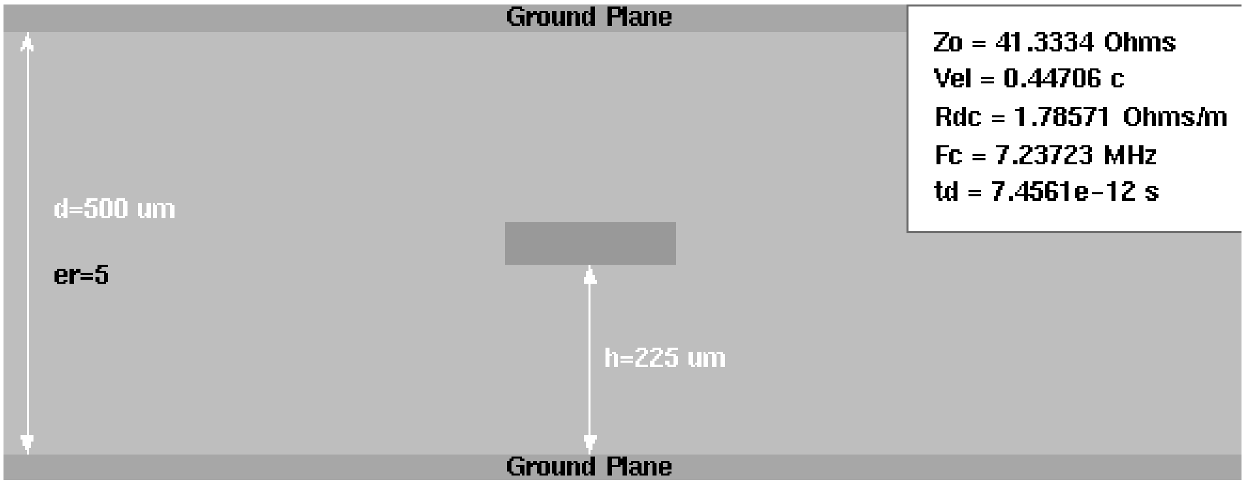
This calculates the per unit length LRCG matrixes and the following model parameter values:

- Characteristic impedance, Z_0 of 41.3334 Ohms
- Propagation velocity, Vel of 0.44706 c
- DC resistance per meter, R_{dc} of 1.78571 Ohms/m
- Corner frequency of the conductor, F_c of 7.23723 MHz
- Time delay of the transmission line, td of $7.4561e^{-12}$ s

The display section changes to reflect the values you specified. The calculated parameter values display in the top right corner of the display section.

Note: The calculated parameter values display in LMG only when the *No. of Lines* field is 1. (The *No. of Lines* field is in the second line of data entry fields from the top of the

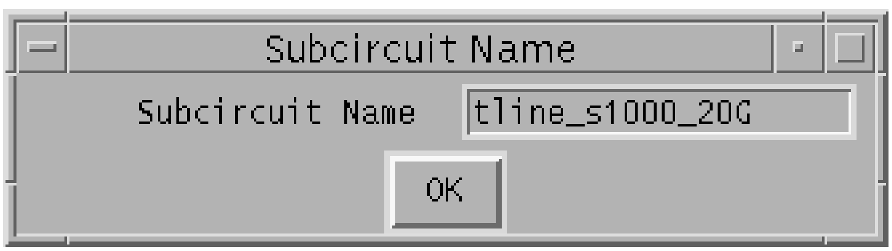
GUI.)



At this point, the *Create Macromodel* button at the bottom of the GUI is enabled.

10. Choose *File—Subcircuit Name*.

The Subcircuit Name form appears.

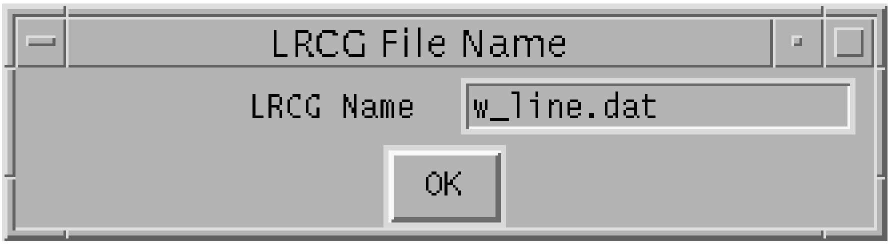


11. Type `tline_s1000_20G` in the *Subcircuit Name* field and click *OK*.

This model name provides descriptive information about the model type, its length, and the maximum frequency of the subcircuit.

12. Choose *File—LRCG file Name*.

The LRCG File Name form appears.

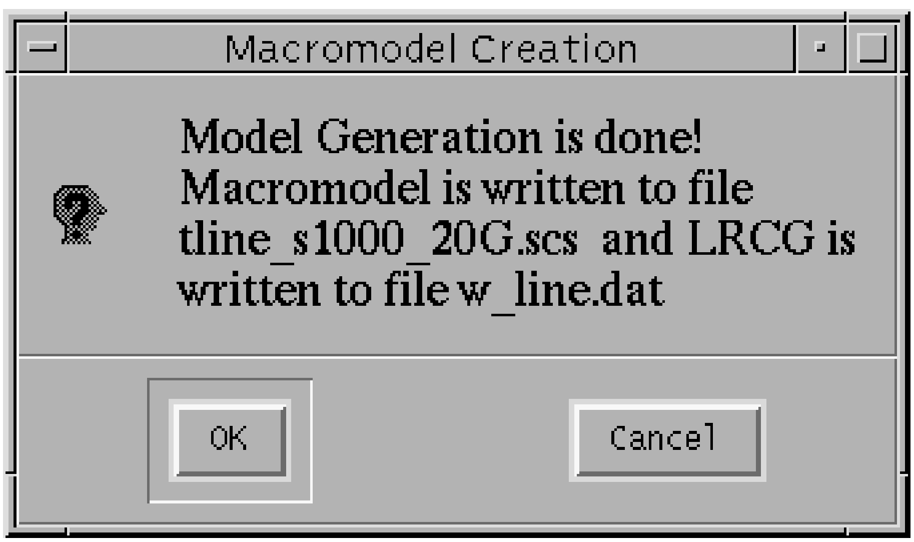


13. Type `w_line.dat` in the *LRCG Name* field and click *OK*.

This is the name of the file containing the LRCG matrixes in a format consistent with the *tline3* model.

14. Click *Create Macromodel*, to create the macromodel and the LRCG data file.

The Macromodel Creation form displays the names of the macromodel netlist file and the LRCG data file.



For this example, the final results are written to the following two files

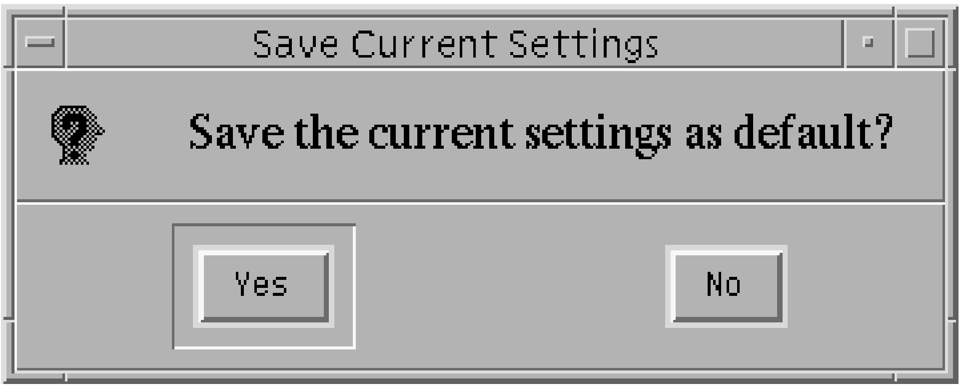
- ❑ A macromodel netlist file with the name you specified and a `scs` suffix—`tline_s1000_20G.scs`.
- ❑ An LRCG file with the name `w_line.dat`.

Note: Make a note of the name of this model as you will need to enter the model name in another example, "Using an Existing tline3 Macromodel in the Schematic Flow" on page 507.

Later, you will use the macromodel file name as a property of a *tline3* symbol in the analog design environment.

15. Click *OK* in the Macromodel Creation form.

16. To save the current settings as default values for subsequent LMG runs, choose *File—Save Current Settings*.



The current settings are saved in file ./`.initlmg`. LMG reads this file in subsequent start-ups. The ./`.initlmg` file contains all the quantities you can change. As long as you preserve its format, you can edit this file directly to choose the LMG startup defaults

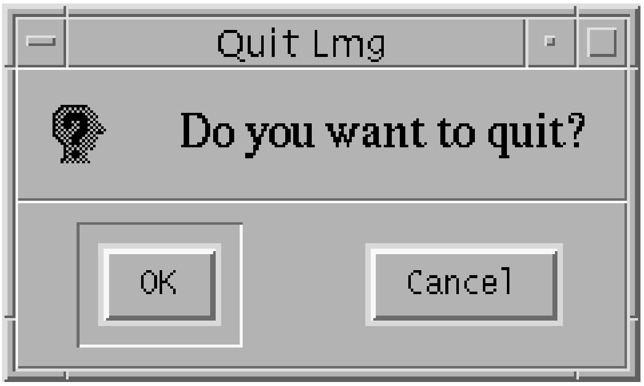
The contents of the ./`.initlmg` file following this example are as follows.

```
lmgLineType = Stripline
lmgLengthUnit = um
lmgFreqUnit = GHz
lmgDielectricPermittivity = 5
lmgDielectricThickness = 500
lmgConductorWidth = 200
lmgConductorThickness = 50
lmgConductorHeight = 225
lmgConductorLength = 1000
lmgConductivity = 5.6e7
lmgMaxFreq = 20
lmgSubCircuitName = tline_s1000_20G
```

17. Click *Yes*.

18. At the end of the run, choose *File—Quit* to exit LMG.

The following form displays



19. Click *OK*.

LMG closes.

Using an Existing tline3 Macromodel in the Schematic Flow

This example illustrates how to call an existing *tline3* macromodel from the schematic flow while you are using the analog design environment.

This example uses the circuit `tline3oscRFlmg` from your editable copy of the `rfExamples` library (*my_rfExamples* here). This library also includes other sample circuits used in this manual. If you need assistance setting up or accessing your editable copy of the `rfExamples` library, refer to Chapter 3, “Setting Up for the Examples.” You will also need the absolute path to the *tline3* macromodel you created in the first example.

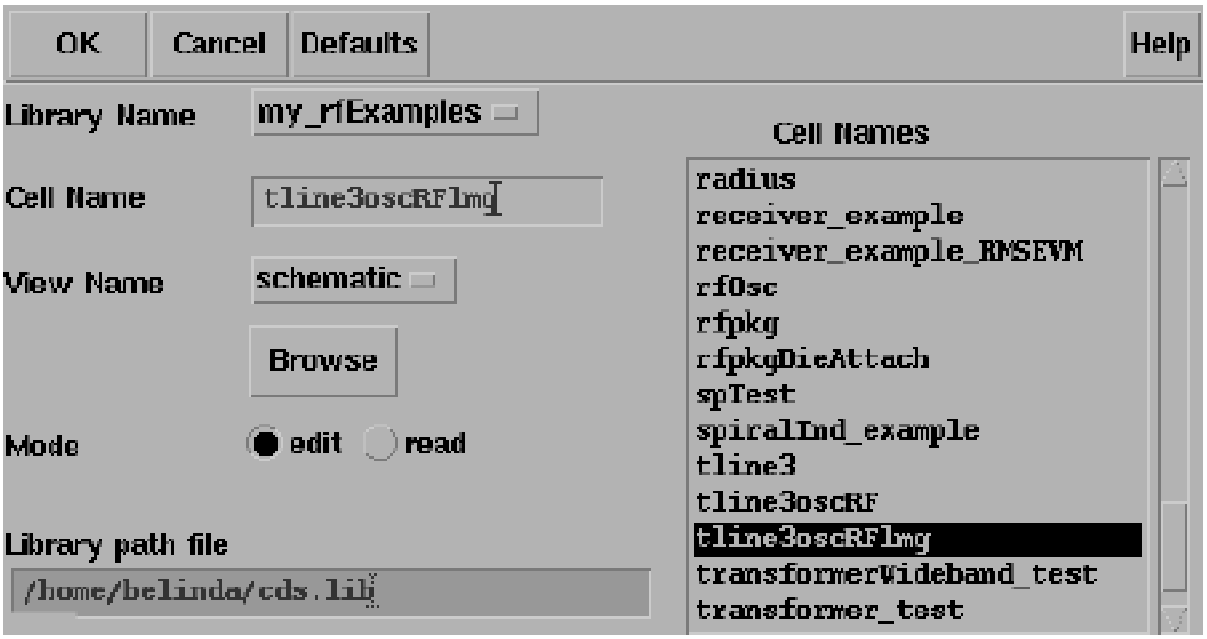
Opening the tline3oscRFlmg Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

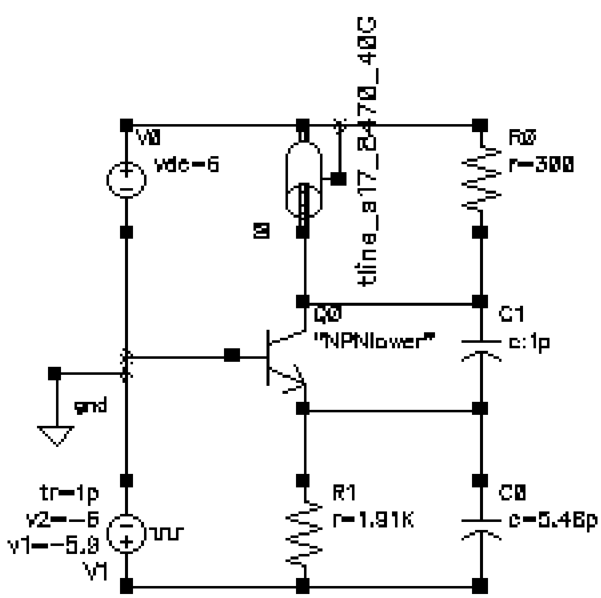
2. In the Open File form, choose *my_rfExamples* in the *Library Name* cyclic field and choose *tline3oscRFlmg* in the *Cell Names* list box.

The completed Open File form appears like the one below.



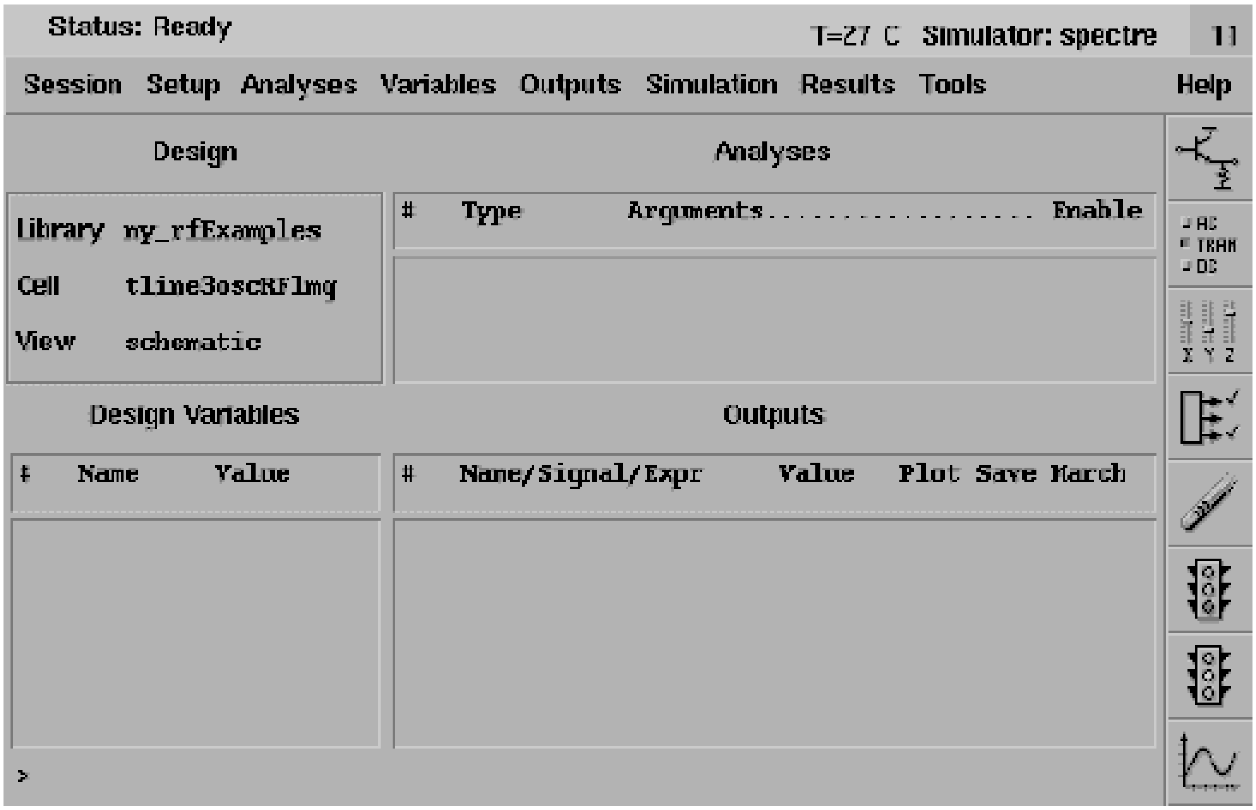
3. Click *OK*.

The Schematic window for the `tline3oscRFlmg` circuit appears.



4. In the Schematic window, choose *Tools—Analog Environment*.

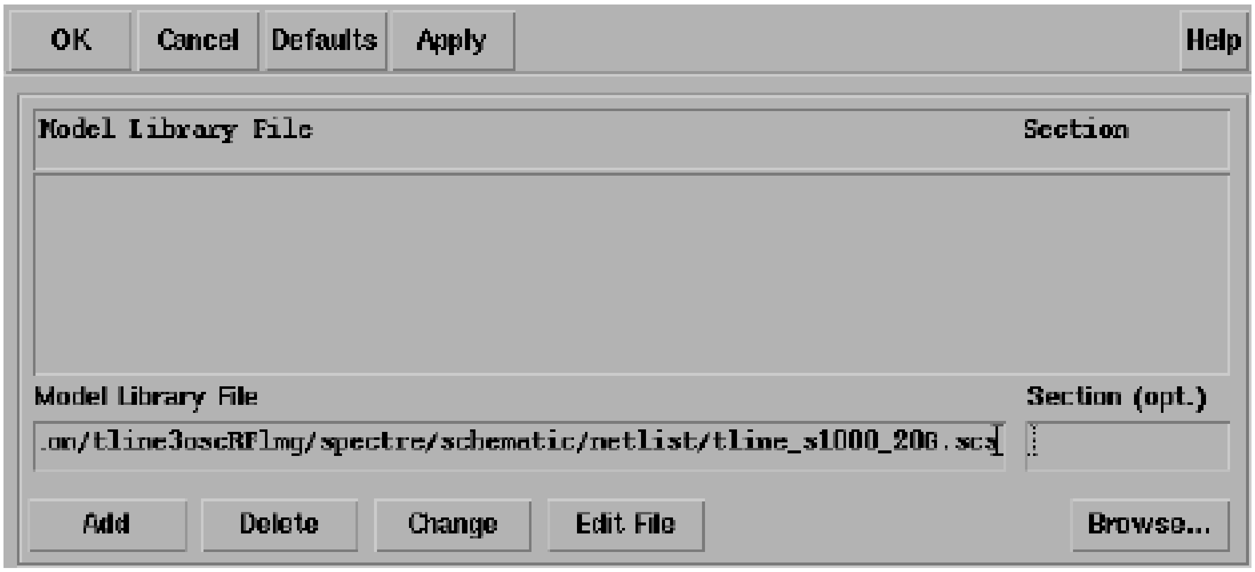
The Simulation window opens.



Setting Up the Model Path

1. In the Simulation window, choose *Setup—Model Libraries*.

The Model Library Setup form appears.



2. In the *Model Library File* field, type the full, absolute path to the model file including the model file name, `tline_s1000_20G.scs`.
    - ❑ The absolute path is displayed in the Transmission Line Modeler form in the first example.
    - ❑ The model name is the name of the model generated during the first example. The model name is displayed in the Subcircuit Name form in the first example.

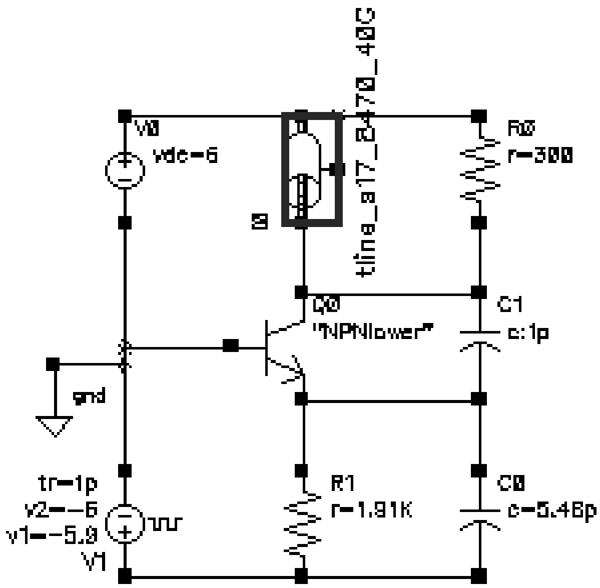
    For example, type the following in the *Model Library File* field:

    ```
    /home/belinda/simulation/tline3oscRFlmg/spectre/schematic/netlist/
    tline_s1000_20G.scs
    ```
3. Click *Add*.
4. Click *OK*.

Associating the External Model File with the Transmission Line Component

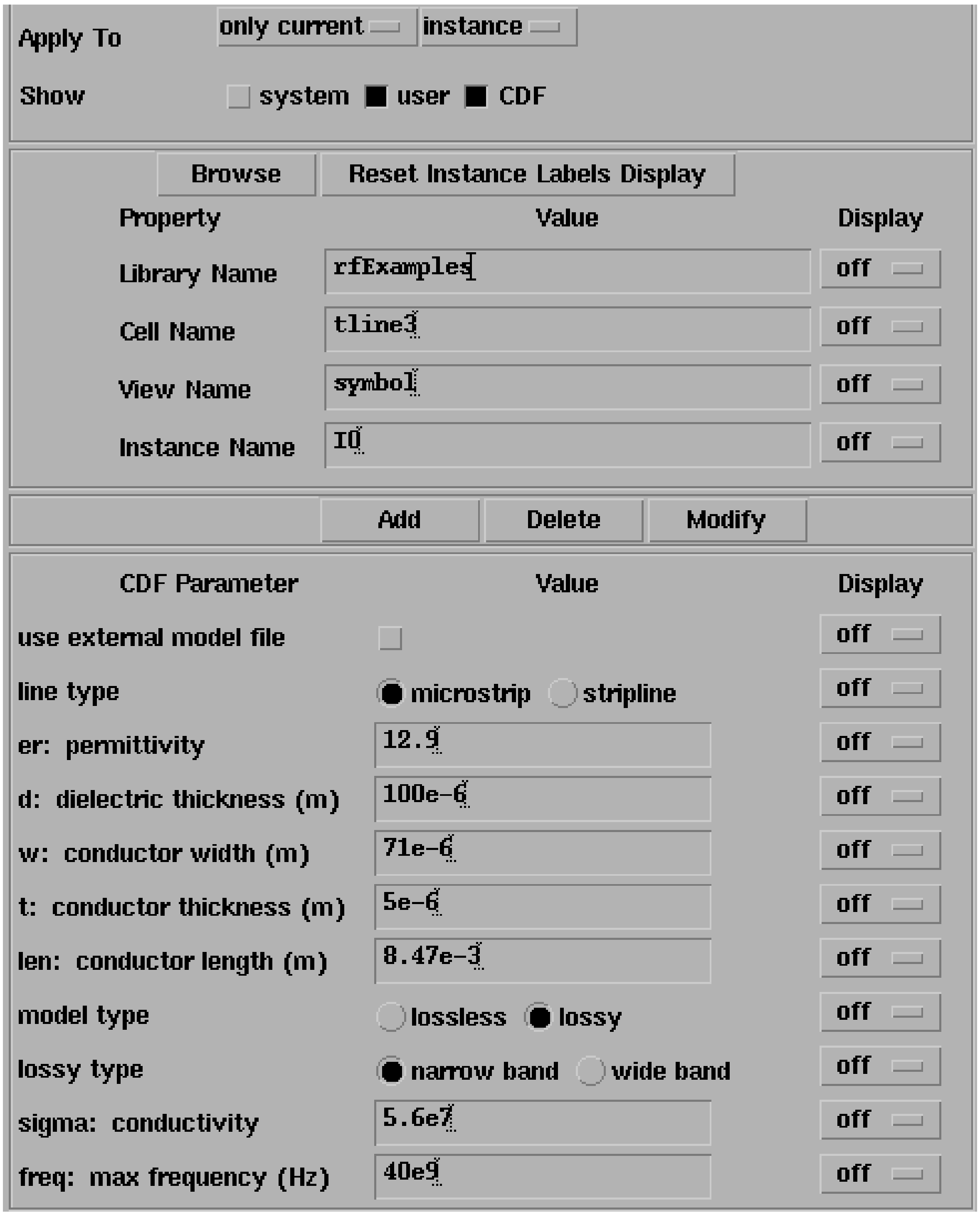
1. In the Schematic window, click on the transmission line component *tline_s17_8470_40G* near the top of the schematic.

The transmission line is now selected.



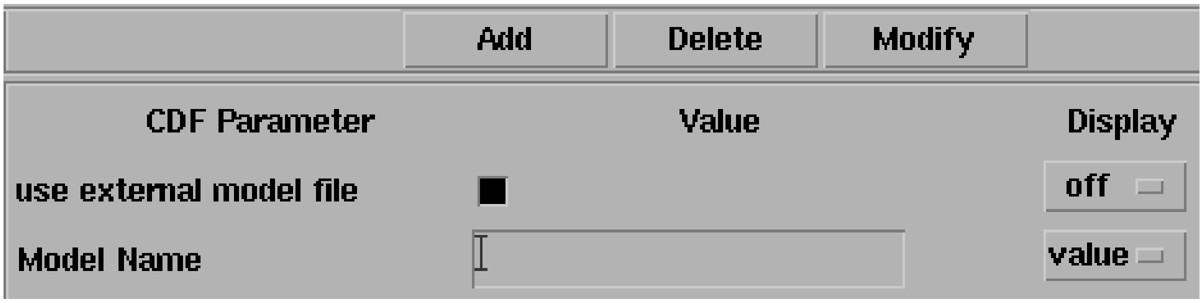
2. With the transmission line selected in the Schematic window, choose *Edit—Properties —Objects* to open the Edit Object Properties form for the selected *tline_s17_8470_40G* transmission line component.

The *Edit—Properties—Objects* form appears similar to the following.



3. In the Edit Object Properties form, highlight *use external model file*.

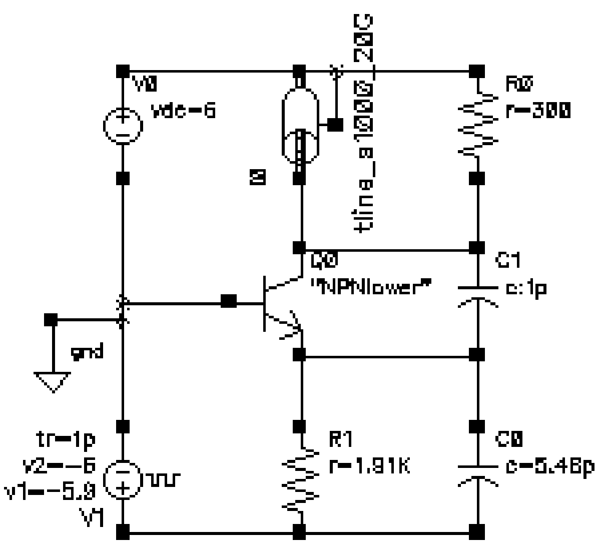
When you highlight *use external model file*, the form changes to let you specify only the *Model Name*.



4. In the *Model Name* field, type `tline_s1000_20G`, the name of the existing *tline* macromodel you created using LMG.

5. In the Edit Object Properties form, click *OK*.

The component name in the Schematic window changes to reflect the new *tline* macromodel name.



6. Check and Save the `tline3oscRFlmg` schematic.

You can now set up and run a simulation using the new *tline* macromodel in the analog design environment.

Resolving A Possible Error Message for a tline3 Model File

In some circumstances the analog design environment might issue an error message stating that a `tline3` subckt has four terminals, while in a schematic it is a 3-terminal device.

To solve this problem, you can open and edit the subcircuit file with a text editor of your choice.For example, the `tline_s1000_20G.scs` file looks like the following.

```
//
// Gaussian quadrature model to transmission line.
// Transmission line type: Stripline
// Number of signal conductors: 1
subckt tline_s1000_20G (in0 out0 ref refequal)
  l0_0 (in0 n1_0) inductor l=1.445788e-11
  l1_0 (n1_0 n2_0) inductor l=5.666485e-11
  l2_0 (n2_0 n3_0) inductor l=8.297930e-11
  l3_0 (n3_0 n4_0) inductor l=8.297930e-11
  l4_0 (n4_0 n5_0) inductor l=5.666485e-11
  l5_0 (n5_0 out0) inductor l=1.445788e-11
  c0_0 (n1_0 ref) capacitor c=2.137783e-14
  c1_0 (n2_0 ref) capacitor c=4.318660e-14
  c2_0 (n3_0 ref) capacitor c=5.133076e-14
  c3_0 (n4_0 ref) capacitor c=4.318660e-14
  c4_0 (n5_0 ref) capacitor c=2.137783e-14
ends tline_s1000_20G
```

To edit the `tline_s1000_20G.scs` subcircuit file with the text editor, use the following steps.

1. Find `refequal` and delete it from the `tline_s1000_20G.scs` file.

```
//
// Gaussian quadrature model to transmission line.
// Transmission line type: Stripline
// Number of signal conductors: 1
subckt tline_s1000_20G (in0 out0 ref)
  l0_0 (in0 n1_0) inductor l=1.445788e-11
  l1_0 (n1_0 n2_0) inductor l=5.666485e-11
```

```
  l2_0 (n2_0 n3_0) inductor l=8.297930e-11
  l3_0 (n3_0 n4_0) inductor l=8.297930e-11
  l4_0 (n4_0 n5_0) inductor l=5.666485e-11
  l5_0 (n5_0 out0) inductor l=1.445788e-11
  c0_0 (n1_0 ref) capacitor c=2.137783e-14
  c1_0 (n2_0 ref) capacitor c=4.318660e-14
  c2_0 (n3_0 ref) capacitor c=5.133076e-14
  c3_0 (n4_0 ref) capacitor c=4.318660e-14
  c4_0 (n5_0 ref) capacitor c=2.137783e-14
ends tline_s1000_20G
```

2. Save the edited `tline_s1000_20G.scs` file.

After you edit the model file, you can perform a simulation using the `tline_s1000_20G.scs` model file without errors.

Creating a tline3 Macromodel in the Schematic Flow

This example illustrates how to create a *tline3* transmission line macromodel from the schematic flow while you are using the analog design environment.

This example follows the second use model in that it does not use the GUI. You can use this procedure if you do not need to see the two-dimensional cross-section display that the LMG GUI provides. This example creates a transmission line that is identical to the one created in the previous example "Using an Existing tline3 Macromodel in the Schematic Flow" on page 507.

This example uses the circuit `tline3oscRFlmg` from your editable copy of the `rfExamples` library, *my_rfExamples*. This library also includes other sample circuits used in this manual. If you need assistance setting up or accessing the `rfExamples` library, refer to Chapter 3, "Setting Up for the Examples."

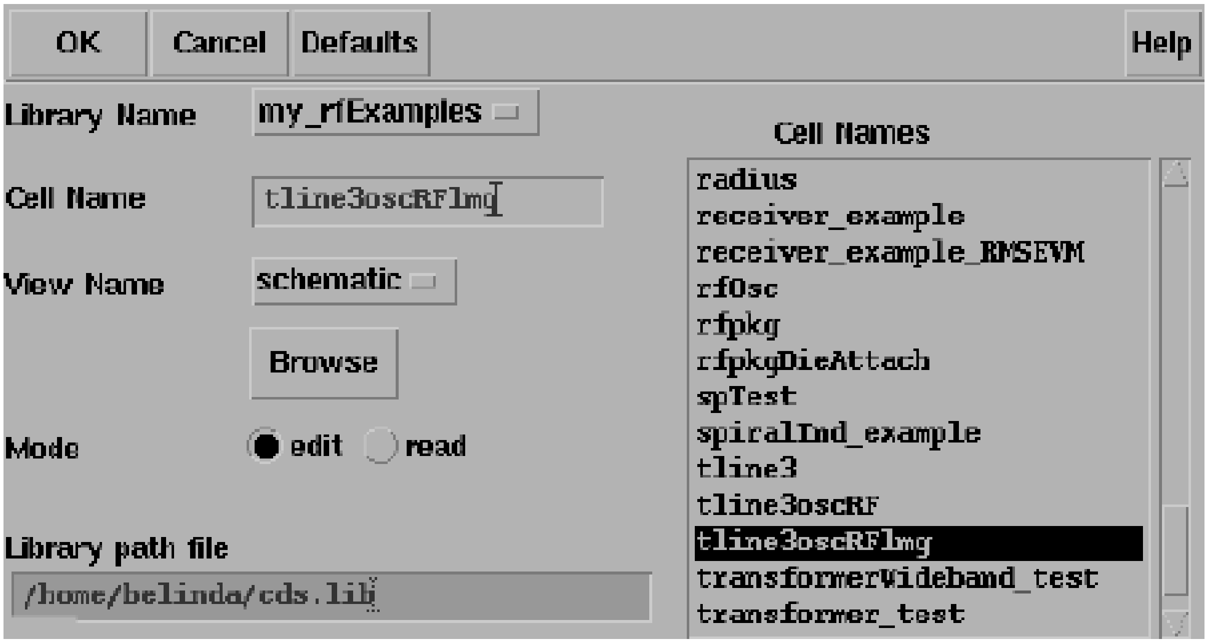
Opening the tline3oscRFlmg Circuit in the Schematic Window

1. In the CIW, choose *File—Open*.

   The Open File form appears.

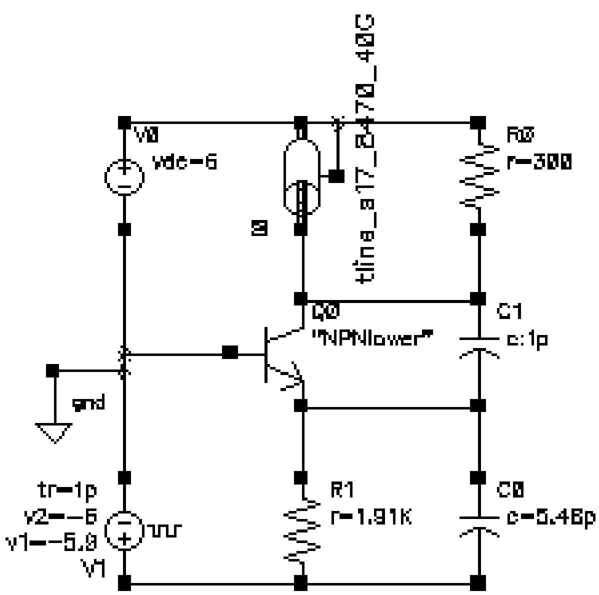
2. In the Open File form, choose your editable copy of the *rfExamples* library, in this case *my_rfExamples* in the *Library Name* cyclic field and choose *tline3oscRFlmg* in the *Cell Names* list box.

The completed Open File form appears like the one below.



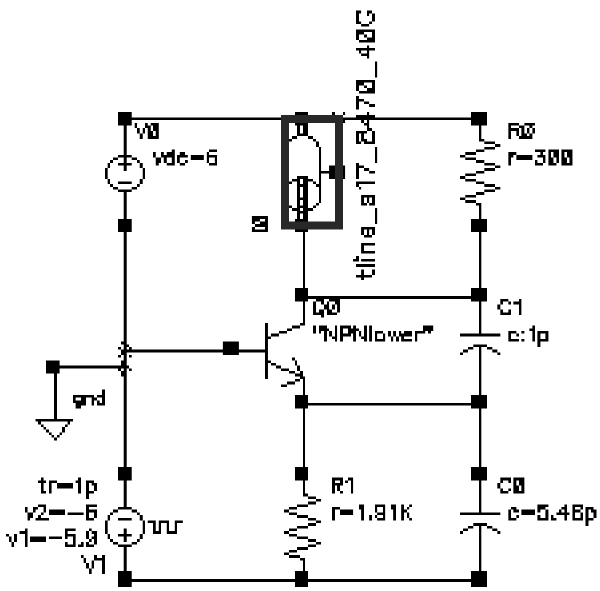
3. Click *OK*.

The Schematic window for the `tline3oscRFlmg` circuit appears.



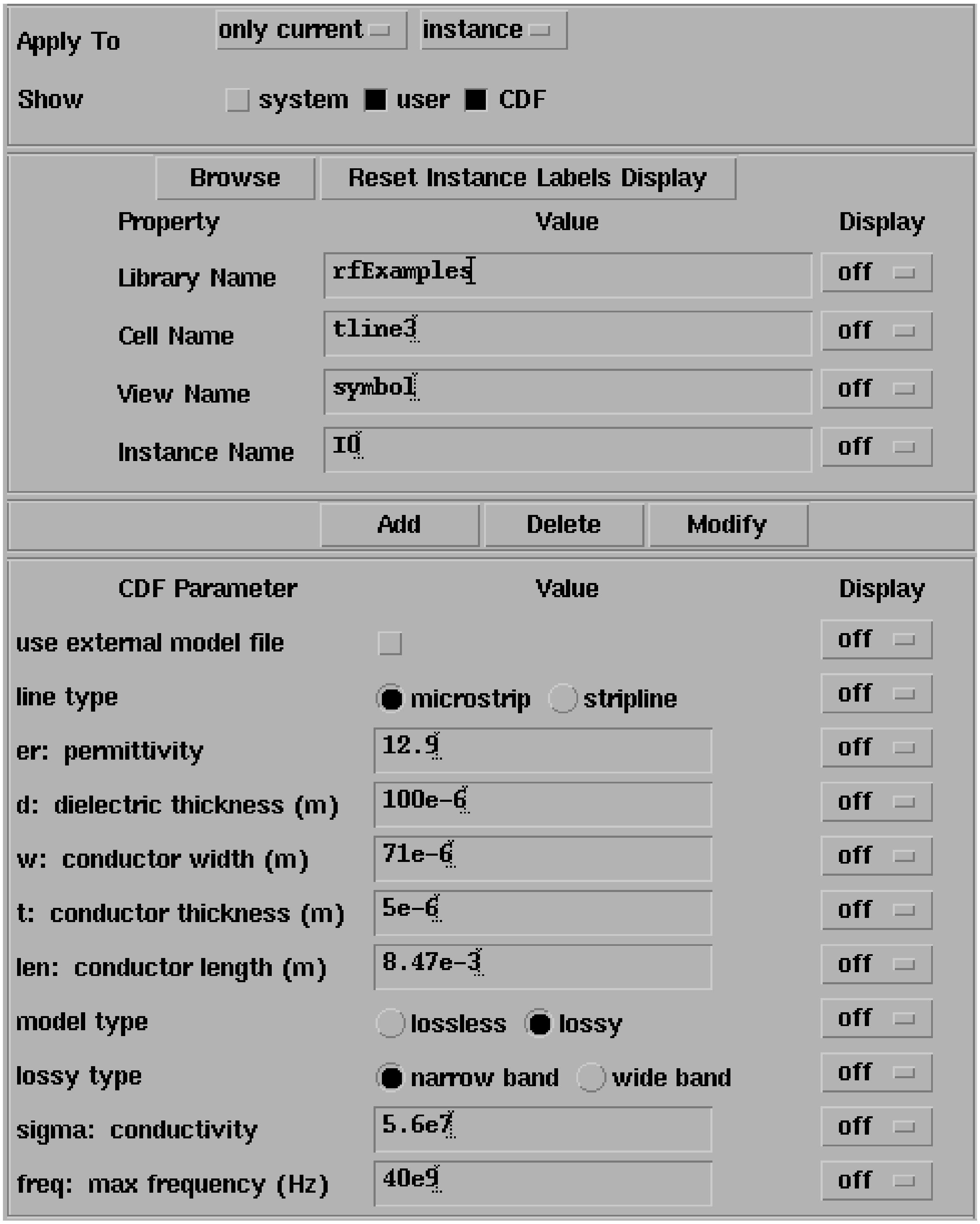
4. In the Schematic window, click on the transmission line *tline_s17_8470_40G* near the top of the schematic.

   The transmission line is now selected.



5. In the Schematic window, choose *Edit—Properties—Objects* with the transmission line selected to open the Edit Object Properties form for the transmission line component.

The *Edit – Properties – Objects* form for *tline_s17_8470_40G* appears.



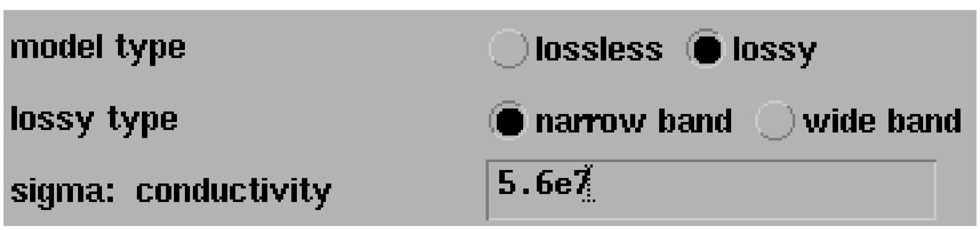
6. In the Edit Object Properties form, perform the following edits in the *CDF Parameter* section in the bottom half of the form:

- ❑ Highlight *stripline* for *line type*.
- ❑ Type 5 for *er: permittivity*.
- ❑ Type `500e-6` for *d: dielectric thickness (m)*.

Note: Notice that in the Edit Object Properties form, the values you enter are in units of meters and Hertz. These units are probably different than the units you used in LMG.

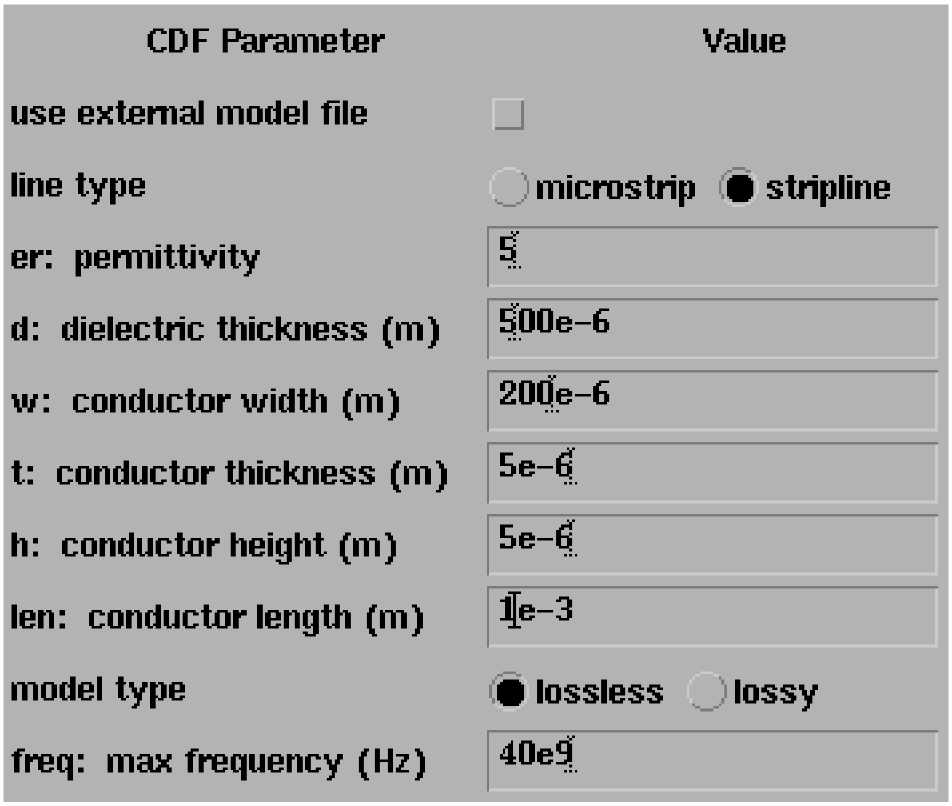
- ❑ Type `200e-6` for *w: conductor width (m)*.
- ❑ Type `5e-6` for *t: conductor thickness (m).*
- ❑ Type `5e-6` for *h: conductor height (m)*.
- ❑ Type `1e-3` for *len: conductor length (m).*
- ❑ Highlight *lossless* for the *model type*.

  Note: If you choose *lossy* for the *model type*, two additional fields open to let you specify the *lossy type* and *conductivity*.



- ❑ Type `40e9` for *freq: max frequency (Hz)*.

After edits, the CDF parameters appear as follows



CDF Parameter	Value
use external model file	☐
line type	○ microstrip ● stripline
er: permittivity	5
d: dielectric thickness (m)	500e-6
w: conductor width (m)	200e-6
t: conductor thickness (m)	5e-6
h: conductor height (m)	5e-6
len: conductor length (m)	1e-3
model type	● lossless ○ lossy
freq: max frequency (Hz)	40e9

7. In the Edit Object Properties form, click *OK*.

8. Check and Save the *tline3oscRFlmg* schematic.

You can now set up and run a simulation using the analog design environment.

Mline Transmission Line Modeling Examples

This section presents a series of examples showing how to create an *mline* transmission line model. Because the procedures for creating microstrip and stripline transmission lines are very similar, a microstrip example is shown here.

Creating an mline Transmission Line Model Starting LMG From UNIX

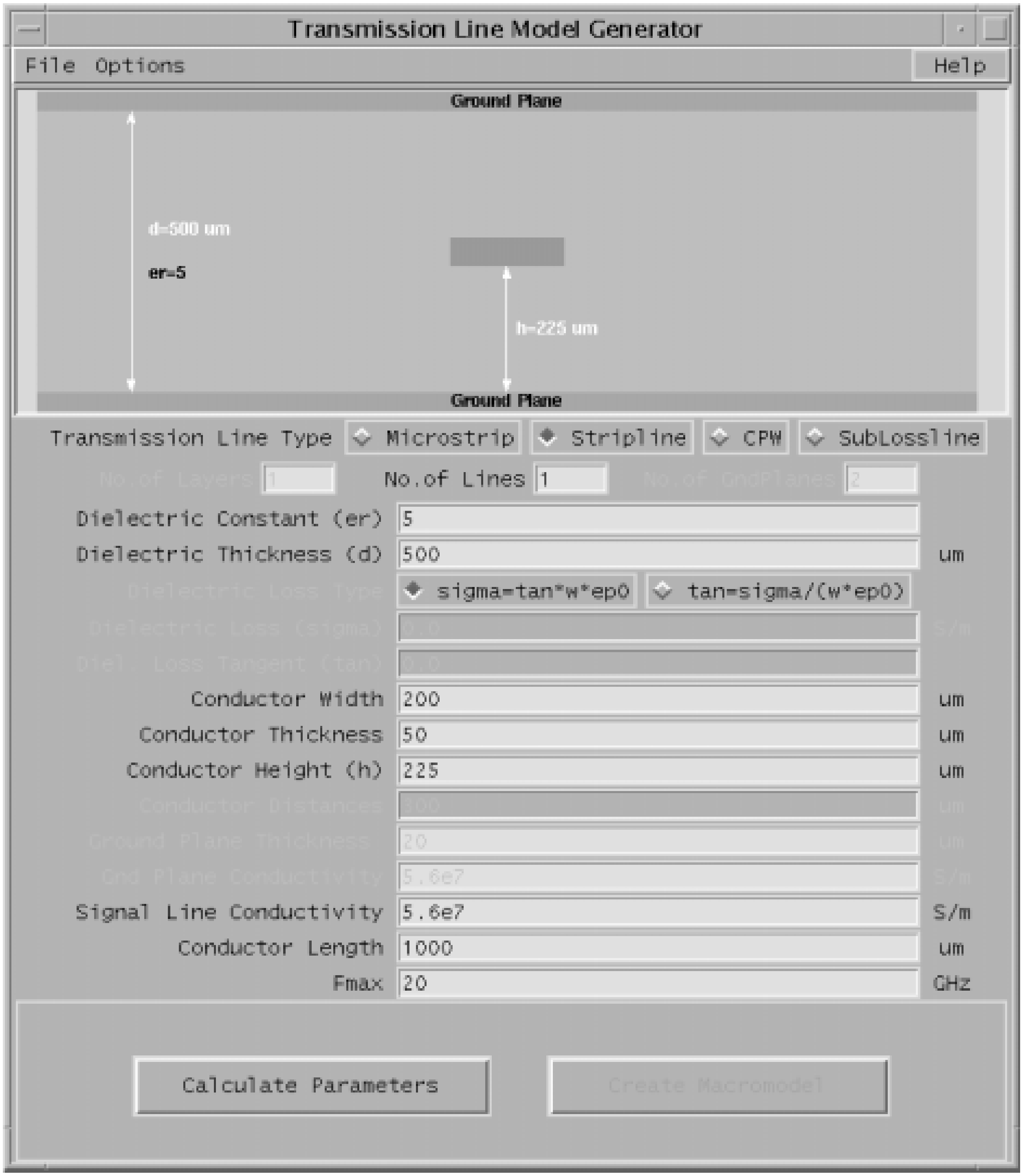
This example starts LMG from the Unix prompt. It illustrates how to create an *mline* model for 2 microstrip transmission lines using the LMG GUI.

1. At the Unix prompt, type `lmg &` and press `return` to start the LMG GUI.

```
% lmg &
```

Note: When you start LMG from the UNIX prompt, LMG does not display the Transmission Line Modeler form and thus does not display the path to the directory where the models will be saved.

The LMG GUI displays similar to the one below.



2. Choose *Options—Model Type—Lossless* to model a lossless transmission line.

3. In the LMG GUI, choose *Options—Subckt Format—Spectre*.

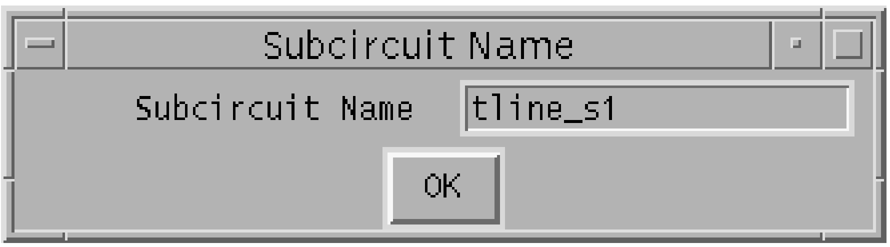
   This choice assumes that you are running your simulation in the analog design environment.

4. Choose *Options—Length Unit—um* to specify *um* as the length unit.

5. Choose *Options—Freq. Unit—GHz* to specify *GHz* as the frequency unit.

6. Choose *File—Subcircuit Name*.

   The Subcircuit Name form appears.



   Type `tline_s1` in the *Subcircuit Name* field and click *OK*.

7. Choose *File—LRCG file Name*.

   The LRCG file Name form appears.

   Type `w_line1.dat` in the *LRCG Name* field and click *OK*.

8. Specify the following values in the data entry fields:

   Transmission Line Type: `Microstrip`

   No. of Lines: `2`

   Dielectric Constant (er): `12.9`

   Dielectric Thickness (d): `200` *um*

   *sigma=tan*w*ep0*

Conductor Width: `300` *um*

Conductor Thickness: `10` *um*

Conductor Distances: `150` *um*

Signal Line Conductivity: `5.6e7` *S/m*

Conductor Length: `5000` *um*

Fmax: `30` *GHz*

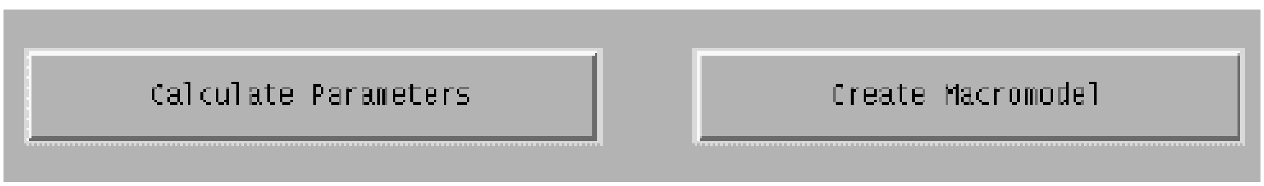
9. Choose *File—Save Current Settings* to save the current settings in the LMG GUI as default values for subsequent LMG runs.

   The current settings are saved in the file ./`.initlmg`.

   When LMG starts, it reads the ./`.initlmg` file which contains all the quantities that can be modified. As long as you preserve the ./`.initlmg` file format, you can edit this file directly to choose the LMG startup defaults.

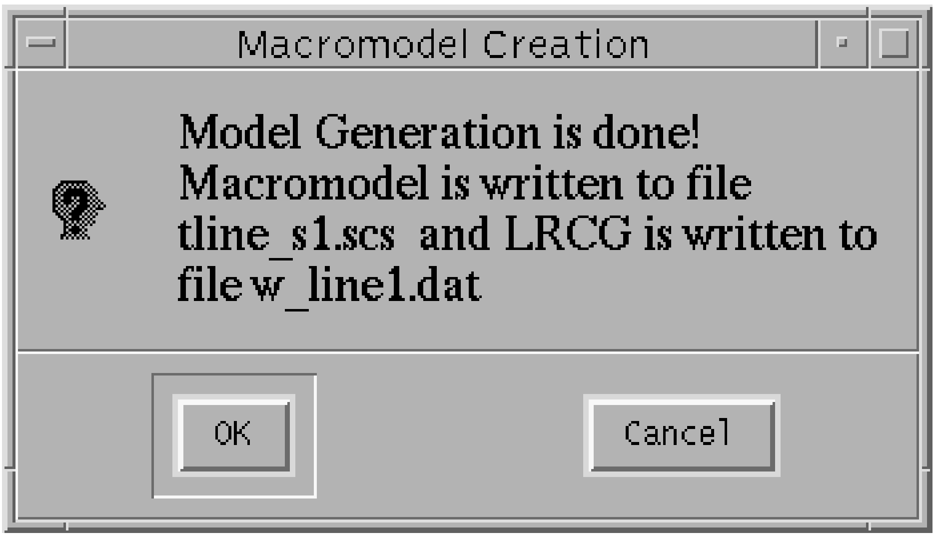
10. Click the *Calculate Parameters* button to extract the microstrip model parameters.

    The *Create Macromodel* button is enabled when the parameters are extracted.



11. Click the *Create Macromodel* button when it is enabled.

The Macromodel Creation form displays when the macromodel and data files are created.



The macromodel subckt is written to file *tline_s1.scs* and the LRCG data file is written to file *w_line1.dat*.

12. Click *OK* to create both files.

13. Choose *File—Quit* to exit LMG and close the LMG GUI.

Using an mline Macromodel in the Schematic Flow

The example presented in this section illustrates how to use an *mline* macromodel you created in the previous example from the schematic flow while you are using the analog design environment. This example uses the *mlineoscRFlmg* circuit from your editable copy of the *rfExamples* library.

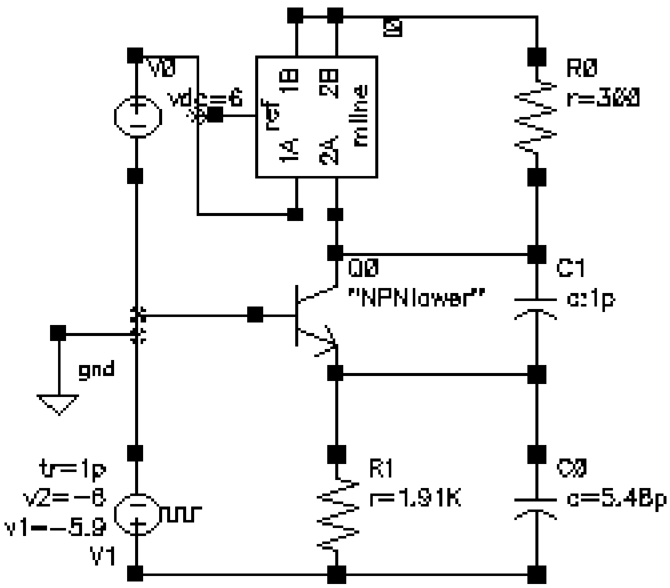
Opening the mlineoscRFlmg Circuit in the Schematic Window

1. In the *CIW*, choose *File—Open.*

The Open File form appears.

2. In the Open File form, choose the editable copy of the *rfExamples* library, *my_rfExamples* in the *Library Name* cyclic field and choose *mlineoscRFlmg* in the *Cell Names* list box and click OK.

The Schematic window for the *mlineoscRFlmg* circuit appears.



Setting Up the Model Path

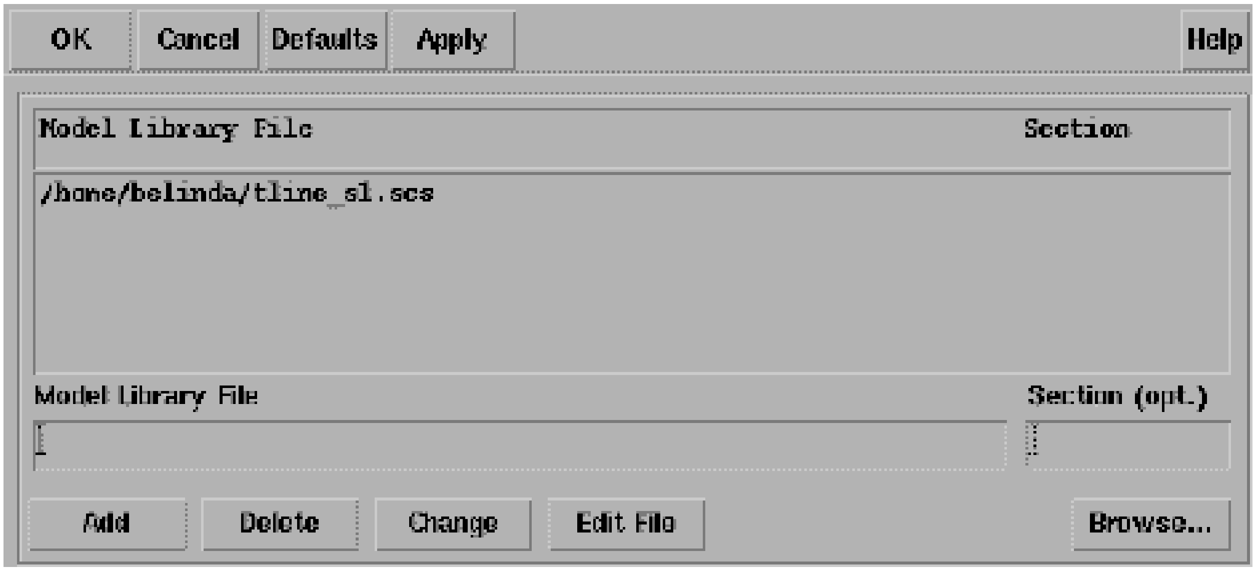
1. In the Schematic window, choose *Tools—Analog Environment* to open the Simulation window.

   The Simulation window appears.

2. In the Simulation window, choose *Setup—Model Libraries*.

   The Model Library Setup form appears.

The Model Library Setup form appears.



3. In the *Model Library File* field, type the full, absolute path to the model file plus the file name `tline_s1.scs`.

   ❑ The absolute path is the path to the directory where you started LMG in the previous example, “Creating an mline Transmission Line Model Starting LMG From UNIX” on page 520.

   ❑ The model name is the name of the *mline* model displayed in the Subcircuit Name form in the previous example.

   For example, type the following in the *Model Library File* field:

   `/home/belinda/tline_s1.scs`

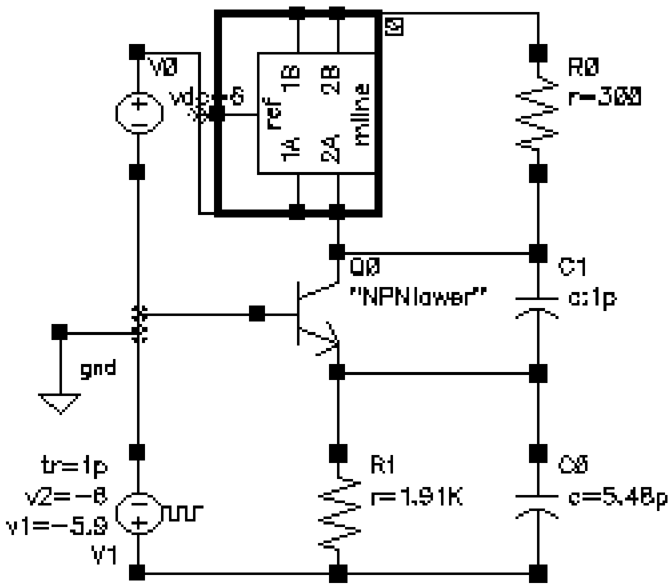
4. Click *Add*.

5. Click *OK*.

Associating the External Model File with the Transmission Line Component

1. In the Schematic window, click on the *mline* symbol near the top of the schematic.

The *mline* transmission line is now selected.



2. Choose *Edit—Properties—Objects* in the Schematic window to open the Edit Object Properties form for the selected *mline* transmission line component. The CDF parameters display at the bottom of the form.

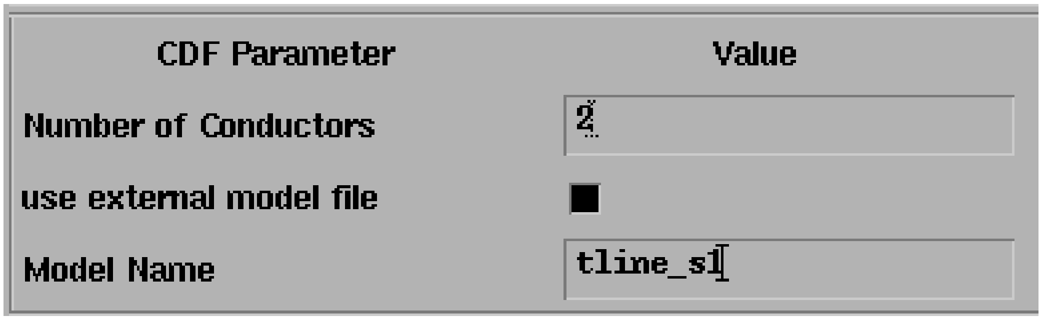
The CDF parameters for the *mline* component appear below.

CDF Parameter	Value
Number of Conductors	2
use external model file	☐
line type	◉ microstrip ○ stripline
er: permittivity	12.9
d: dielectric thickness (m)	100e-6
w: conductor widths (m)	300e-6
cd: conductor distances (m)	50e-6
t: conductor thickness (m)	5e-6
len: conductor length (m)	8.47e-3
sigma: signal conductivity	5.6e7
model type	◉ lossless ○ lossy
freq: max frequency (Hz)	40e9

3. In the CDF Parameters area, highlight *use external model file*, so the form changes to let you specify the model name.

CDF Parameter	Value
Number of Conductors	2
use external model file	■
Model Name	tline_s17_8470_40G

4. In the Model Name field, type the name of the mline model you created, `tline_s1`.



5. In the Edit Object Properties form, click *OK*.

6. Check and Save the *mlineoscRFlmg* schematic.

You can now set up and run a simulation using the new *mline* macromodel in the analog design environment.

Resolving a Possible Error Message for an mline Model File

In some circumstances the analog design environment might issue an error message stating that an `mline subckt` has six terminals, while in the schematic it is a 5-terminal device.

To solve this problem, you can open and edit the subcircuit file with the text editor of your choice.The `tline_s1.scs` file looks like the following.

```
//
// Gaussian quadrature model to transmission line.
// Transmission line type: Microstrip
// Number of signal conductors: 2
subckt tline_s1 (in0 out0 in1 out1 ref refequal)
  l0_0 (in0 n1_0) inductor l=2.655224e-11
  l1_0 (n1_0 n2_0) inductor l=1.101868e-10
  l2_0 (n2_0 n3_0) inductor l=1.856834e-10
  l3_0 (n3_0 n4_0) inductor l=2.411049e-10
  l4_0 (n4_0 n5_0) inductor l=2.704059e-10
  l5_0 (n5_0 n6_0) inductor l=2.704059e-10
  l6_0 (n6_0 n7_0) inductor l=2.411049e-10
  l7_0 (n7_0 n8_0) inductor l=1.856834e-10
  l8_0 (n8_0 n9_0) inductor l=1.101868e-10
  l9_0 (n9_0 out0) inductor l=2.655224e-11
  l0_1 (in1 n1_1) inductor l=2.655224e-11
```

```
l1_1 (n1_1 n2_1) inductor l=1.101868e-10
l2_1 (n2_1 n3_1) inductor l=1.856834e-10
l3_1 (n3_1 n4_1) inductor l=2.411049e-10
l4_1 (n4_1 n5_1) inductor l=2.704059e-10
l5_1 (n5_1 n6_1) inductor l=2.704059e-10
l6_1 (n6_1 n7_1) inductor l=2.411049e-10
l7_1 (n7_1 n8_1) inductor l=1.856834e-10
l8_1 (n8_1 n9_1) inductor l=1.101868e-10
l9_1 (n9_1 out1) inductor l=2.655224e-11
ml0_1_0 mutual_inductor coupling=2.207763e-01 ind1=l0_1 ind2=l0_0
ml0_1_1 mutual_inductor coupling=2.207763e-01 ind1=l1_1 ind2=l1_0
ml0_1_2 mutual_inductor coupling=2.207763e-01 ind1=l2_1 ind2=l2_0
ml0_1_3 mutual_inductor coupling=2.207763e-01 ind1=l3_1 ind2=l3_0
ml0_1_4 mutual_inductor coupling=2.207763e-01 ind1=l4_1 ind2=l4_0
ml0_1_5 mutual_inductor coupling=2.207763e-01 ind1=l5_1 ind2=l5_0
ml0_1_6 mutual_inductor coupling=2.207763e-01 ind1=l6_1 ind2=l6_0
ml0_1_7 mutual_inductor coupling=2.207763e-01 ind1=l7_1 ind2=l7_0
ml0_1_8 mutual_inductor coupling=2.207763e-01 ind1=l8_1 ind2=l8_0
ml0_1_9 mutual_inductor coupling=2.207763e-01 ind1=l9_1 ind2=l9_0
c0_0 (n1_0 ref) capacitor c=5.315752e-14
c1_0 (n2_0 ref) capacitor c=1.181530e-13
c2_0 (n3_0 ref) capacitor c=1.704520e-13
c3_0 (n4_0 ref) capacitor c=2.042912e-13
c4_0 (n5_0 ref) capacitor c=2.159935e-13
c5_0 (n6_0 ref) capacitor c=2.042912e-13
c6_0 (n7_0 ref) capacitor c=1.704520e-13
c7_0 (n8_0 ref) capacitor c=1.181530e-13
c8_0 (n9_0 ref) capacitor c=5.315752e-14
c0_1 (n1_1 ref) capacitor c=5.315752e-14
c1_1 (n2_1 ref) capacitor c=1.181530e-13
c2_1 (n3_1 ref) capacitor c=1.704520e-13
c3_1 (n4_1 ref) capacitor c=2.042912e-13
c4_1 (n5_1 ref) capacitor c=2.159935e-13
c5_1 (n6_1 ref) capacitor c=2.042912e-13
c6_1 (n7_1 ref) capacitor c=1.704520e-13
c7_1 (n8_1 ref) capacitor c=1.181530e-13
c8_1 (n9_1 ref) capacitor c=5.315752e-14
cm0_1_0 (n1_1 n1_0) capacitor c=5.821279e-15
cm1_1_0 (n2_1 n2_0) capacitor c=1.293893e-14
cm2_1_0 (n3_1 n3_0) capacitor c=1.866619e-14
cm3_1_0 (n4_1 n4_0) capacitor c=2.237193e-14
```

```
  cm4_1_0 (n5_1 n5_0) capacitor c=2.365344e-14
  cm5_1_0 (n6_1 n6_0) capacitor c=2.237193e-14
  cm6_1_0 (n7_1 n7_0) capacitor c=1.866619e-14
  cm7_1_0 (n8_1 n8_0) capacitor c=1.293893e-14
  cm8_1_0 (n9_1 n9_0) capacitor c=5.821279e-15
ends tline_s1
```

To edit the `tline_s1.scs` subcircuit file with the text editor, use the following steps.

1. Find `refequal` and delete it from the `tline_s1.scs` file.

```
//
// Gaussian quadrature model to transmission line.
// Transmission line type: Microstrip
// Number of signal conductors: 2
subckt tline_s1 (in0 out0 in1 out1 ref)
  l0_0 (in0 n1_0) inductor l=2.655224e-11
  l1_0 (n1_0 n2_0) inductor l=1.101868e-10
  l2_0 (n2_0 n3_0) inductor l=1.856834e-10
  l3_0 (n3_0 n4_0) inductor l=2.411049e-10
  l4_0 (n4_0 n5_0) inductor l=2.704059e-10
  l5_0 (n5_0 n6_0) inductor l=2.704059e-10
  l6_0 (n6_0 n7_0) inductor l=2.411049e-10
  l7_0 (n7_0 n8_0) inductor l=1.856834e-10
  .
  .
  .
  cm6_1_0 (n7_1 n7_0) capacitor c=1.866619e-14
  cm7_1_0 (n8_1 n8_0) capacitor c=1.293893e-14
  cm8_1_0 (n9_1 n9_0) capacitor c=5.821279e-15
ends tline_s1
```

2. Save the edited `tline_s1.scs` file.

After you edit the model file, you can perform a simulation using the `tline_s1.scs` model file without errors.

Creating an mline Transmission Line in the Schematic Flow

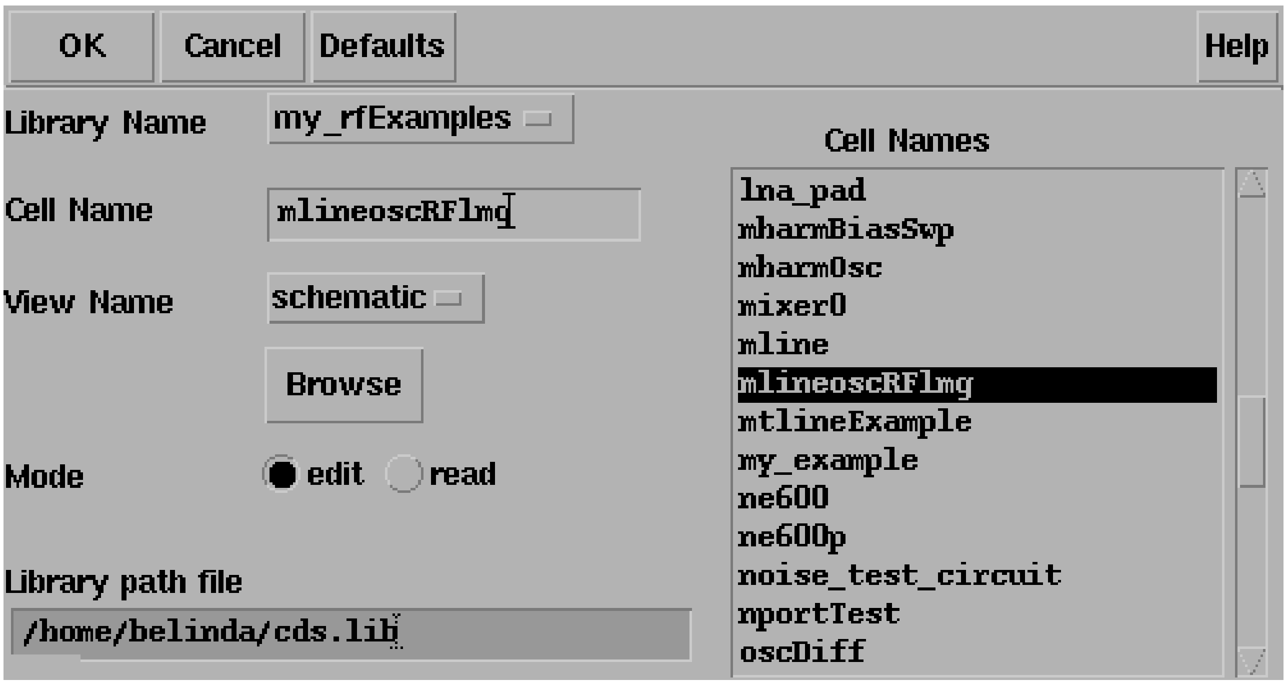
This example illustrates how to create an *mline* transmission line macromodel from the schematic flow while you are using the analog design environment. The *mline* CDF is used to invoke LMG to produce the subcircuit for the transmission line when the netlist is generated.

This example uses the circuit *mlineoscRFlmg* from your editable copy of the *rfExamples* library, *my_rfExamples*.

Opening the mlineoscRFlmg Circuit in the Schematic Window

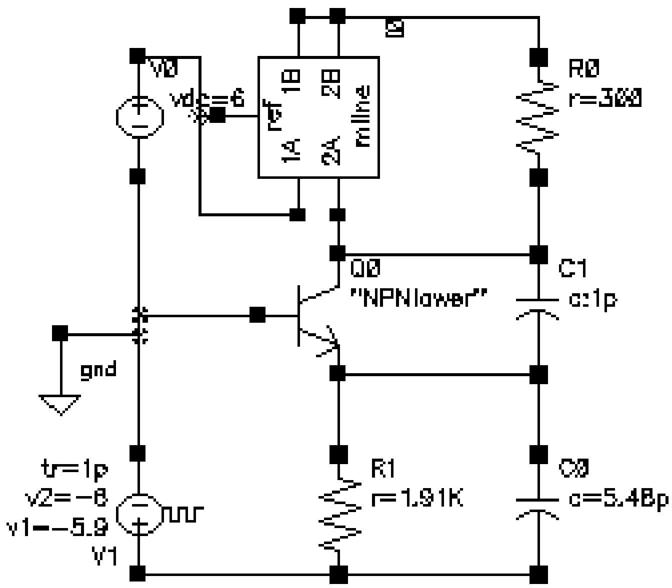
1. In the CIW, choose *File—Open*.

   The Open File form appears.

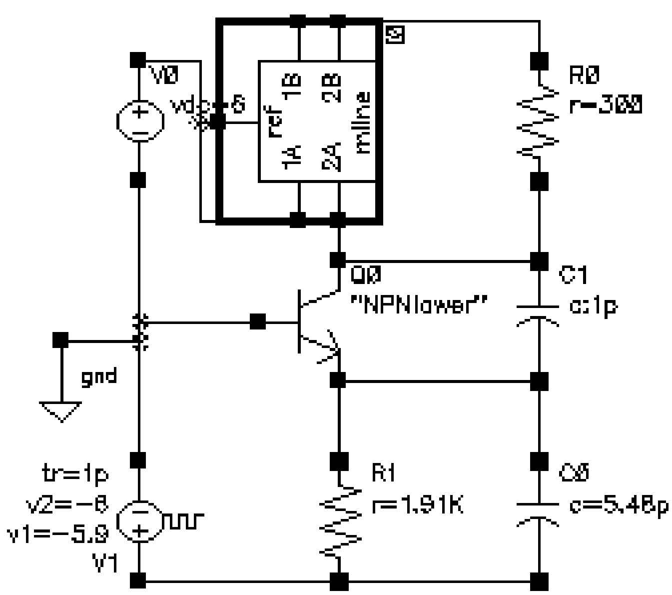


2. In the Open File form, choose the editable copy of the *rfExamples* library, *my_rfExamples,* in the *Library Name* cyclic field.
3. Choose *mlineoscRFlmg* in the *Cell Names* list box.

The Schematic window for the *mlineoscRFlmg* circuit appears.

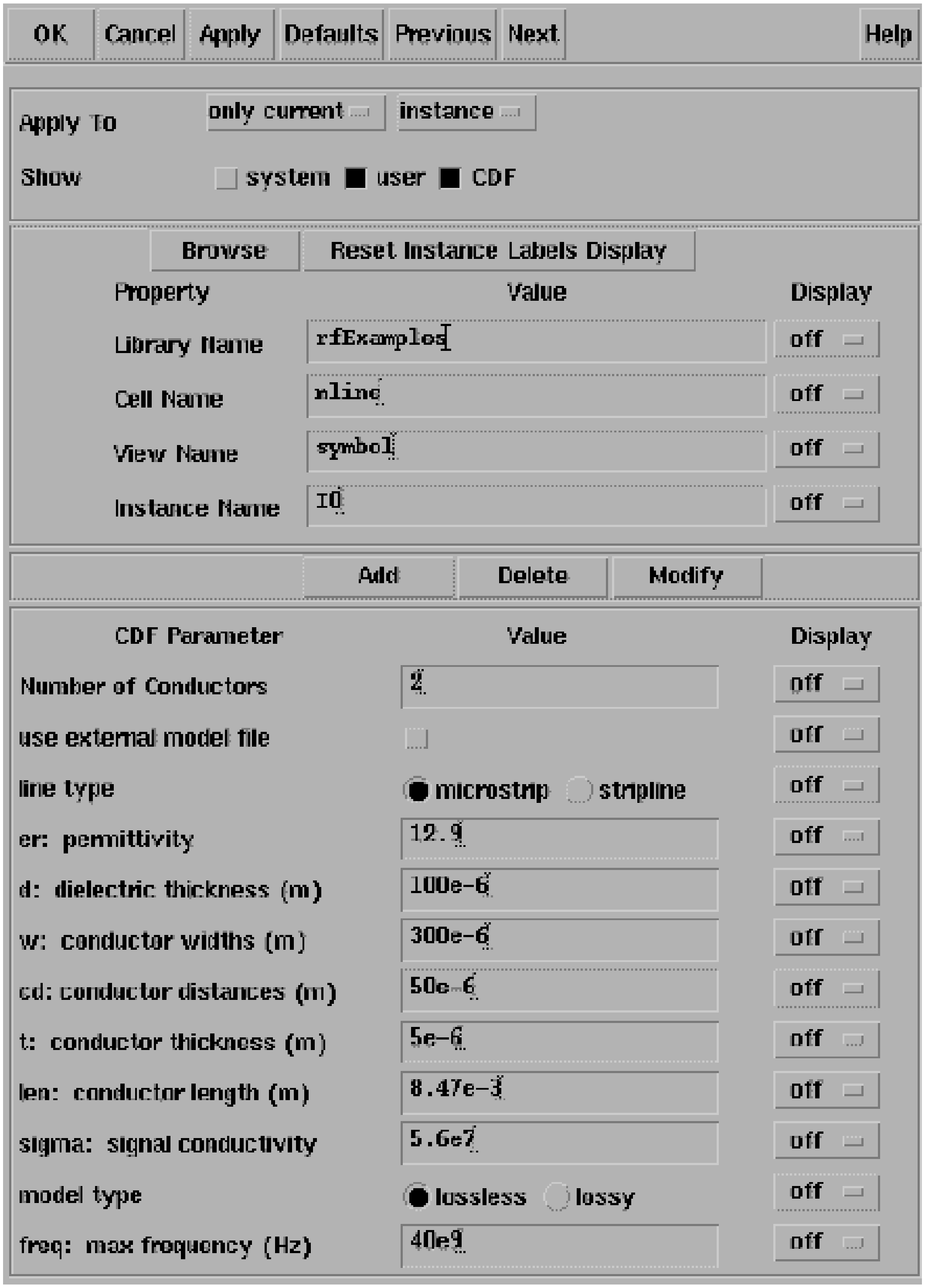


4. In the Schematic window, select the *mline* symbol near the top of the schematic.



5. Choose *Edit—Properties—Objects* in the Schematic window to open the Edit Object Properties form for the selected *mline* transmission line component.

The Edit—Properties—Objects form for *mline* component appears.

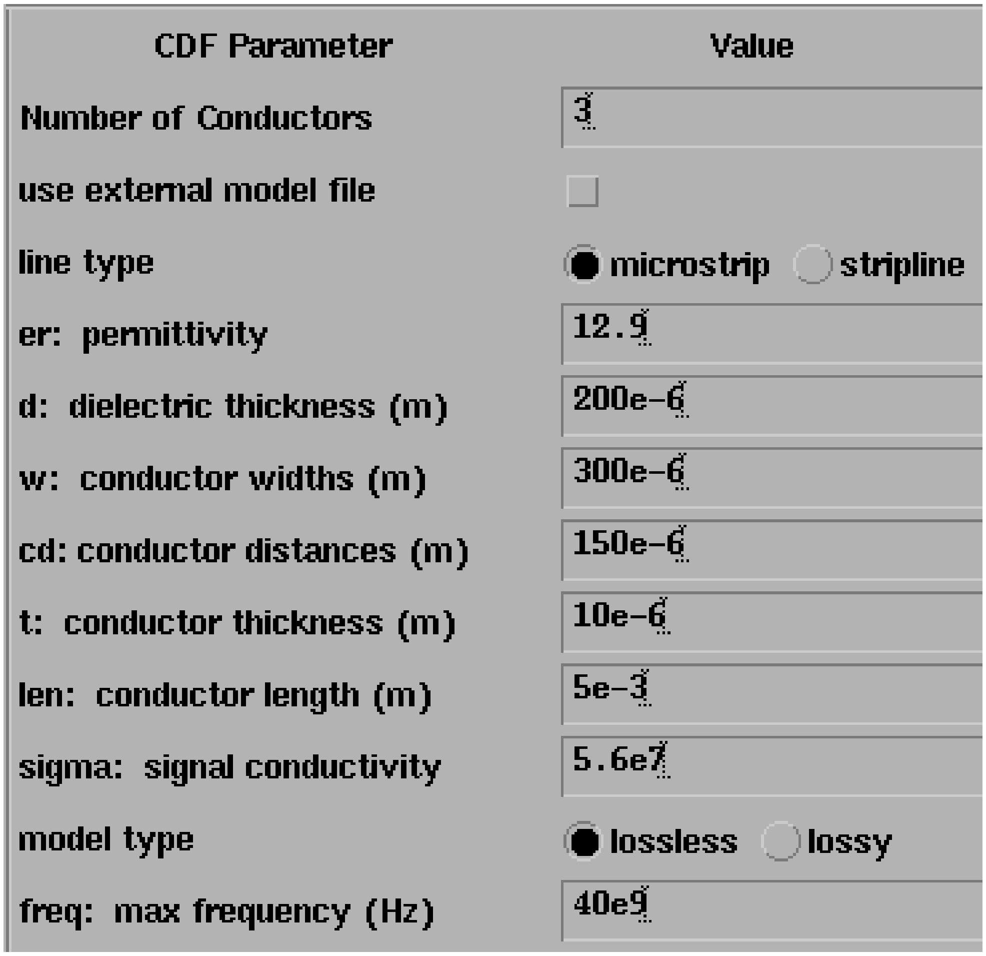


6. In the CDF Parameters section at the bottom of the Edit Object Properties form, edit the CDF parameter values as shown in Table 7-1.

Table 7-1 CDF Parameter Values for the mline Component

CDF Parameter	Value
Number of Conductors	3
use external model file	
line type	microstrip
er: permittivity	12.9
d: dielectric thickness (m)	$200e^{-6}$
w: conductor widths (m)	$300e^{-6}$
cd: conductor distances (m)	$150e^{-6}$
t: conductor thickness (m)	$10e^{-6}$
len: conductor length (m)	$5e^{-3}$
sigma: signal conductivity	$5.6e^{7}$
model type	lossless
freq: max frequency (Hz)	$40e^{9}$

The edited CDF parameters look like the following:



7. In the Edit Object Properties form, click *OK*.

8. Check and Save the *mlineoscRFlmg* schematic.

You can now set up and run a simulation using the analog design environment.

Using LMG and mtline Together in the Analog Design Environment

This example illustrates how LMG and *mtline* work together to model a transmission line in the analog design environment.

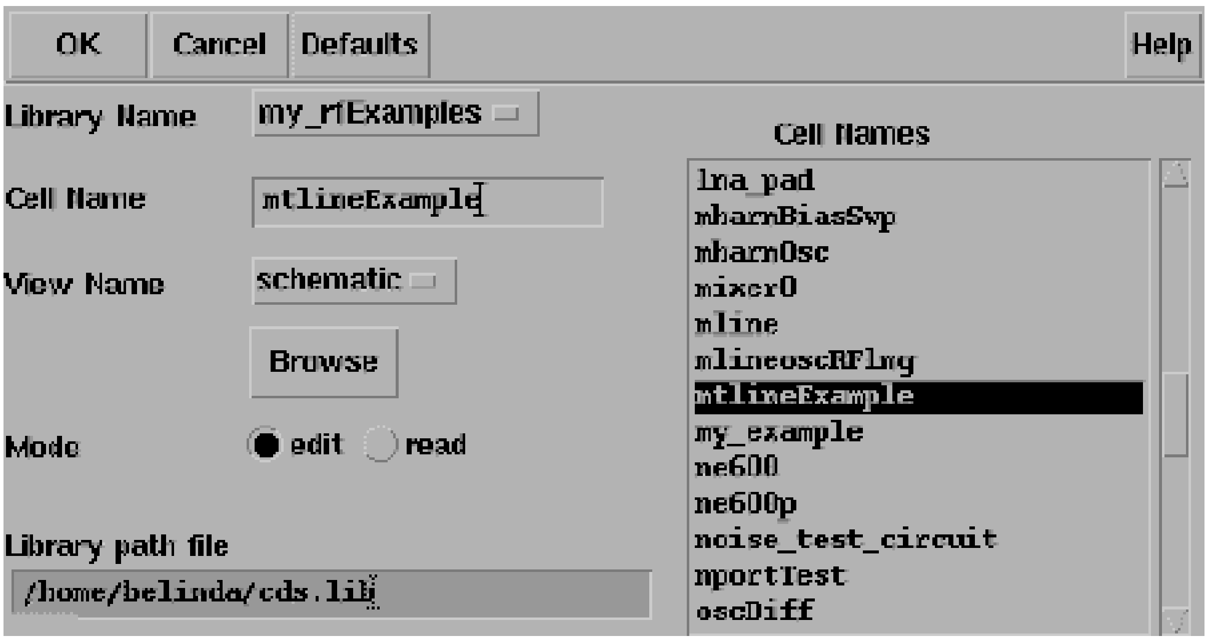
This example uses the *mtlineExample* circuit from your editable copy of the *rfExamples* library, *my_rfExamples*. This library also includes other sample circuits used in this manual.

If you need assistance setting up or accessing the *my_rfExamples* library, refer to Chapter 3, "Setting Up for the Examples."

Opening the mtlineExample Circuit in the Schematic Window

1. In the CIW, choose *File—Open*.

   The Open File form appears.



2. In the Open file form, make the following selections.

   a. In the *Library Name* cyclic field choose *my_rfExamples*, your editable copy of the *rfExamples* library.

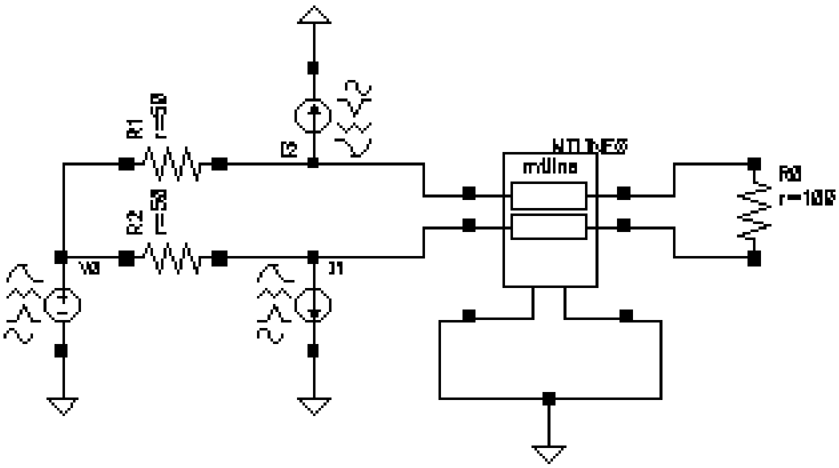
   b. In the *Cell Names* list choose *mtlineExample*.

   c. In the *View Name* cyclic field choose *schematic*.

   d. For *Mode* highlight *edit*.

3. Click *OK*.

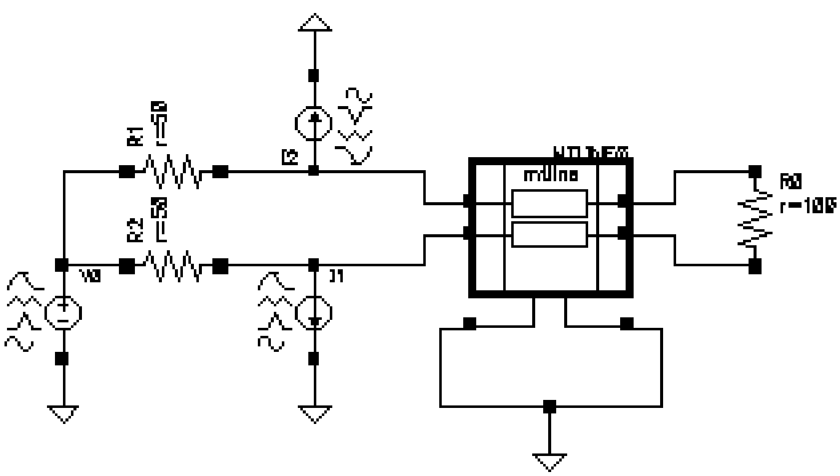
The *mtlineExample* schematic displays.



Editing CDF Parameters and Invoking LMG

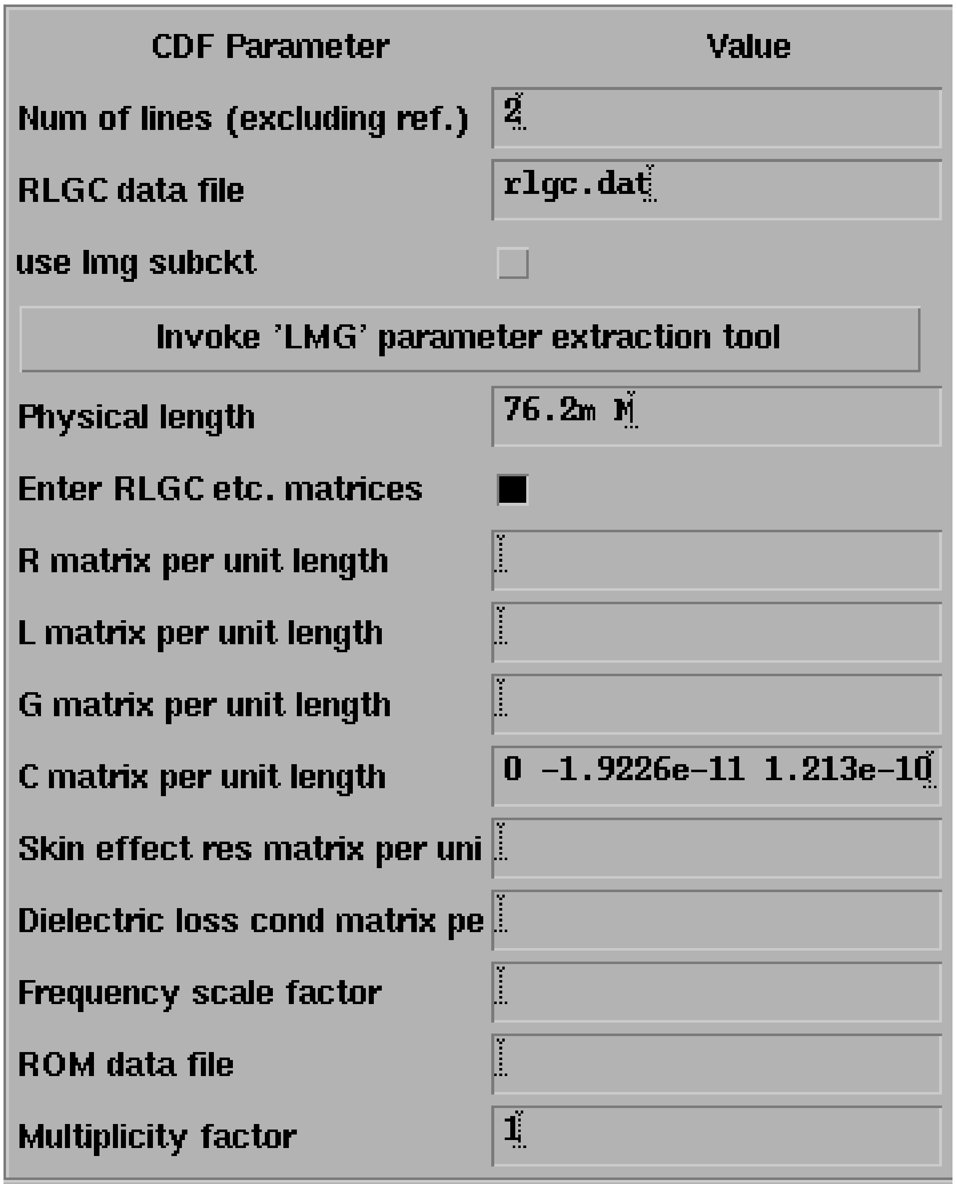
1. In the Schematic window, select the transmission line component *mtline* at the center of the schematic.

   The transmission line component is now highlighted.



2. With the transmission line selected in the Schematic window, choose *Edit – Properties – Objects* to open the Edit Object Properties form for the *mtline* component.

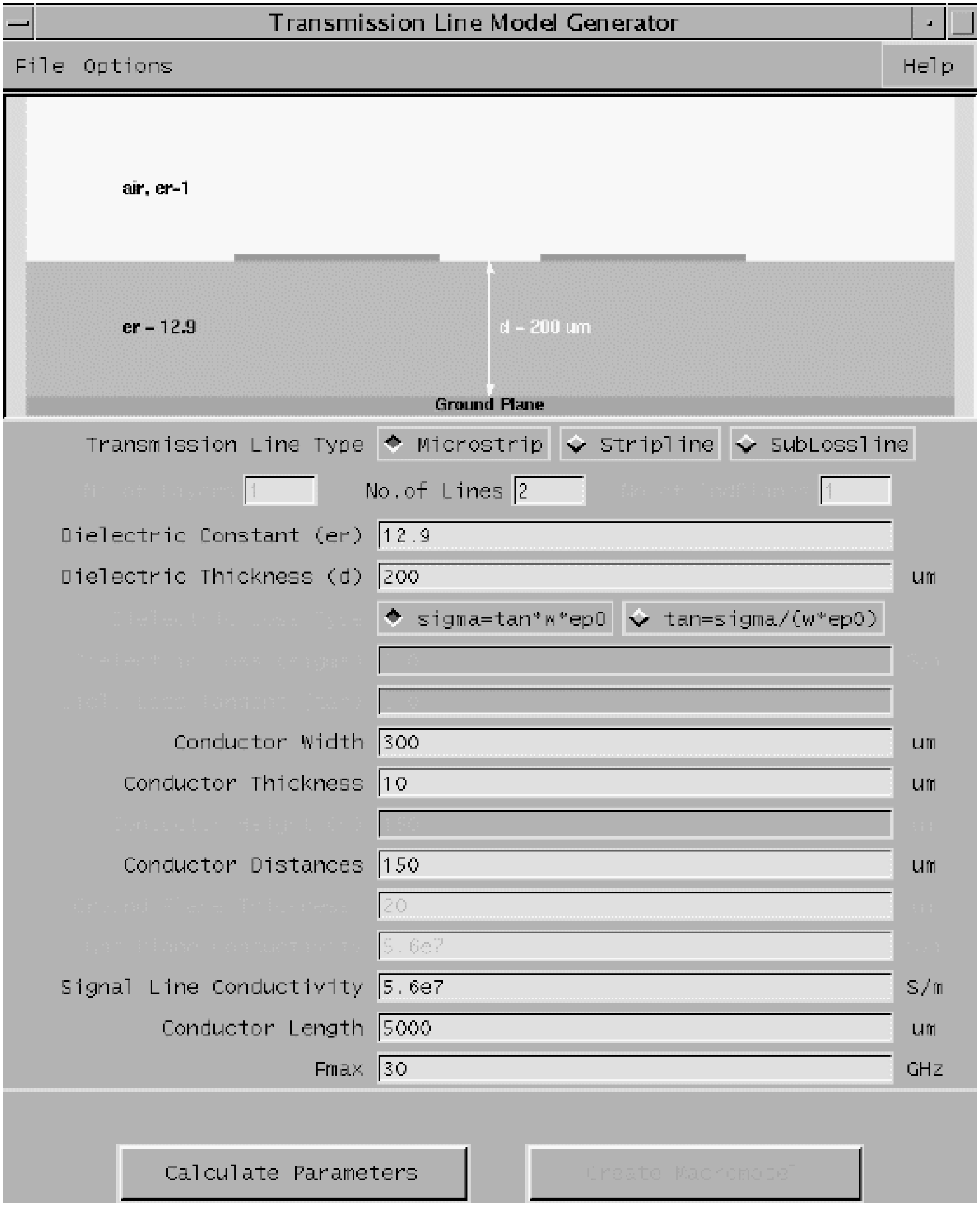
The CDF Parameter section in the bottom part of the *Edit – Properties – Objects* form for the *mtline* transmission line model appears below.



Starting LMG

1. Click *Invoke LMG parameter extraction tool* to start LMG.

The LMG GUI opens

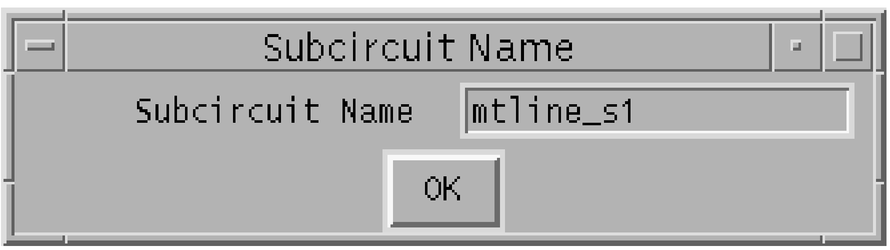


Make the following edits in the LMG GUI.

1. Choose *Options—Model Type—Lossy, Narrow Band* to model a lossy narrow band transmission line.
2. Choose *Options—Subckt Format—Spectre*.

   This choice assumes that you are running your simulation in the analog design environment.
3. Choose *Options—Length Unit—um* to specify *um* as the length unit.
4. Choose *Options—Freq. Unit—GHz* to specify *GHz* as the frequency unit.
5. Choose *File—Subcircuit Name*.

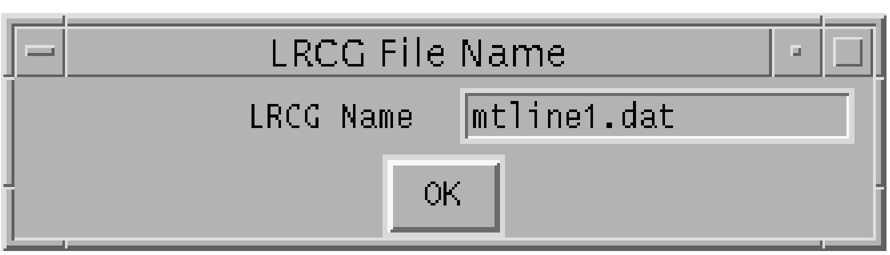
   The Subcircuit Name form appears.



6. Type `mtline_s1` in the *Subcircuit Name* field and click *OK*.

   This model name provides descriptive information about the model type of the subcircuit.
7. Choose *File—LRCG File Name*.

   The LRCG File Name form appears.



8. Type `mtline1.dat` in the *LRCG Name* field and click *OK*.

   This is the name of the file containing the LRCG matrixes in a format consistent with the *mtline* model.

9. Enter the following data in the LMG data fields:

Transmission Line Type: SubLossline

No. of Layers: `2`

No. of Lines: `2`

No. of GndPlanes: `1`

Dielectric Constant (er): `12.9 1`

Dielectric Thickness (d): `200` um

*Dielectric Loss Type: sigma = tan*w*ep0*

Dielectric Loss (sigma): `2.0 0.0` S/m

Conductor Width: `300` um

Conductor Thickness: `25` um

Conductor Height (h): `200` um

Conductor Distances: `250` um

Ground Plane Thickness: `25` um

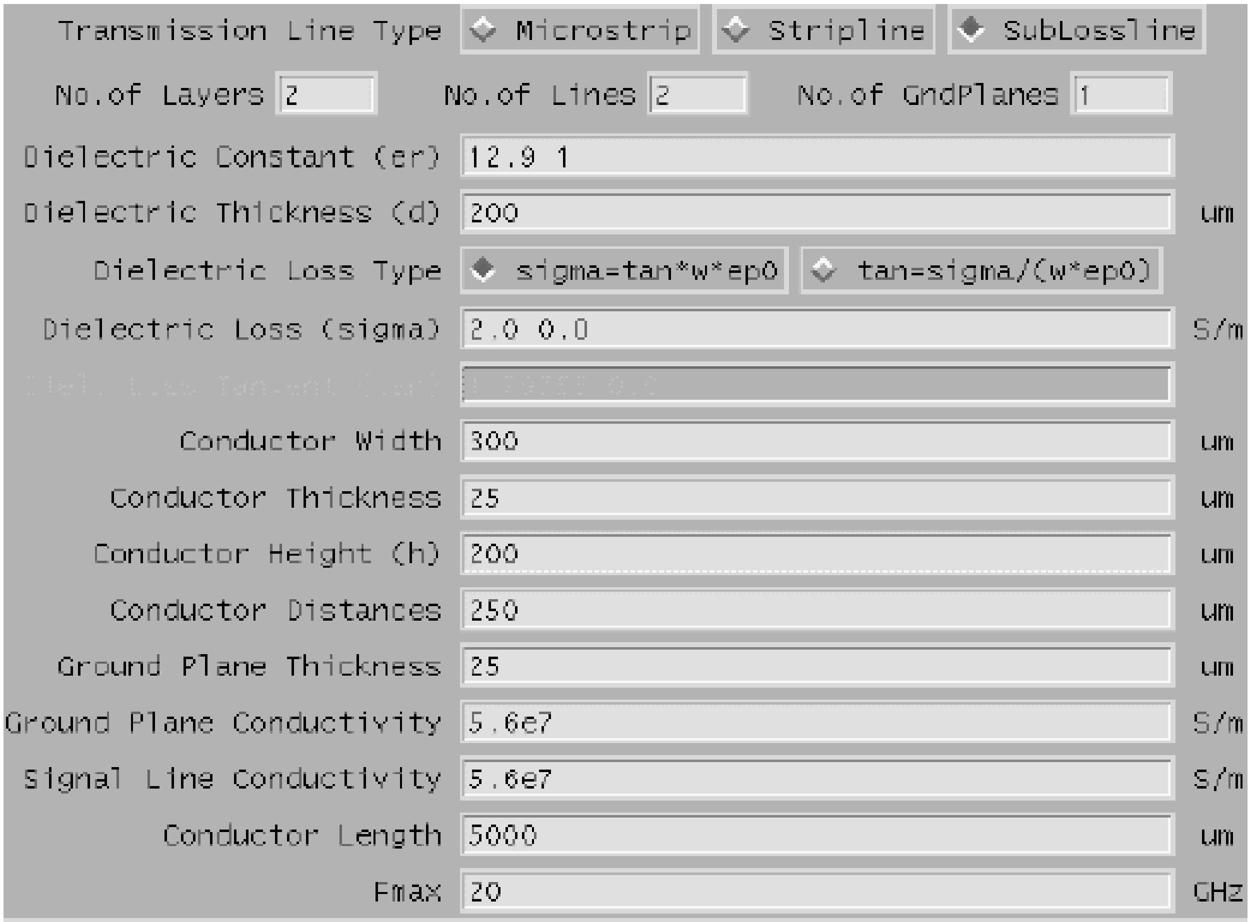
Ground Plane Conductivity: `5.6e7` S/m

Signal Line Conductivity: `5.6e7` S/m

Conductor Length: `5000` um

Fmax: `20` GHz

The LMG GUI looks like the following:

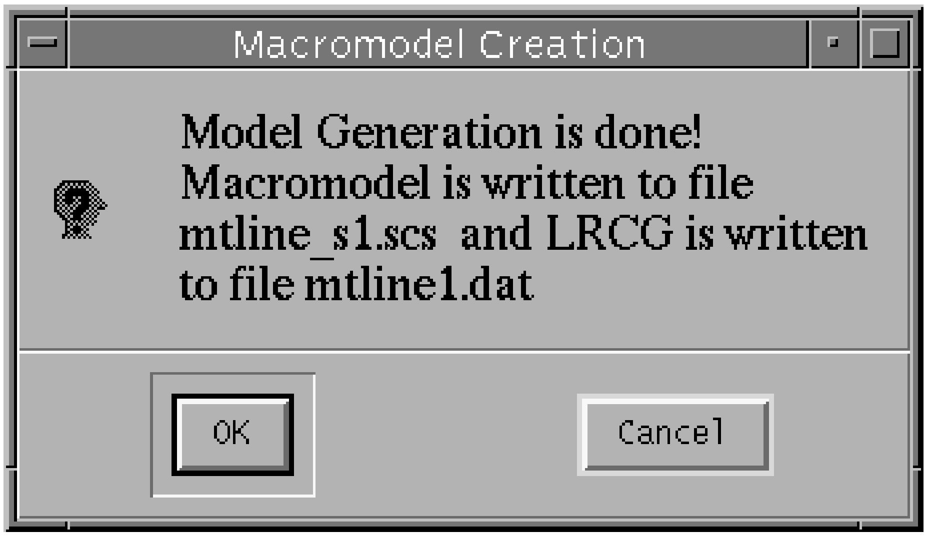


10. Click *Calculate Parameters* to extract model parameters.

When parameter extraction completes the *Create Macromodel* button becomes active.

11. Click *Create Macromodel* to generate the model and the LRCG matrix data file.

When the macromodel and LRCG matrix are generated the Macromodel Generation form displays with the names of the generated files.



12. Click *OK* to create both files.

- The macromodel subckt is written to file *mtline_s1.scs*
- The LRCG data file is written to file *mtline1.dat*

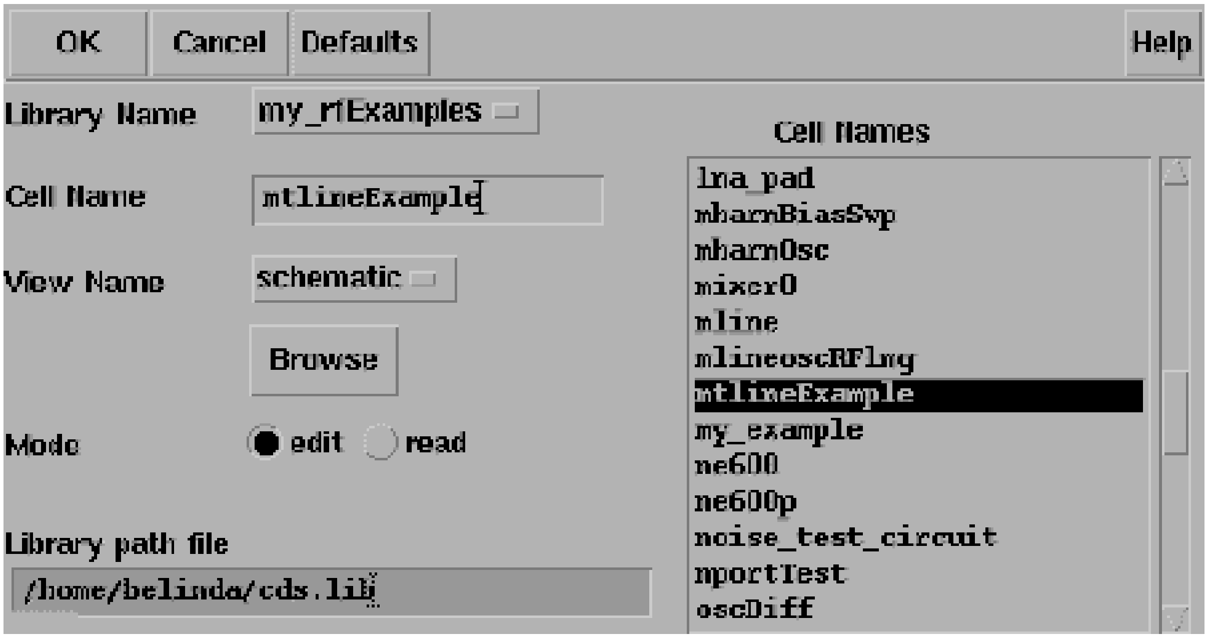
13. Choose *File — Quit* to exit LMG and close the LMG GUI.

Using the LRCG Data File in the Component CDF Form

Open the *mtlineExample* circuit and the *Edit – Properties – Objects* form for the *mtline* component.

1. In the CIW, choose *File—Open*.

The Open File form appears.



2. In the Open file form, make the following selections.

   a. In the *Library Name* cyclic field choose *my_rfExamples*, your editable copy of the *rfExamples* library.

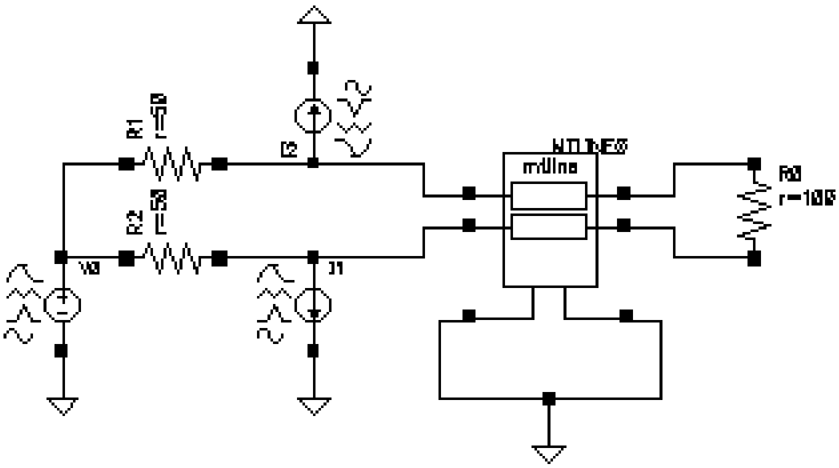
   b. In the *Cell Names* list choose *mtlineExample*.

   c. In the *View Name* cyclic field choose *schematic*.

   d. For *Mode* highlight *edit*.

3. Click *OK*.

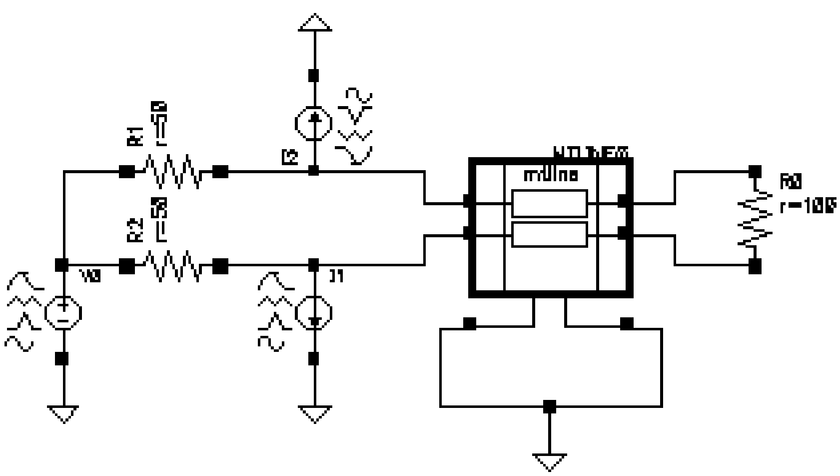
The *mtlineExample* schematic displays.



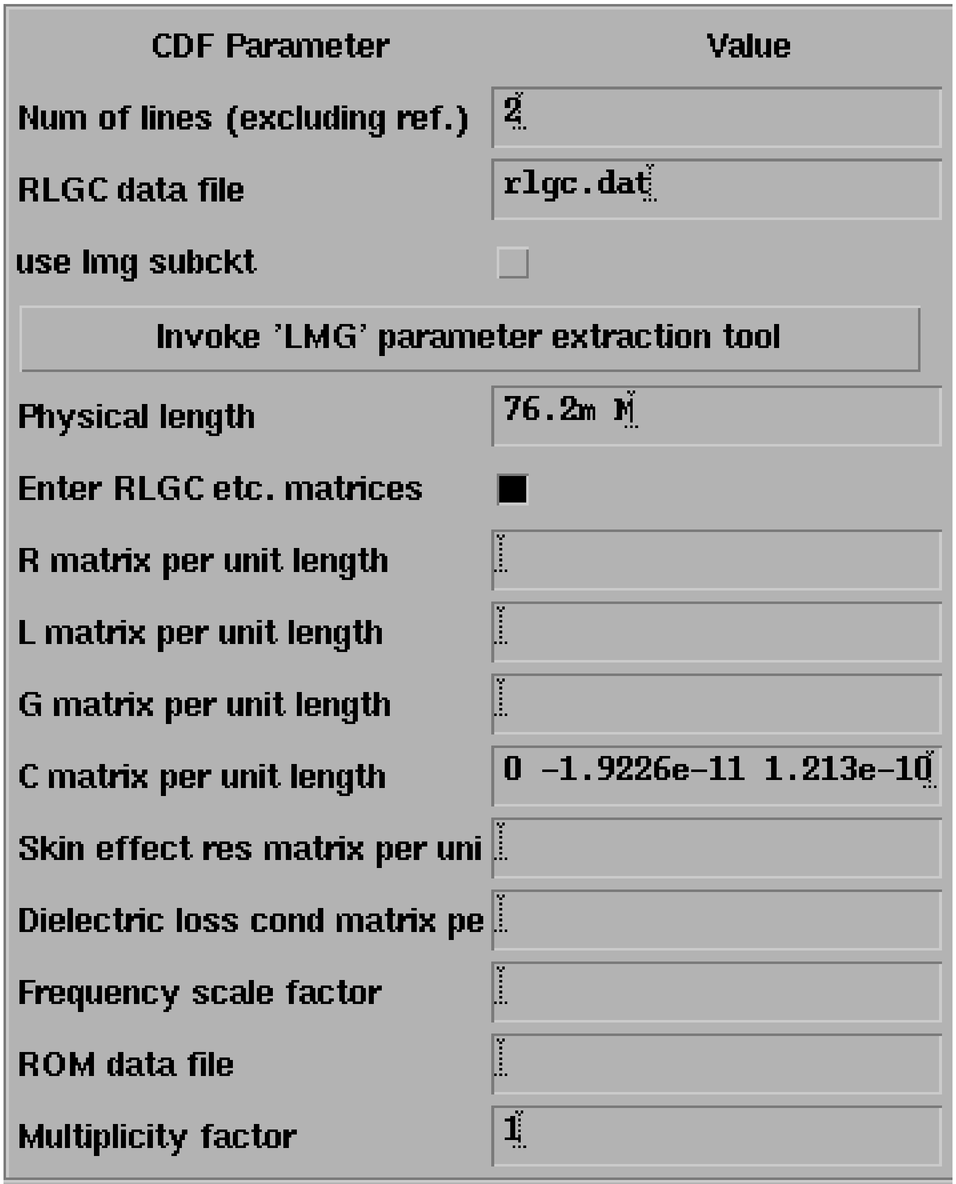
Editing CDF Parameters

1. In the Schematic window, select the transmission line component *mtline* at the center of the schematic.

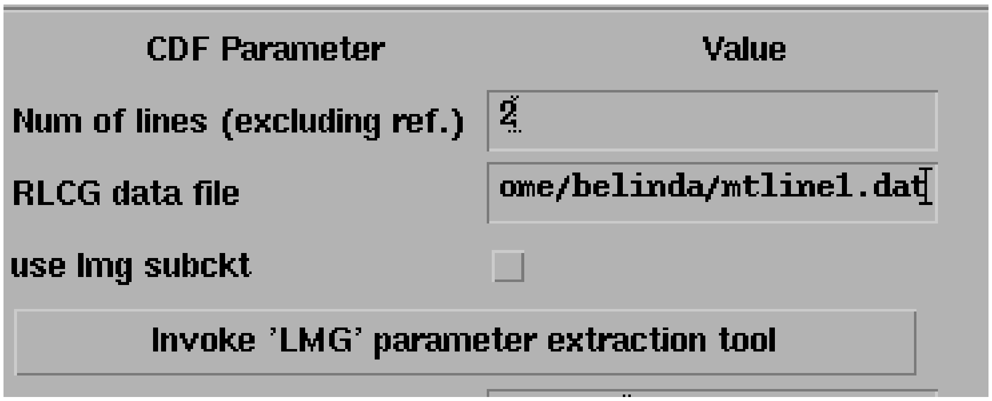
   The transmission line component is now highlighted.



2. With the transmission line selected in the Schematic window, choose *Edit – Properties – Objects* to open the Edit Object Properties form for the *mtline* component.



3. Edit the *CDF Parameter* data field value for *LRCG data file* by deleting `lrcg.dat` and replacing it with the full, absolute path to the data file you generated with LMG in the previous example, For example, `/home/belinda/mtline1.dat`

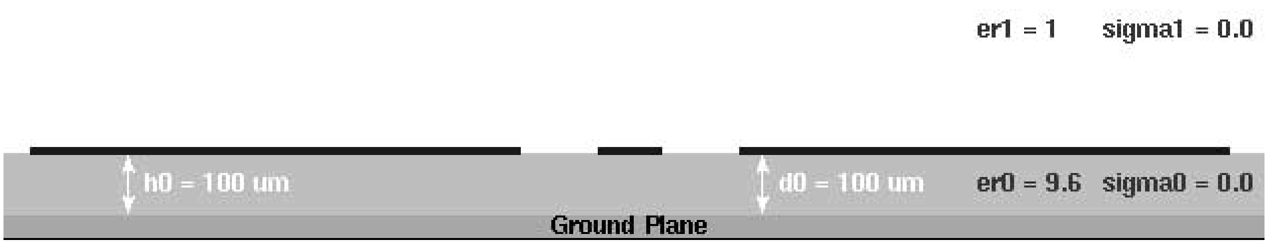


4. In the *Edit – Properties – Objects* form, click *OK*.

   When you click *OK*, the form closes and the `mtline1.dat` file is saved in the schematic editor. When you simulate with the *mtline* component, the `mtline1.dat` LRCG data file is used.

Coplanar Waveguide Modeling and Analysis

A single-signal coplanar waveguide is a transmission line made up of three conductors separated by gaps and lying on the surface of a dielectric substrate.



From left to right, the first and third conductors are ground planes which are 8 to12 times wider than the middle conductor, which is the signal line. When the signal line in the middle is too narrow, the first and third conductors are 5 to 7 times the dielectric thickness of the narrow signal line in the center.

The characteristic impedance (Z_c) of the CPW is a function of the ratio of the width of the gap between the signal line and the ground planes to the thickness of the dielectric substrate and its dielectric constant.

The first example, Using LMG to Obtain Subcircuit Macromodel and LRCG Files, creates a CPW lossy, narrow-band model with one signal line. The signal line is100 um wide with130 um gaps on either side between the signal line and the 800 um wide ground planes. The characteristic impedance of the CPW is Z_c=50 Ohm.

Using LMG to Obtain Subcircuit Macromodel and LRCG Files

When you use LMG to model and analyze coplanar waveguide (CPW) transmission lines, you begin by verifying the content of the `.initlmg` file. You then open the LMG GUI, select the CPW transmission line type, and LMG extracts parameters and generates a CPW model.

Verifying and Editing the Content of the `initlmg` File

Many features of the CPW model are displayed in both the `.initlmg` file illustrated in Example 7-1 and in the LMG GUI displayed in Figure 7-1 on page 552.

1. In the directory where you plan to invoke LMG, use the text editor of your choice to create the .initlmg file shown in Example 7-1.

Example 7-1 Sample initlmg file for a Coplanar Waveguide

```
lmgLineType = CPW
lmgModelType = LossyNarrow
lmgSubcktFormat = 0
lmgNumLayers = 2
lmgNumLines = 3
lmgNumGndPlanes = 1
lmgLengthUnit = um
lmgFreqUnit = GHz
lmgDielectricPermittivity = 9.6 1
lmgDielectricThickness = 100 400
lmgLossSigmaLossTang = VSIGMA
lmgDielectricConductivity = 0.0
lmgGndThickness = 20
lmgGndSigma = 5.6e7
lmgConductorWidth = 800 100 800
lmgConductorGaps = 130
lmgConductorThickness = 10
lmgConductorHeight = 100
lmgConductorLength = 2000
_lmgConductivity = 5.6e7
lmgMaxFreq = 1
lmgSubCircuitName = cpw1
lmgLRCGName = w_linecpw1.dat
lmgOutputID = BOTH
```

2. Be sure to save the edited `.initlmg` file in the directory where you plan to invoke LMG.

   The edited `.initlmg` file defines the properties of the CPW. Once you open the LMG GUI, you can modify or define the properties of the CPW by entering values in the GUI. When you exit LMG, you can save the values in the LMG GUI to create a new `.initlmg` file.

Opening the LMG GUI

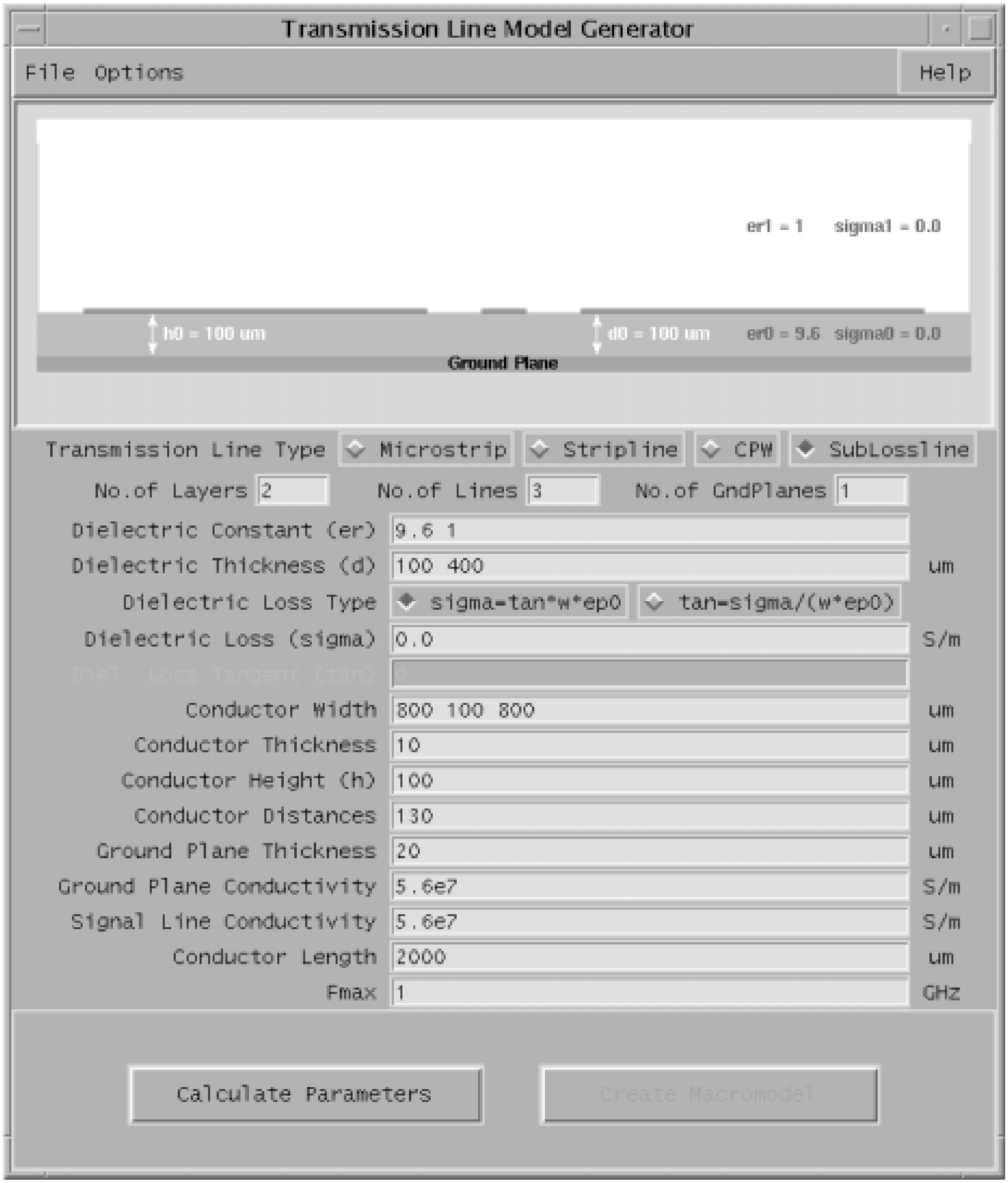
1. From the directory where you created and saved the `.initlmg` file, invoke the LMG GUI from the Unix command line by typing:

   ```
   % lmg &
   ```

   Note: When you start LMG from the UNIX prompt, LMG saves the model files in the directory where you start it. The LMG GUI displays directly without first displaying the Transmission Line Modeler form which shows the path to the model directory (the directory where LMG saves model files).

   The LMG GUI opens reflecting the CPW parameter values from the `.initlmg` file.

Figure 7-1 LMG UI set up to Model a Coplanar Waveguide



Notice the following *File* menu choices made in the `.initlmg` file.

`initlmg` File Entry	**File Menu Choice**
`lmgSubCircuitName` = `cpw1`	File—Subcircuit Name
`lmgLRCGName` = w_`linecpw1.dat`	File—LRCG file Name

Notice the following *Options* menu choices made in the `.initlmg` file.

`initlmg` File Entry	**Options Menu Choice**
`lmgModelType` = `LossyNarrow`	Options—Model Type—Lossy, Narrow Band
`lmgSubcktFormat` = `0`	Options—Subckt Format—Spectre
`lmgLengthUnit` = `LossyNarrow`	Options—Length Unit—um
`lmgFreqUnit` = `LossyNarrow`	Options—Freq. Unit—GHz

Notice the following GUI field choices made in the `.initlmg` file and displayed in the LMG GUI.

`initlmg` File Entry	**GUI Choice**
`lmgLineType` = `CPW`	Transmission Line Type is CPW.
`lmgNumLines` = `3`	Number of Lines is 3. The 2 wide ground planes and the 1 narrow signal line.
`lmgDielectricPermitivity` = `9.6 1`	Dielectric Constants are 9.6 and 1. The values are separated by a space.
`lmgConductorWidth` = `800 100 800`	Conductor Widths are 800, 100 and 800 um. The values are separated by a space.
`lmgConductorGaps` = `130`	Conductor Distances are both 130 um.
`lmgConductorLength` = `2000`	Conductor length is 2000 um.

Whenever you enter multiple values in the LMG GUI for a parameter that accepts multiple values, separate the values with a space.

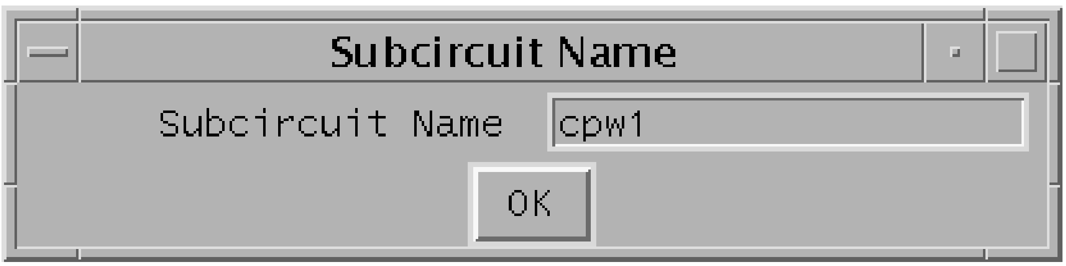
The Conductor Width values have a ratio of 8 to 1. Usually the larger the ratio between the two numbers, the more accurate the result but the slower the speed.

Since both Conductor Distances are the same, only one entry is required.

Generating the Subcircuit and LRCG Files

1. To enter a name for the subcircuit file, choose *File—Subcircuit Name*.

The Subcircuit Name form appears.



a. The subcircuit name, `CPW1`, from the `initlmg` file appears in the *Subcircuit Name* field.

b. Click *OK*.

This name gives information about the type of model subcircuit created. The suffix `scs` is attached to the subcircuit model file.

2. To enter a name for the LRCG file, choose *File—LRCG file Name*.

The LRCG File Name form appears.

a. The LRCG file name, `w_linecpw1.dat` from the `initlmg` file appears in the *LRCG Name* field.

b. Click *OK*.

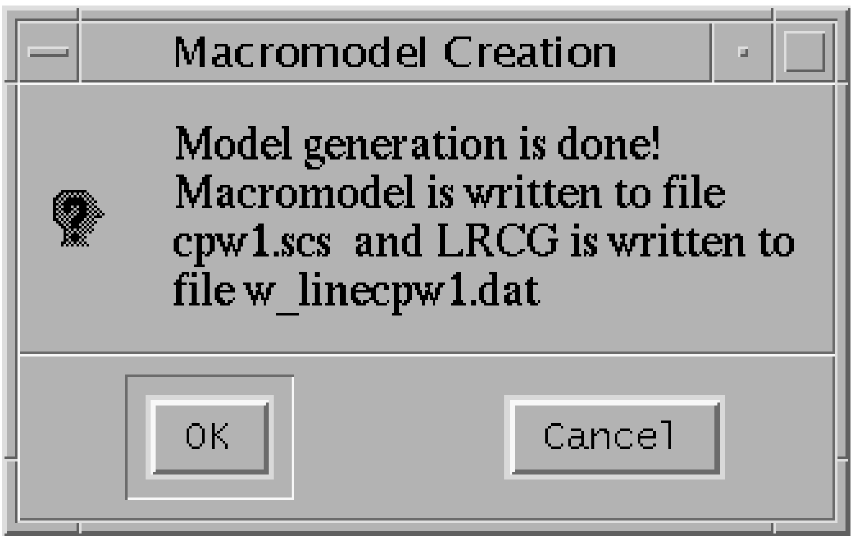
This name gives information about the type of model this LRCG file is associated with.

3. Click *Calculate Parameters* at the bottom of the LMG GUI.

The *Create Macromodel* button is active after the parameters are calculated.

4. Click *Create Macromodel* at the bottom of the LMG GUI to create the CPW macromodel netlist file and the LRCG data file.

   The Macromodel Creation form displays the names of the two final result files.



   The two result files are created in the directory where you started LMG:

   - An LRCG file with the name `w_linecpw1.dat`. (See Example 7-2 on page 555).
   - A macromodel netlist file with the name `cpw1.scs`. (See Example 7-3 on page 556).

   Note: Be sure to record the absolute path to the subckt model file `cpw1.scs` and the LRCG file `w_linecpw1.dat`. You will need the absolute path and filenames when you use the files in CPW simulations later in this section.

5. Click *OK* in the Macromodel Creation form.

Example 7-2 The Contents of the LRCG (w_linecpw1.dat) File

```
;  RLCG Matrices produced by LMG
;  It's Coplanar waveguide (CPW) line, first and last lines are regarded as ground
;  Size Reduction is done automatically, reference FAQs in the Documents for details
;  The Inputs are:
;  lmgLineType = CPW
;  lmgModelType = LossyNarrow
;  lmgSubcktFormat = 0
;  lmgNumLayers = 2
;  lmgNumLines = 3
;  lmgNumGndPlanes = 1
;  lmgLengthUnit = um
;  lmgFreqUnit = GHz
;  lmgDielectricPermittivity = 9.6 1
```

```
;  lmgDielectricThickness = 100 400
;  lmgLossSigmaLossTang = VSIGMA
;  lmgDielectricConductivity = 0.0
;  lmgGndThickness = 20
;  lmgGndSigma = 5.6e7
;  lmgConductorWidth = 800 100 800
;  lmgConductorGaps = 130
;  lmgConductorThickness = 10
;  lmgConductorHeight = 100
;  lmgConductorLength = 2000
;  _lmgConductivity = 5.6e7
;  lmgMaxFreq = 1
;  lmgSubCircuitName = cpw1
;  lmgLRCGName = w_linecpw1.dat
;  lmgOutputID = BOTH

FORMAT Freq:   L1:1
               R1:1
               C1:1
               G1:1

1.000000e+09 :    3.967294e-07
                  6.935151e+01
                  1.574700e-10
                  0.000000e+00
```

6. Choose *File—Quit* to exit LMG. Click *OK* in the Quit form.

You will use the LRCG file `w_linecpw1.dat` and the subckt model file `cpw1.scs` you just created in different simulations later in this section.

Resolving a Possible Error Message in the mtline Model File

In some circumstances the analog design environment might issue an error message for the `cwp1.scs` subcircuit file.

To solve this problem, you can open and edit the subcircuit file with the text editor of your choice.The `cwp1.scs` file looks like the following.

Example 7-3 The Original Contents of the cpw1.scs File

```
//
// quadrature model to transmission line.
// Transmission line type: CPW
// Number of signal conductors: 1
subckt cpw1 (in0 out0 ref)
  l0_0 (in0 n1_0) inductor l=3.722123e-11 r=6.506573e-03
  l1_0 (n1_0 n2_0) inductor l=1.458814e-10 r=2.550124e-02
  l2_0 (n2_0 n3_0) inductor l=2.136269e-10 r=3.734371e-02
  l3_0 (n3_0 n4_0) inductor l=2.136269e-10 r=3.734371e-02
  l4_0 (n4_0 n5_0) inductor l=1.458814e-10 r=2.550124e-02
  l5_0 (n5_0 out0) inductor l=3.722123e-11 r=6.506573e-03
  c0_0 (n1_0 ref) capacitor c=3.730874e-14
  c1_0 (n2_0 ref) capacitor c=7.536955e-14
```

```
  c2_0 (n3_0 ref) capacitor c=8.958279e-14
  c3_0 (n4_0 ref) capacitor c=7.536955e-14
  c4_0 (n5_0 ref) capacitor c=3.730874e-14
ends cpw1
```

To edit the `cpw1.scs` subcircuit file with the text editor, use the following steps.

1. Find the following text in the `cpw1.scs` file:

   ```
   (in0 out0 ref)
   ```

   Replace it with the following text.

   ```
   (in0 out0 gnd refgnd)
   ```

2. The edited file looks like Example 7-4.

Example 7-4 The Corrected Contents of the cpw1.scs File

```
//
// quadrature model to transmission line.
// Transmission line type: CPW
// Number of signal conductors: 1
subckt cpw1 (in0 out0 gnd refgnd)
  l0_0 (in0 n1_0) inductor l=3.722123e-11 r=6.506573e-03
  l1_0 (n1_0 n2_0) inductor l=1.458814e-10 r=2.550124e-02
  l2_0 (n2_0 n3_0) inductor l=2.136269e-10 r=3.734371e-02
  l3_0 (n3_0 n4_0) inductor l=2.136269e-10 r=3.734371e-02
  l4_0 (n4_0 n5_0) inductor l=1.458814e-10 r=2.550124e-02
  l5_0 (n5_0 out0) inductor l=3.722123e-11 r=6.506573e-03
  c0_0 (n1_0 ref) capacitor c=3.730874e-14
  c1_0 (n2_0 ref) capacitor c=7.536955e-14
  c2_0 (n3_0 ref) capacitor c=8.958279e-14
  c3_0 (n4_0 ref) capacitor c=7.536955e-14
  c4_0 (n5_0 ref) capacitor c=3.730874e-14
ends cpw1
```

3. Save the edited `cpw1.scs` file.

After you edit and save the `cpw1.scs` file, you can perform a simulation using the edited copy of the `cpw1.scs` file in “Simulating Coplanar Waveguides with the Generated Subcircuit Macromodel File” on page 585.

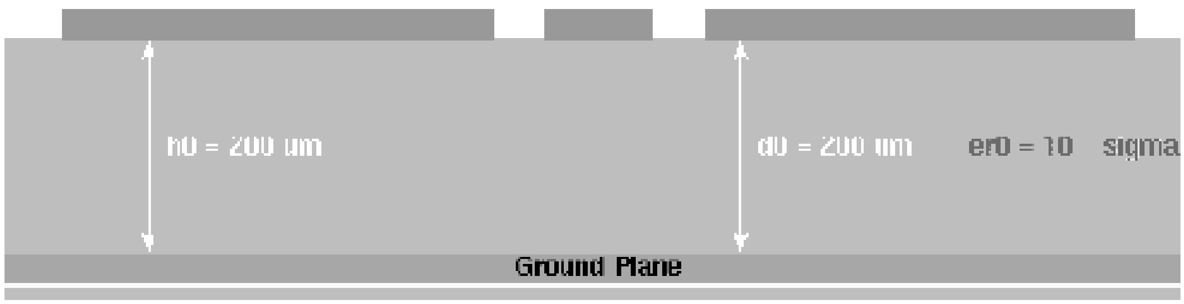
General Theory of Coplanar Waveguide Analysis

You can see from Example 7-2 on page 555 that the size of the LRCG matrixes in file `w_linecpw1.dat` is 1 $*$1 matrix even though the *No. of Lines* value in the LMG GUI is 3. This happens because you selected *CPW* for *Transmission Line Type*. For a single-line coplanar waveguide system, as shown in Figure 7-2 on page 558, the first and last lines are regarded as ground planes and the single, narrow line in the middle is regarded as the signal line.

Generally, a strict coplanar analysis is quite complicated and the parameter extraction requires information about the equivalent magnetic source solver. LMG does not perform this type of analysis. Rather, LMG first processes a coplanar waveguide as a system of regular transmission lines and then performs the size reduction.

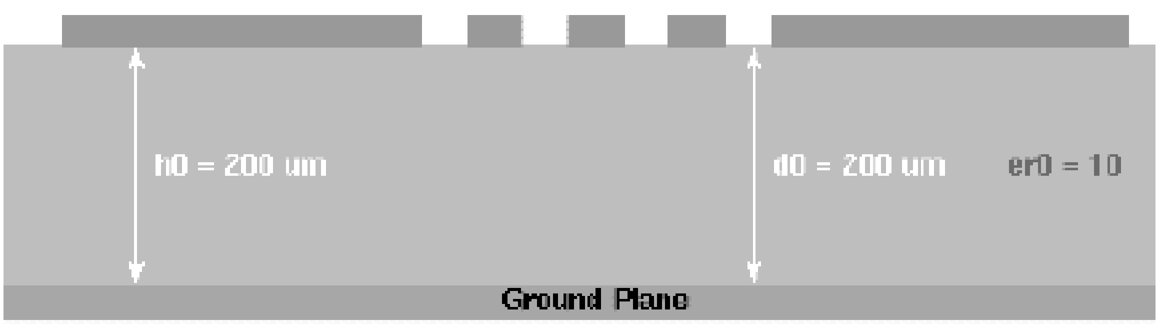
When processing the single-line CPW system shown in Figure 7-2 on page 558, LMG first treats this CPW as a regular 3-line system. It then sets the voltages of the first and third lines to zero (v1=0 and v3=0) and performs the parameter extraction and model generation.

Figure 7-2 Single Line Coplanar Waveguide System



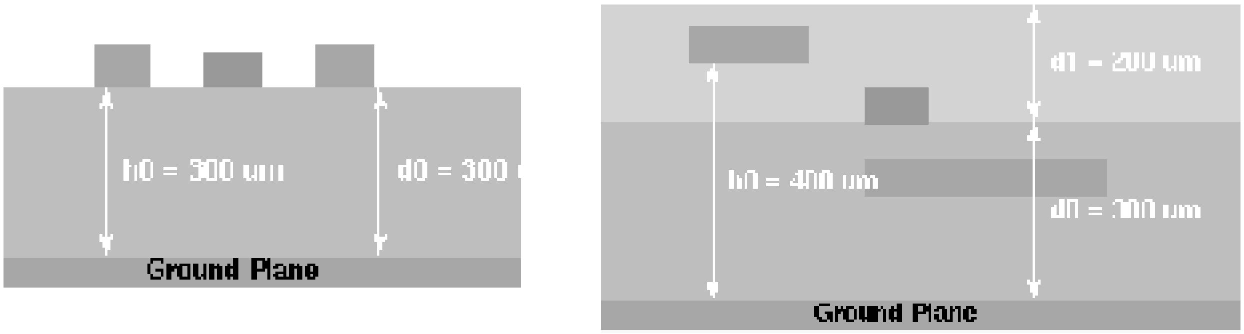
LMG can also handle a multi-line coplanar waveguide system such as the one shown in Figure 7-3.

Figure 7-3 Multi-Line Coplanar Waveguide System



In a general sense, you can use a CPW transmission line to process various side-shielding lines such as the two side-shielding line systems shown in Figure 7-4.

Figure 7-4 Side-Shielding Lines Handled by the CPW Line Type



In the example systems shown in Figure 7-4, the first and last lines are always considered to be grounded. In the LMG GUI, they display in the ground plane color.

In principle, for a CPW, an n-line system will produce $m = n - 2$ order modeling. This means that the LRCG file size is m times m. LMG performs the size reduction automatically and bases lumped models on the reduced-size LRCG file.

Simulating Coplanar Waveguides with the Generated LRCG File

To simulate a Coplanar Waveguide, or CPW, use the LRCG file, `w_linecpw1.dat,` you created, in Section "Using LMG to Obtain Subcircuit Macromodel and LRCG Files" on page 550. Then add the LRCG file to the `mtline` macromodel in the `CPW_tlineSimple` schematic. To do this you will need the absolute path to the LRCG file you noted in Section "Generating the Subcircuit and LRCG Files" on page 554.

This example uses the circuit `CPW_tlineSimple` from your editable copy of the `rfExamples` library (*my_rfExamples* here). This library also includes other sample circuits used in this manual. If you need assistance setting up or accessing your editable copy of the `rfExamples` library, refer to Chapter 3, "Setting Up for the Examples."

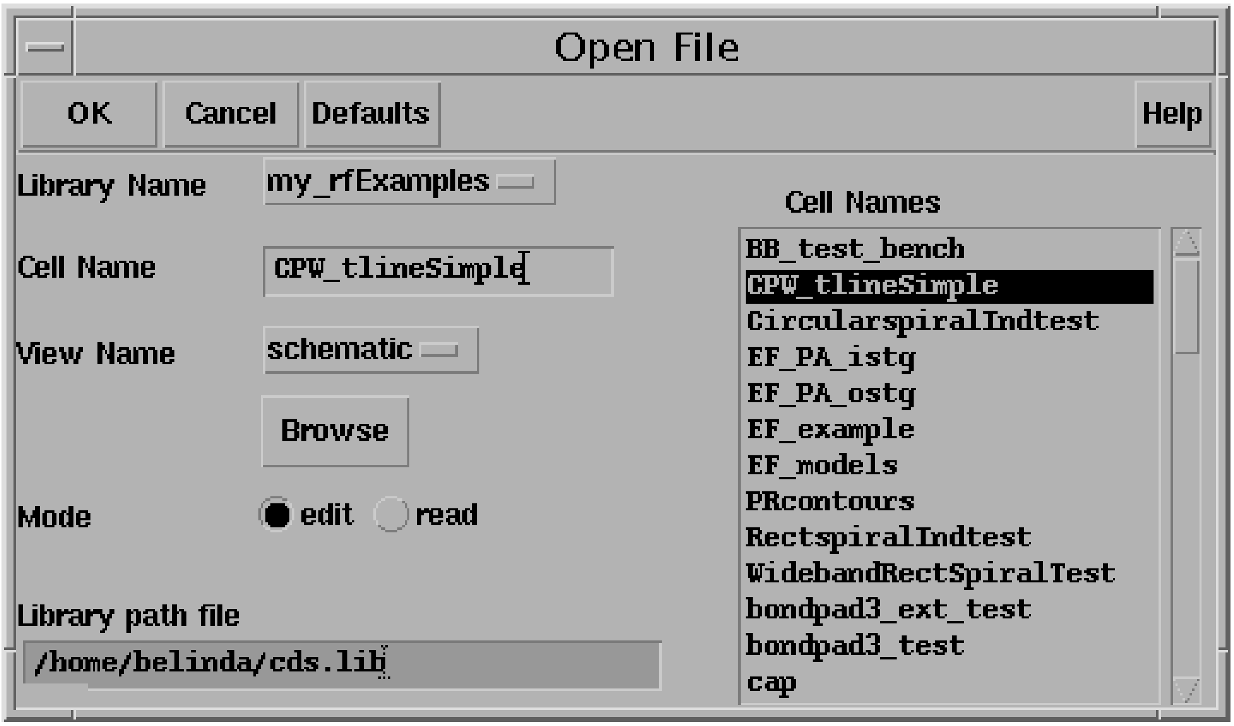
Opening the `CPW_tlineSimple` Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

   The Open File form appears.

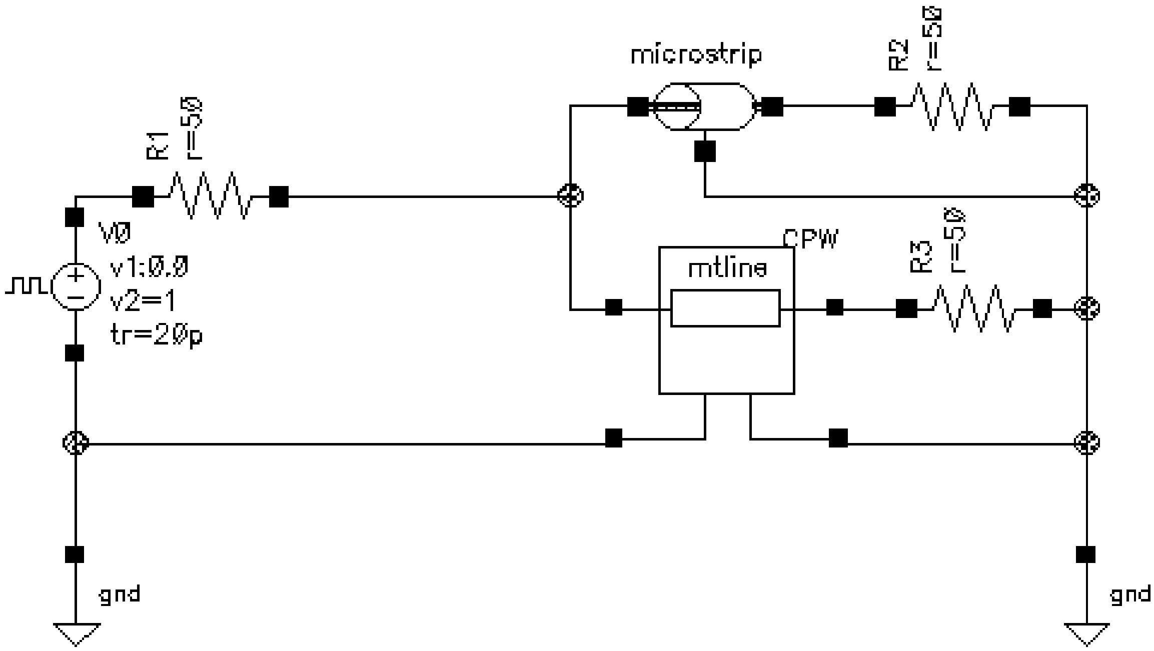
2. In the Open File form, choose *my_rfExamples* in the *Library Name* cyclic field and choose *CPW_tlineSimple* in the *Cell Names* list box.

The completed Open File form appears like the one below.



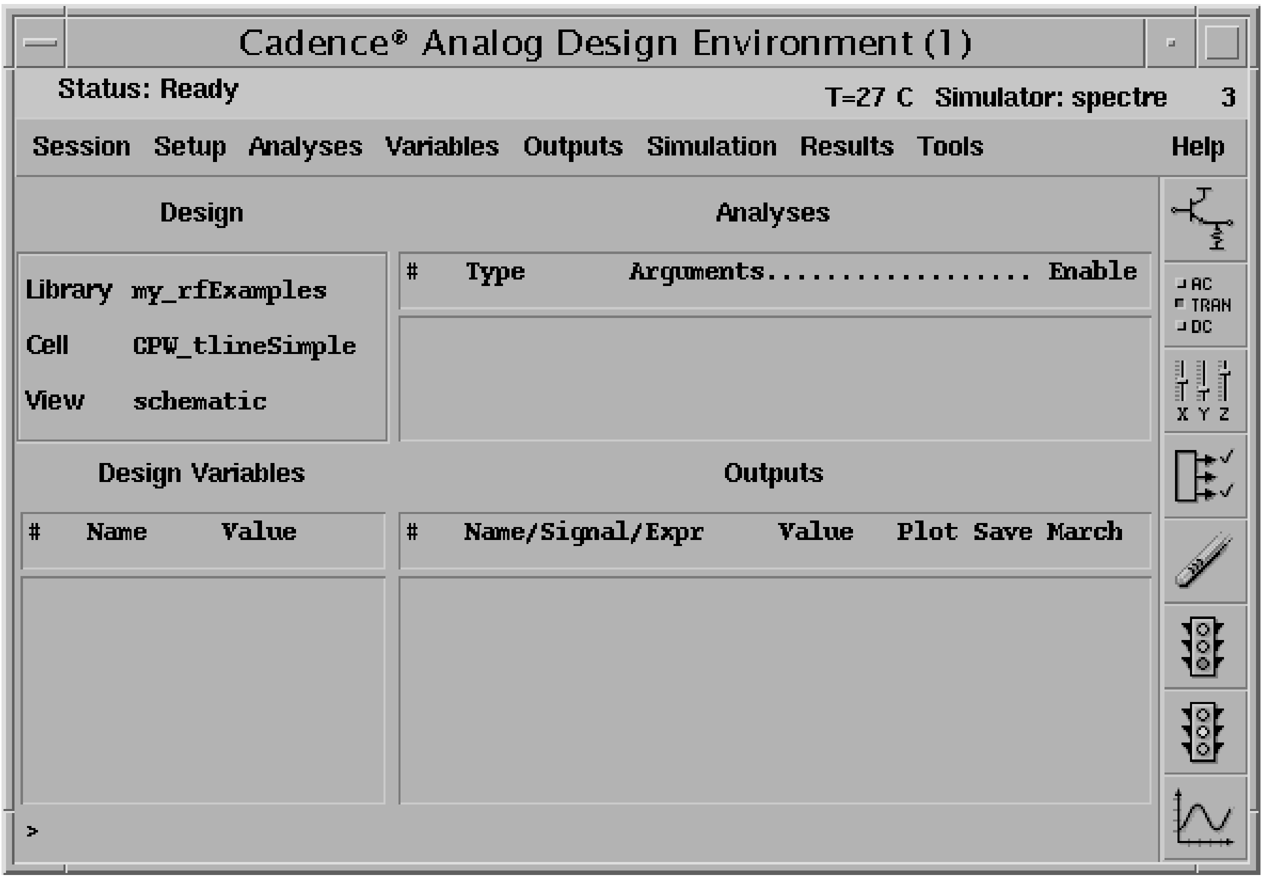
3. Click *OK*.

The Schematic window for the `CPW_tlineSimple` circuit appears.



4. In the Schematic window, choose *Tools—Analog Environment*.

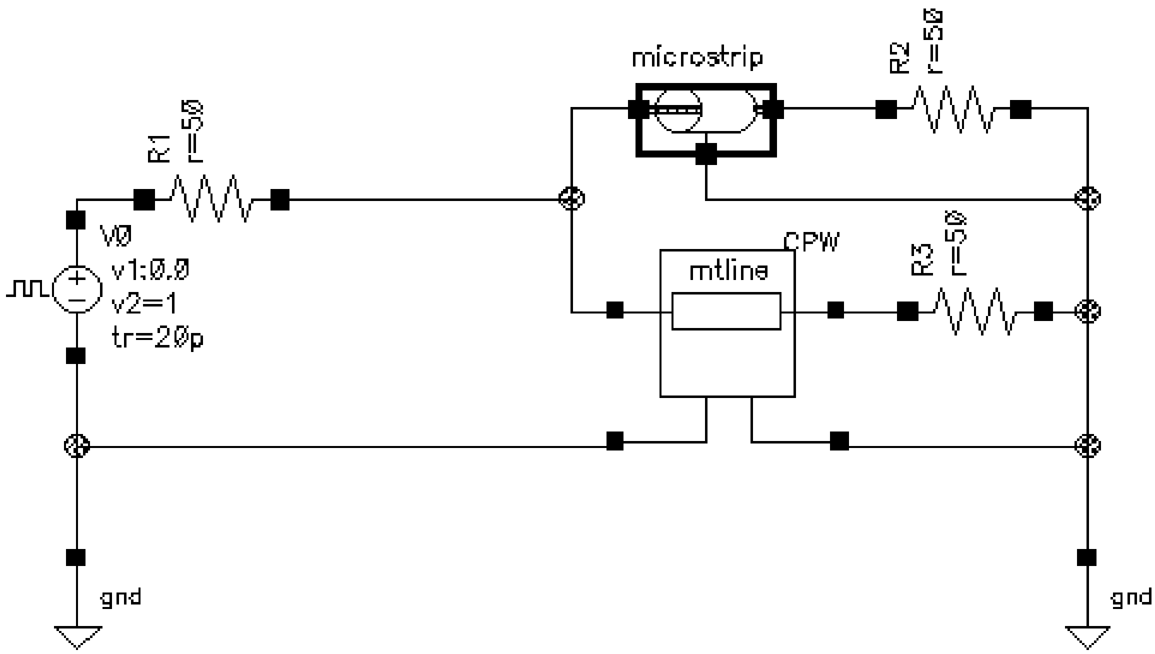
The Simulation window for the `CPW_tlineSimple` circuit opens.



Viewing the Properties of the Microstrip Component

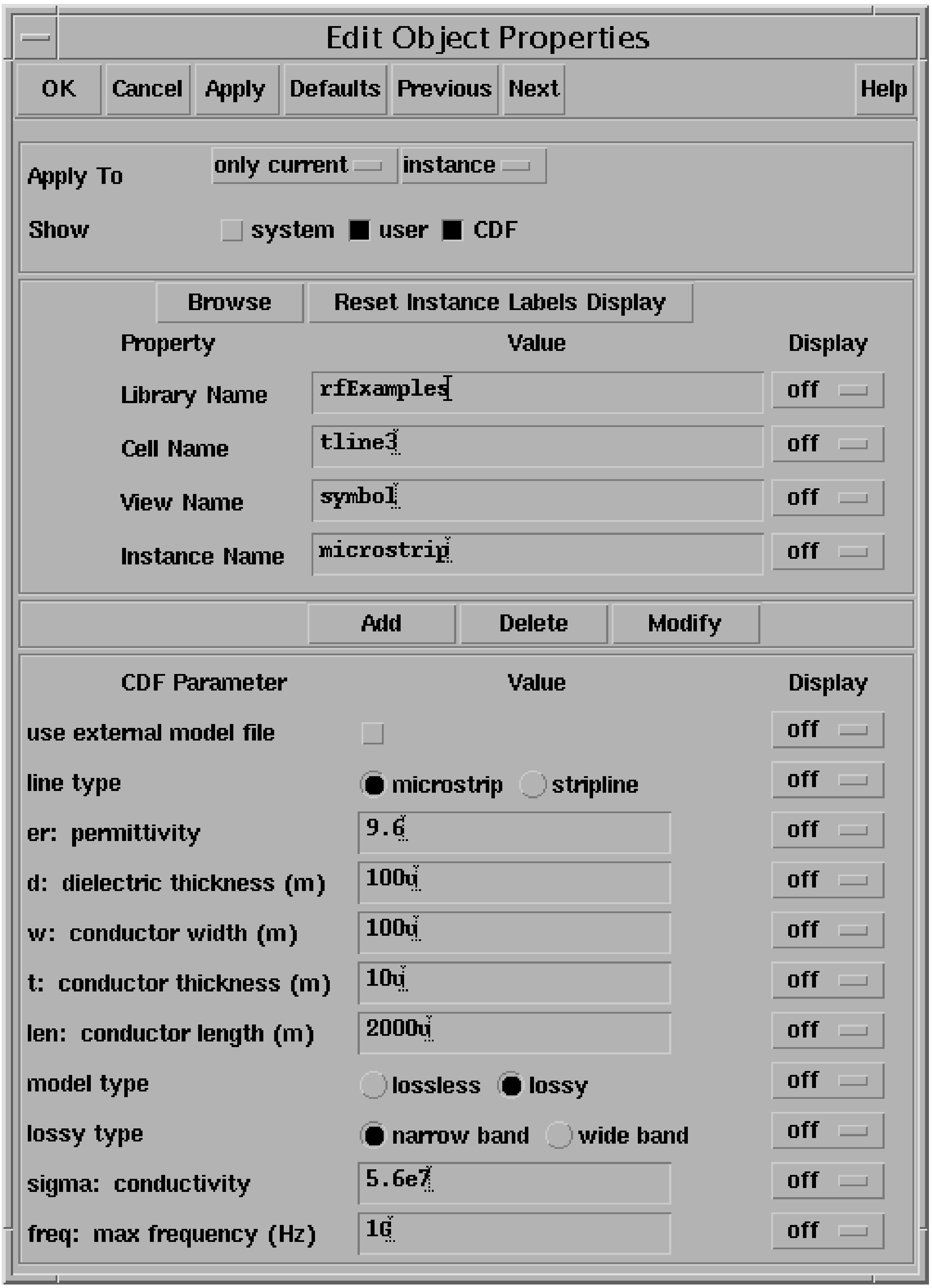
1. In the Schematic window, click on the `tline3` component symbol labeled *microstrip* located at the top of the schematic.

The *microstrip* symbol is now selected.



2. With the *microstrip* (`tline3`) transmission line symbol selected in the Schematic window, choose *Edit—Properties—Objects* to open the Edit Object Properties form for the *microstrip*.

The *Edit—Properties—Objects* form for the *microstrip* appears similar to the following.

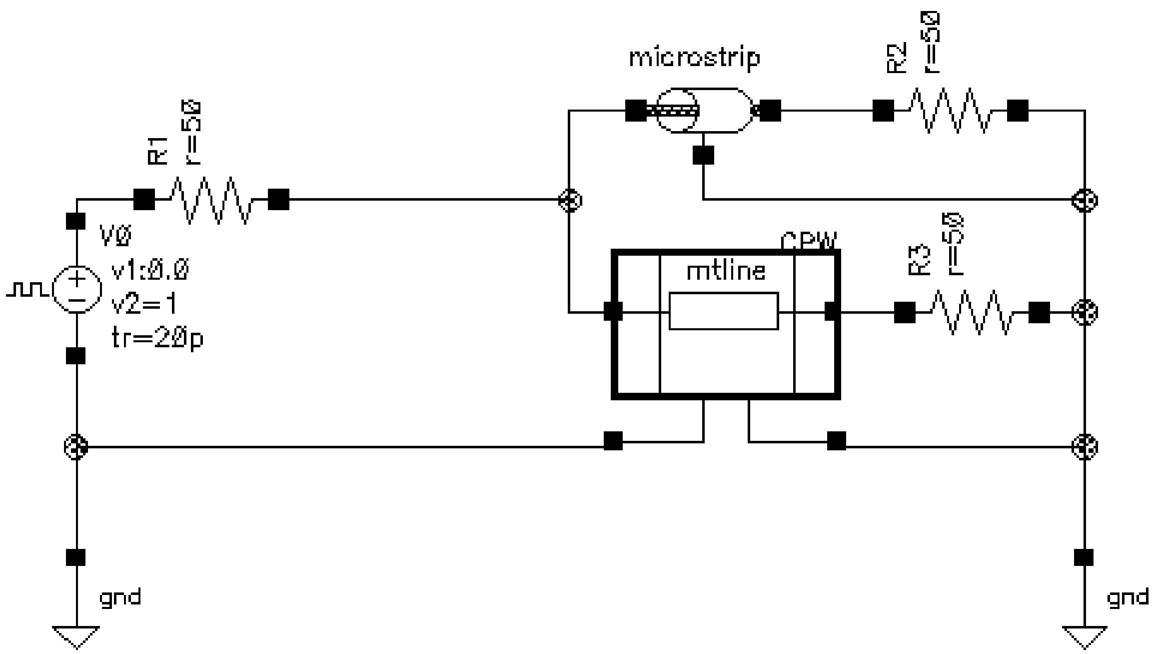


3. In the Edit Object Properties form for the *microstrip*, click *OK*.

Associating the LRCG File with the `Mtine` Component

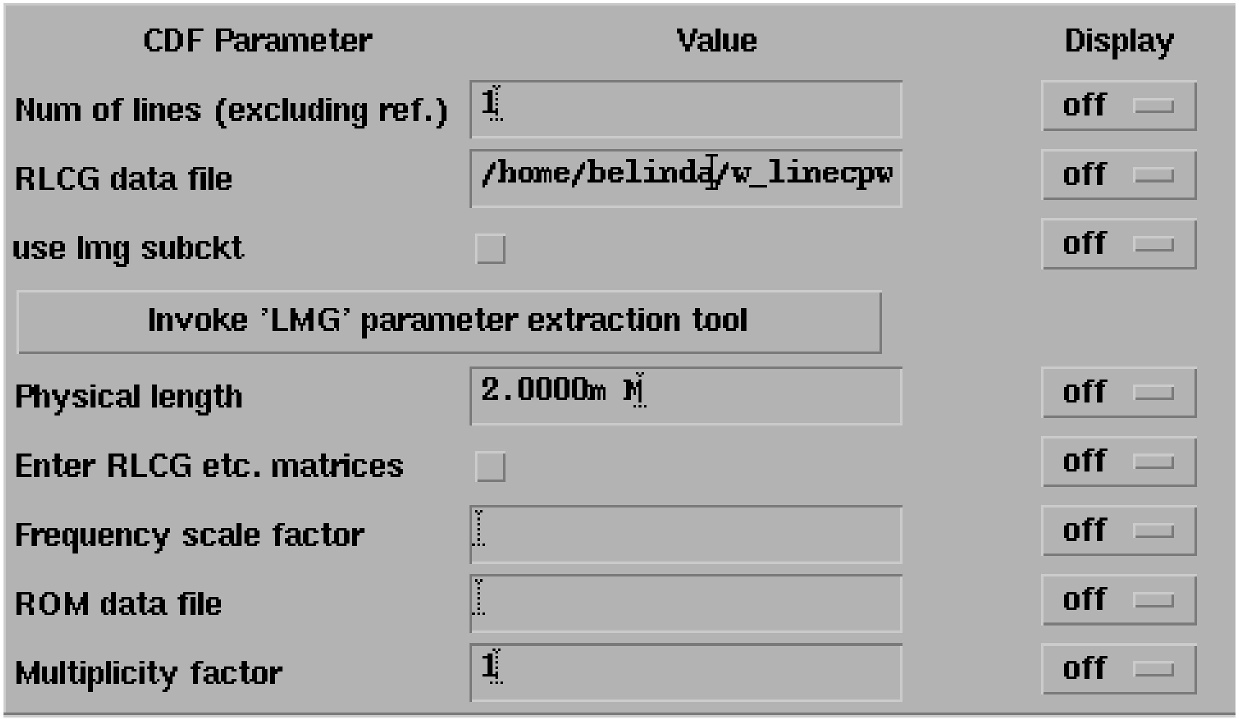
1. In the Schematic window, select the *CPW* (`mtline)` transmission line symbol at the center of the schematic.

   The *CPW* symbol is now selected.



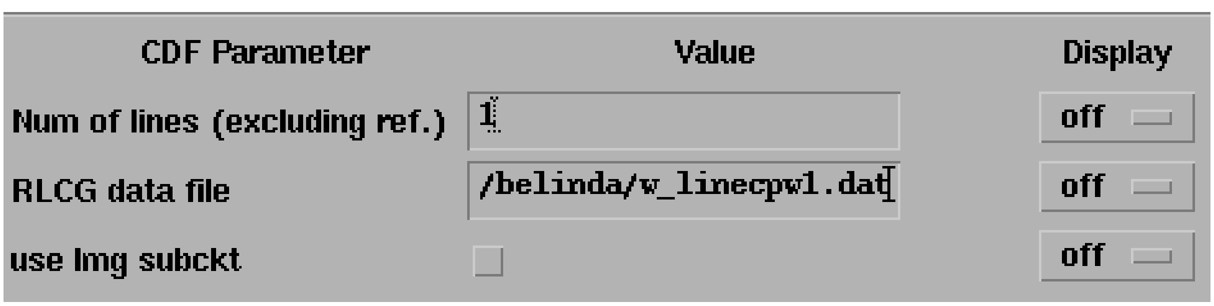
2. With the *CPW* symbol selected, choose *Edit—Properties—Objects* to open the Edit Object Properties form for the *CPW*.

The CDF parameter section of the CPW's Edit Object Properties form appears similar to the following.



3. Edit the *CDF Parameter* data field value for *RLCG data file* by replacing the current contents of the field with the full, absolute path to the LRCG data file, `w_linecpw1.dat,` you generated with LMG in "Generating the Subcircuit and LRCG Files" on page 554.

   For example, enter `/home/belinda/w_linecpw1.dat`



4. In the *Edit—Properties—Objects* form, click *OK*.

When you click *OK*, the form closes and the `w_linecpw1.dat` file is saved in the schematic editor. When you simulate with the CP*W* component, the `w_linecpw1.dat` LRCG data file is used.

5. In the schematic window, choose *Design—Check and Save* for the `CPW_tlineSimple` schematic.

You can now set up and run a simulation using the new `mtline` macromodel for the *CPW* in the analog design environment.

Simulating the CPW

1. In the Simulation window, choose *Analyses—Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click on *tran*.

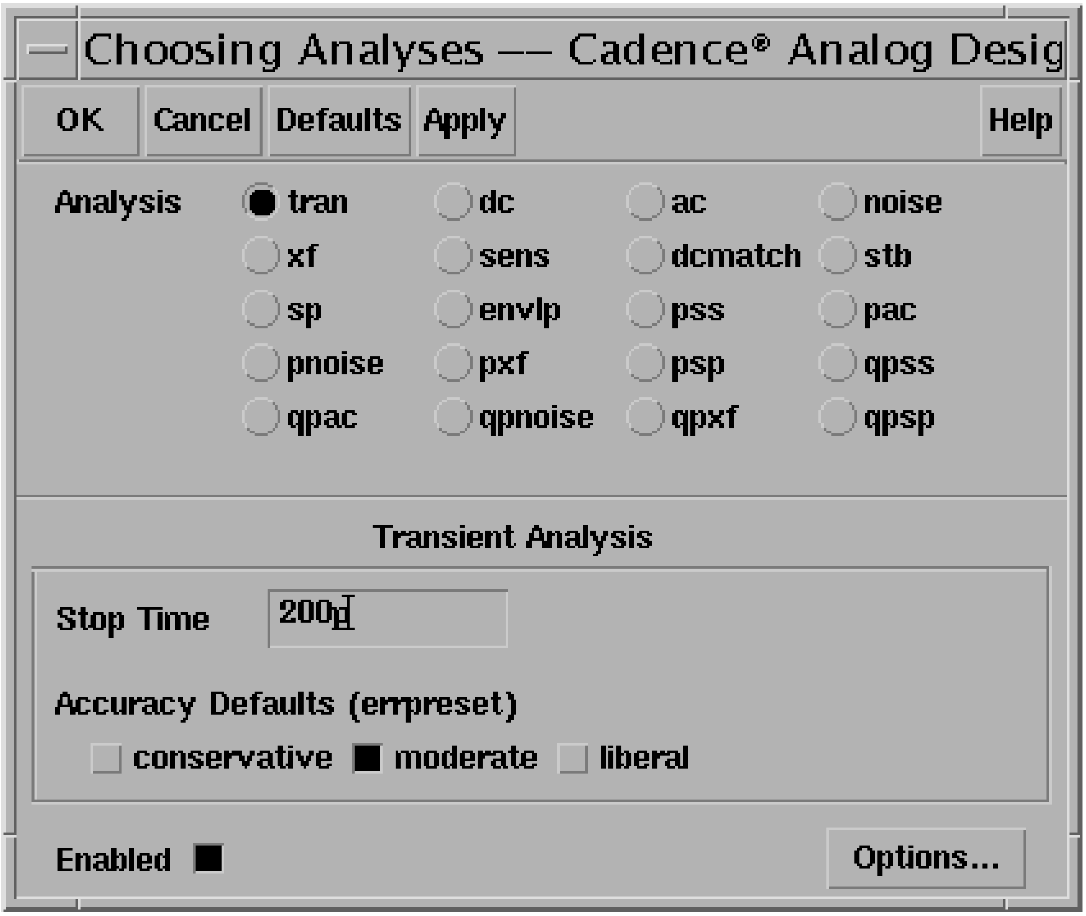
   The form displays options needed for `transient` analysis.

3. In the *Stop Time* field, type your best estimate ....

   Type `200p` in this example.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
5. Verify that *Enabled* is highlighted.

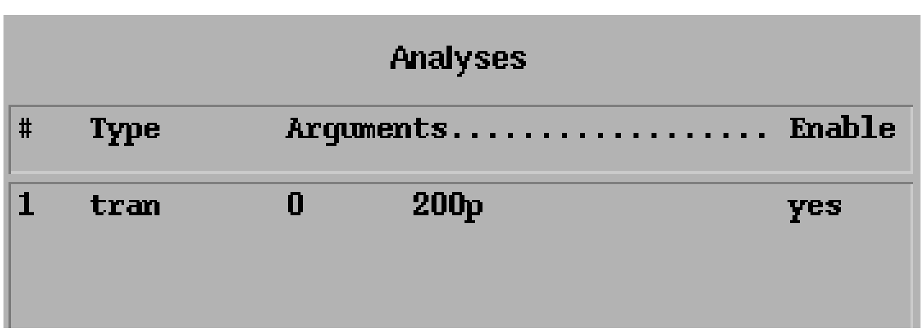
The Choosing Analyses form appears below.



6. Click *Apply* to check for setup errors for the Transient analysis.

7. Click *OK*.

The `Transient` analysis options you choose appear in the *Analyses* list box in the Simulation window.



Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Coplanar Waveguide Signals

➤ To open the Direct Plot form, choose *Results—Direct Plot—Main Form* in the Simulation window.

The Waveform window and the Direct Plot form appear.

In the Direct Plot form do the following.

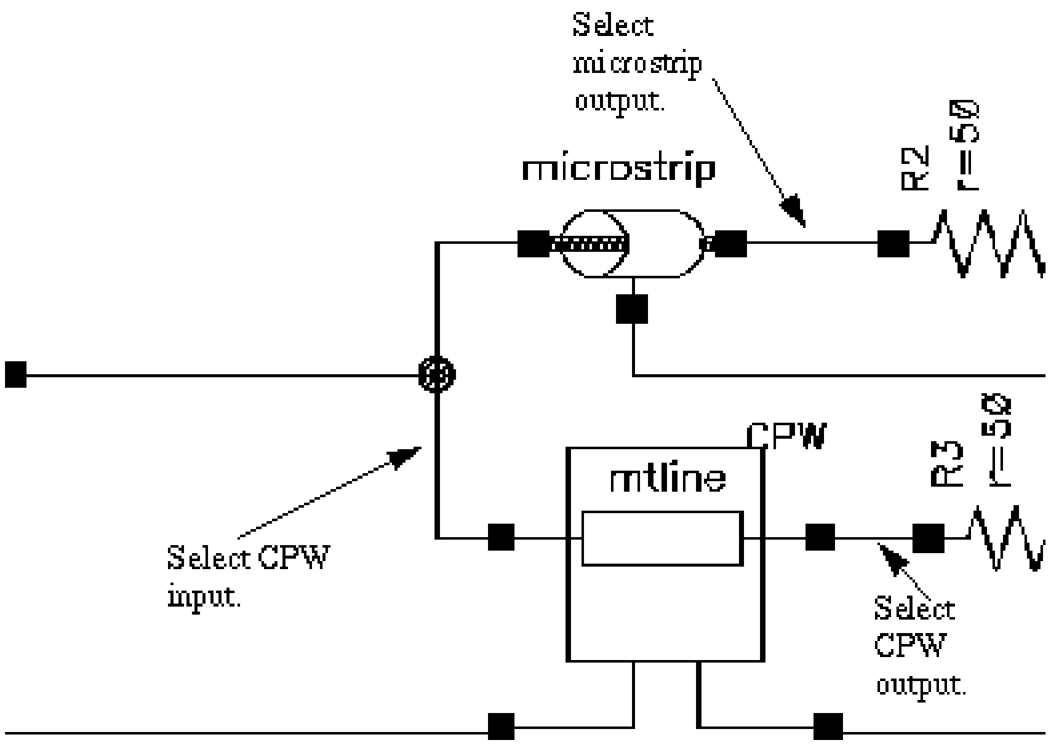
1. Highlight *Append* for *Plot Mode*.
2. Highlight *Tran* for *Analysis*.
3. Highlight *Voltage* for *Function*.

The filled-in form looks like this.

4. Follow the prompt at the bottom of the Direct Plot form.

```
> Select Net on Schematic...
```

Select the following nets in the `CPW_tlineSimple` schematic.



In the Schematic window, select the following nets.

a. Click on the CPW input.

b. Click on the microstrip output.

c. Click on the CPW output.

d. Press *Esc* to stop selecting nets.

The plot in Figure 7-5 appears in the Waveform window.

Figure 7-5 Signals Passing Through the CPW and the 2000u Microstrip

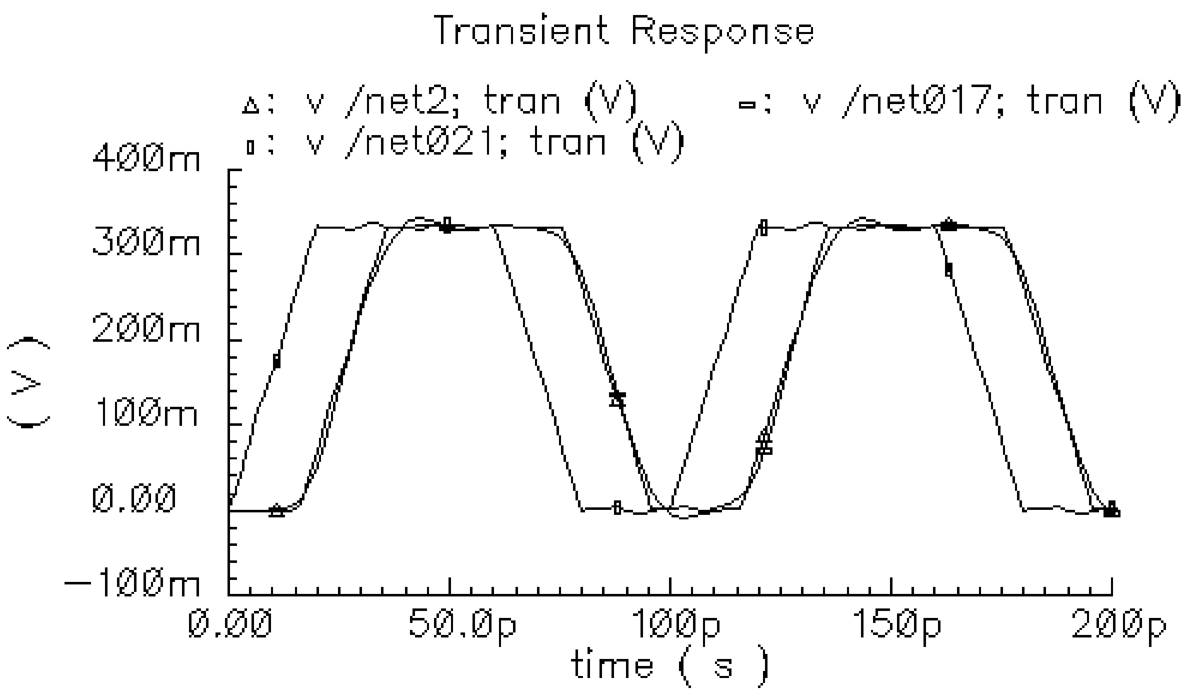


Figure 7-5 shows that frequency dispersion for the CPW is better than it is for the microstrip line. From Figure 7-5, the waveform shape of *net2* (the CPW output) is better than the shape of *net017* (the microstrip output). The phase delays for both lines are correct. You can also see that the input signal is slightly corrupted by the reflection from the microstrip line. This is evident from the ripple on *net021*.

Verifying Input Signal Corruption by Microstrip Line Reflection

Use the following procedures to verify that the input signal is slightly corrupted by reflection from the microstrip line.

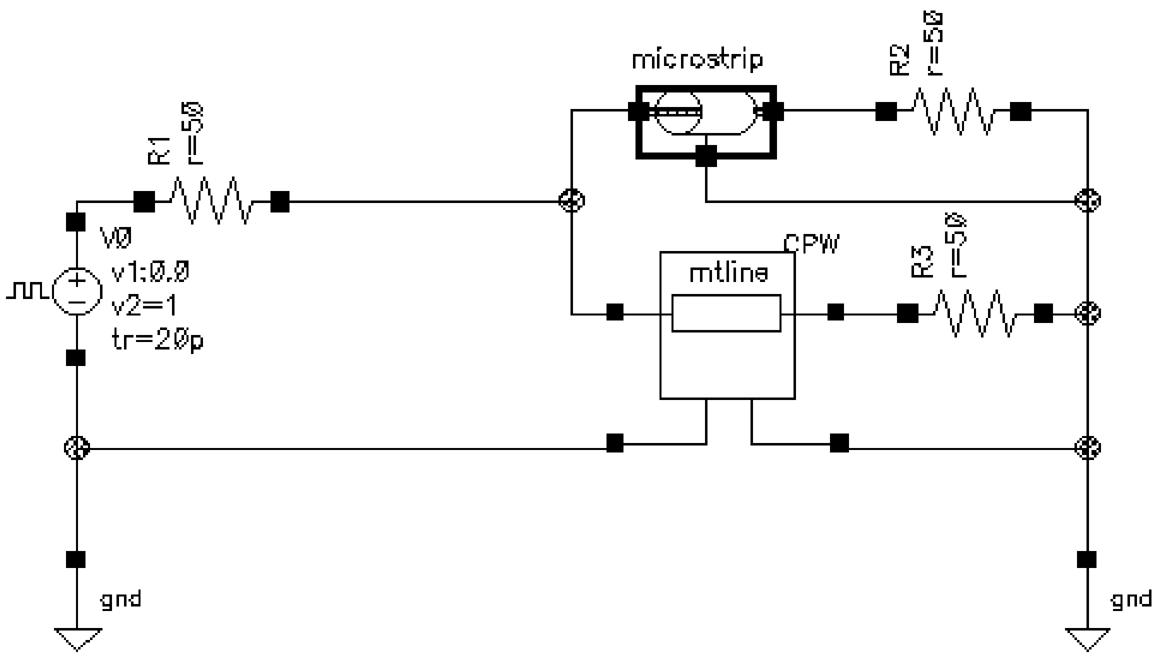
- ❑ In Shortening the Length of the Microstrip Component, decrease the microstrip conductor length from 200u to 10u.
- ❑ In “Increasing the Length of the Microstrip Component” on page 578, increase the microstrip conductor length from 200u to 100000u.

In both cases, simulate and compare the transient responses in Figures 7-5, 7-6 and 7-7.

Shortening the Length of the Microstrip Component

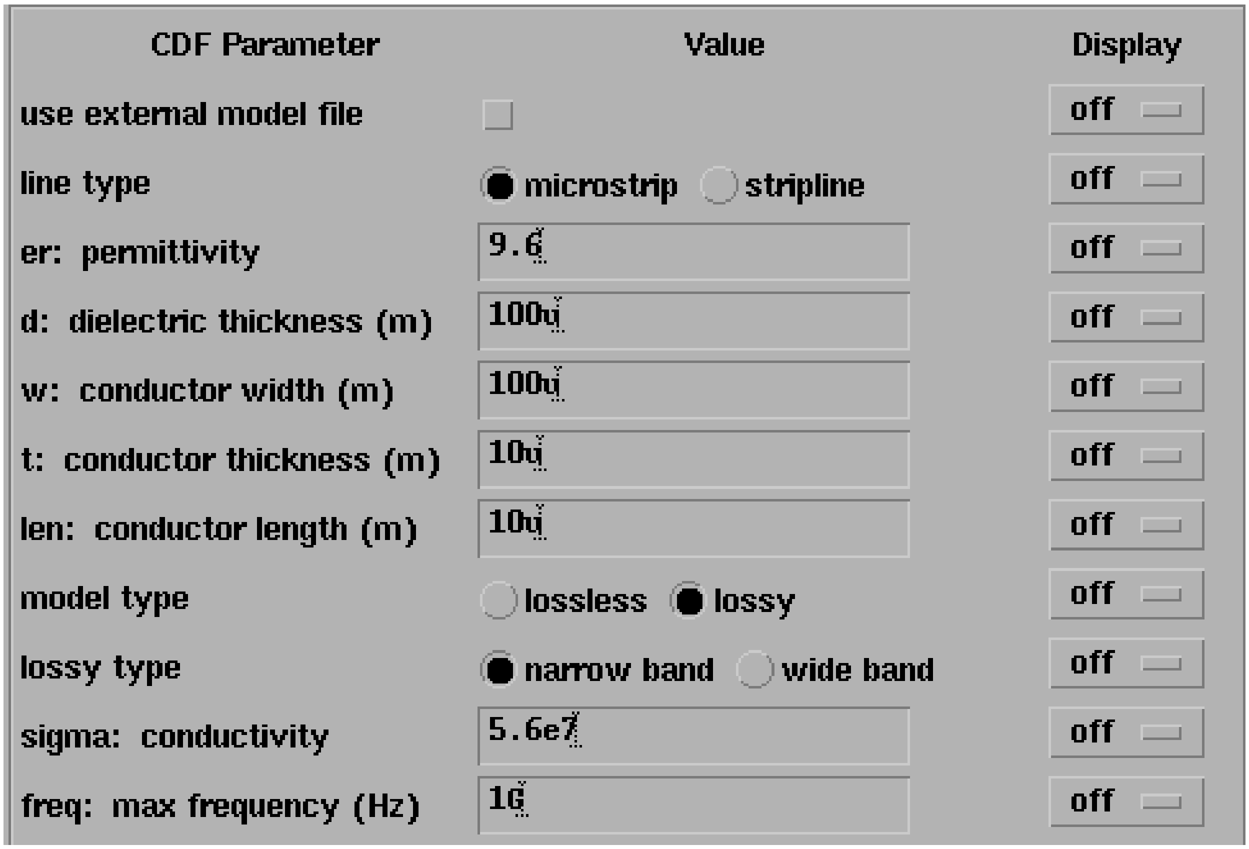
1. In the Schematic window, click on the *microstrip* component symbol located at the top of the schematic.

   The `tline3` symbol is now selected.



2. With `tline3` the transmission line symbol selected, choose *Edit—Properties—Objects* to open the Edit Object Properties form for the *microstrip*.

The *CDF Parameter* section of the Edit Objects Properties form for the *microstrip* appears similar to the following.



CDF Parameter	Value	Display
use external model file	☐	off
line type	● microstrip ○ stripline	off
er: permittivity	9.6	off
d: dielectric thickness (m)	100u	off
w: conductor width (m)	100u	off
t: conductor thickness (m)	10u	off
len: conductor length (m)	10u	off
model type	○ lossless ● lossy	off
lossy type	● narrow band ○ wide band	off
sigma: conductivity	5.6e7	off
freq: max frequency (Hz)	1G	off

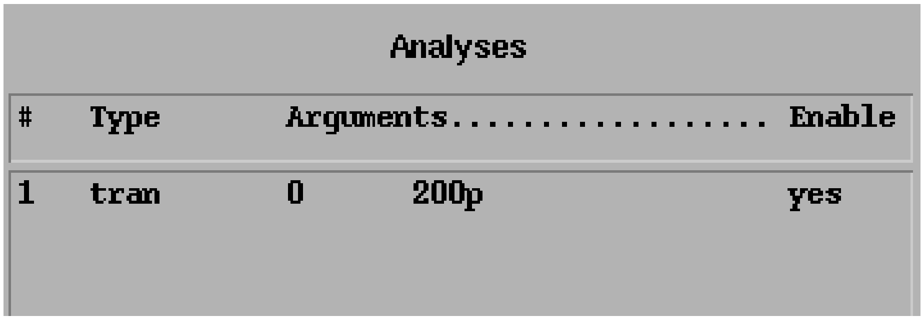
3. Type `10u` in the *len: conductor length (m)* field.
4. In the Edit Object Properties form for the *microstrip*, click *OK*.
5. In the schematic window, choose *Design—Check and Save* to check and save the `CPW_tlineSimple` schematic.

You can now set up and run a simulation in the analog design environment using the new `tline3` macromodel line length for the *microstrip* component.

Simulating the CPW

1. in the Simulation window, verify that the `transient` analysis is still enabled.

The `transient` analysis appear in the *Analyses* list box.



Running the Simulation

1. In the Simulation window, choose *Simulation—Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Coplanar Waveguide Signals

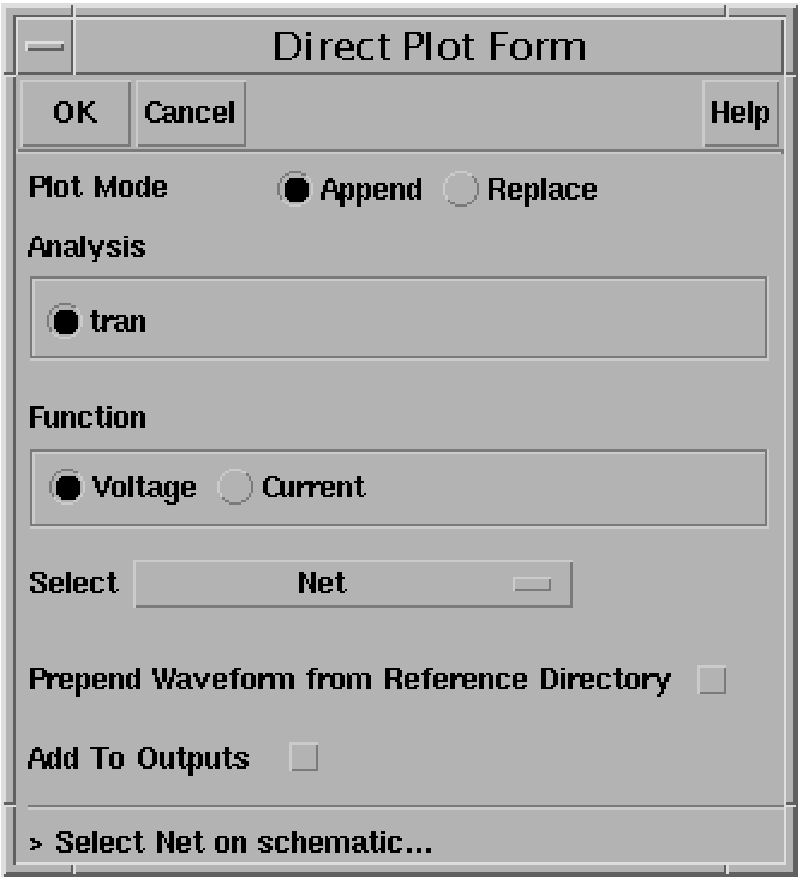
➤ To open the Direct Plot form, choose *Results—Direct Plot—Main Form* in the Simulation window.

The Waveform window and the Direct Plot form appear.

In the Direct Plot form do the following.

1. Highlight *Append* for *Plot Mode*.
2. Highlight *Tran* for *Analysis*.
3. Highlight *Voltage* for *Function*.

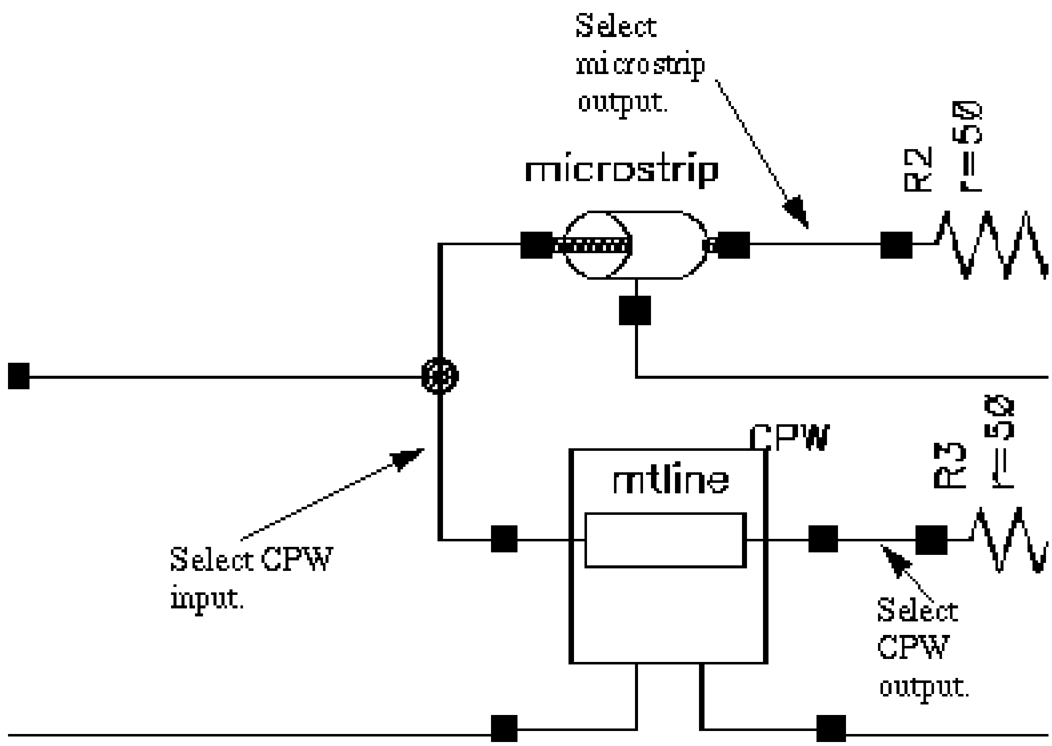
The filled-in form looks like this.



4. Follow the prompt at the bottom of the Direct Plot form.

```
> Select Net on Schematic...
```

Select the following nets in the `CPW_tlineSimple` schematic.



In the Schematic window, select the following nets.

a. Click on the CPW input.

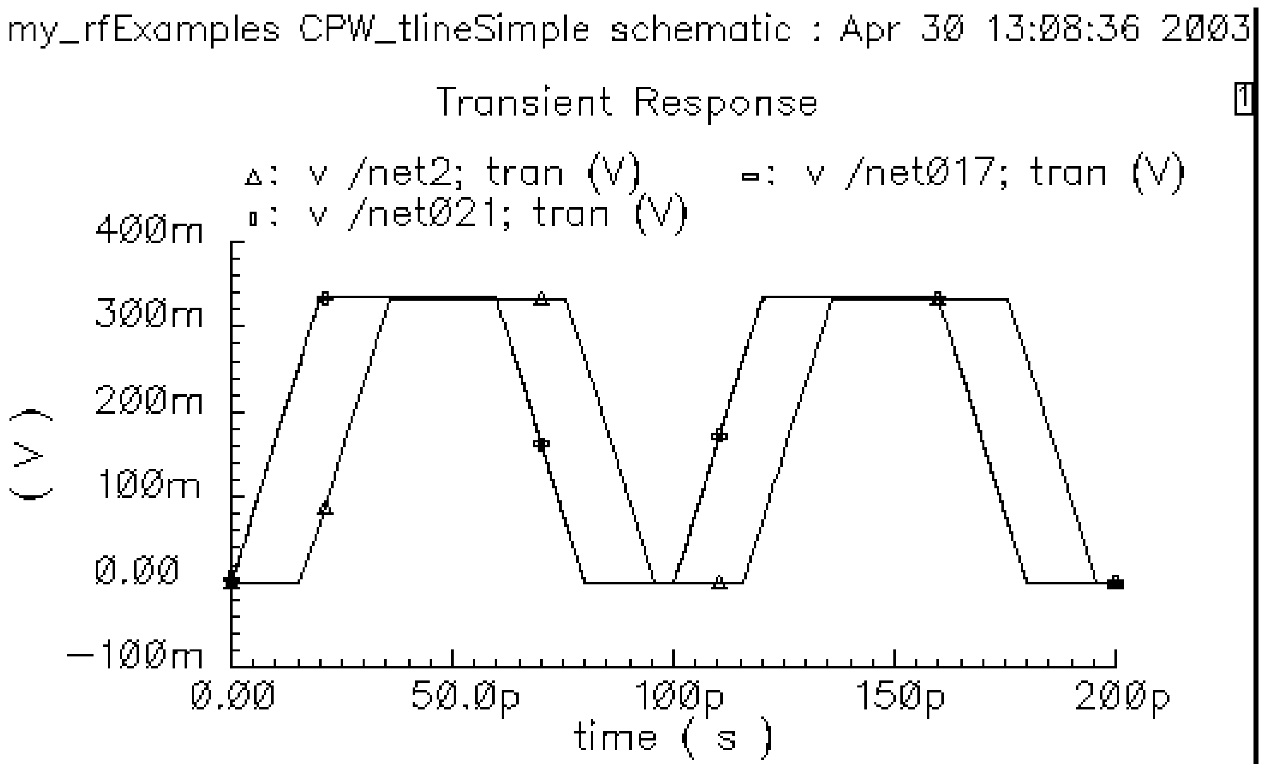
b. Click on the microstrip output.

c. Click on the CPW output.

d. Press *Esc* to stop selecting nets.

The plot in Figure 7-5 appears in the Waveform window.

Figure 7-6 Signals Passing Through the CPW and the 10u Microstrip



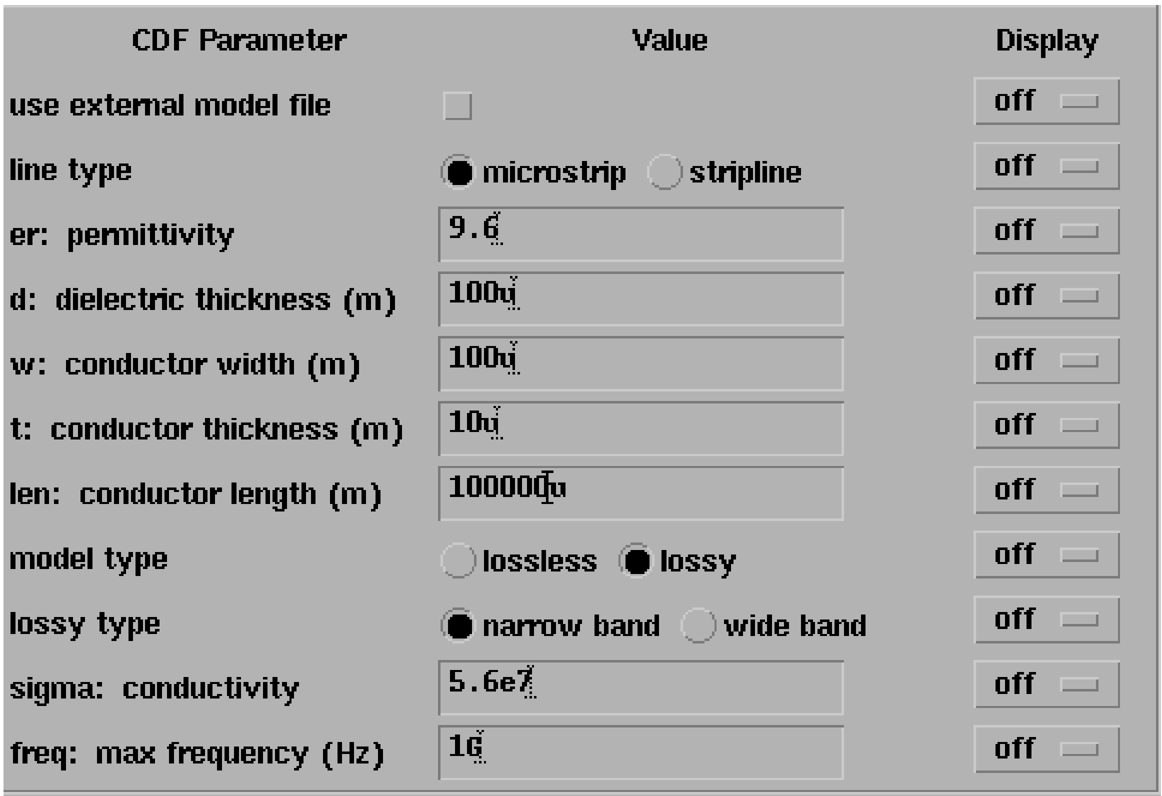
When you compare Figure 7-5 on page 572 with Figure 7-6, you can easily see that the shapes in Figure 7-6 are much cleaner than the shapes in Figure 7-5.

In Figure 7-6, the microstrip line is only 10 um long. Since it is so short, in fact almost a shunt interconnect between *net021* and *net017,* there is no reflection from the *microstrip* to corrupt the input signal. Because the *CPW* is a very low frequency dispersion line, the *CPW* output (*net2*) is almost identical to its input (*net021*) except for the phase shift.

Increasing the Length of the Microstrip Component

1. In the Schematic window, click on the *microstrip* component symbol located at the top of the schematic.
2. With `tline3` the transmission line symbol selected, choose *Edit—Properties—Objects* to open the Edit Object Properties form for the *microstrip*.

The *CDF Parameter* section of the Edit Objects Properties form for the *microstrip* appears similar to the following.



3. This time type `100000u` in the *len: conductor length (m)* field.
4. Click *OK*.
5. In the schematic window, choose *Design—Check and Save* to check and save the `CPW_tlineSimple` schematic.

You can now set up and run a simulation in the analog design environment using the new `tline3` macromodel line length for the *microstrip* component.

Simulating the CPW

1. In the Simulation window, choose *Analyses – Choose*.

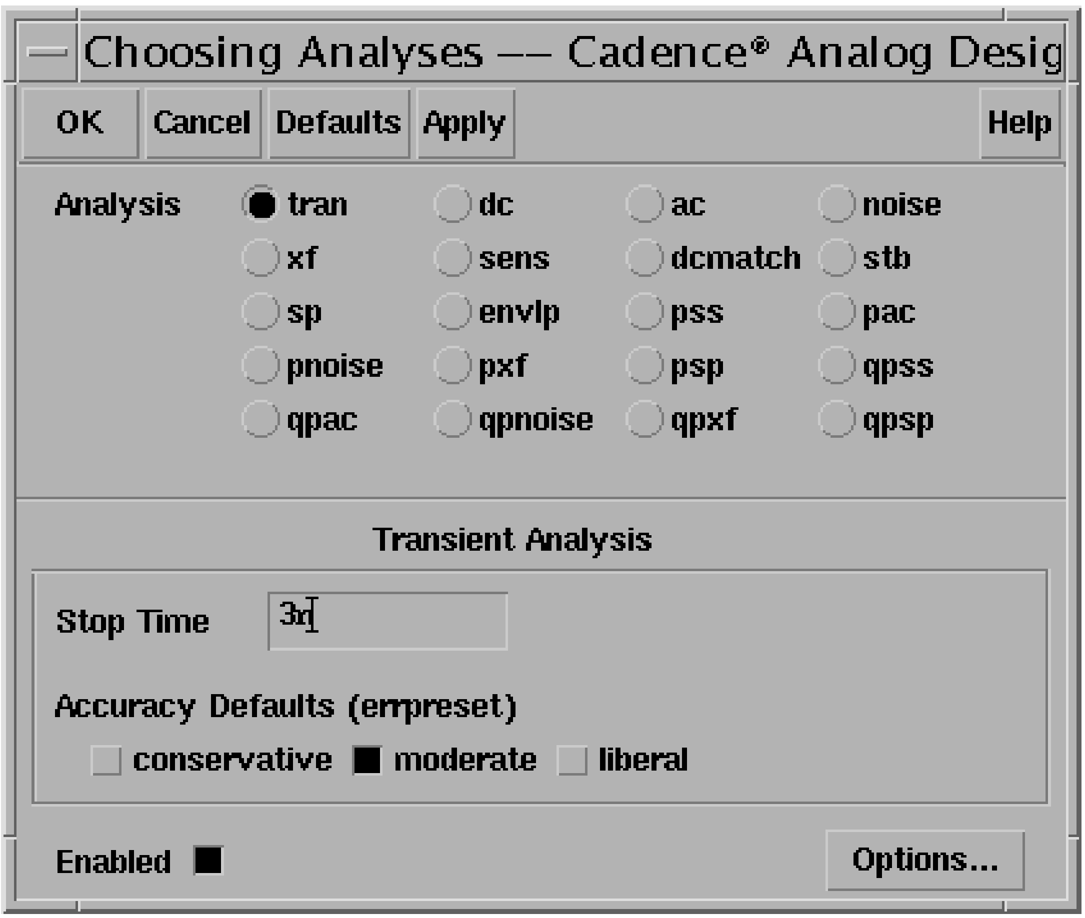
   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click on *tran*.

   The form displays options needed for `tran` analysis.

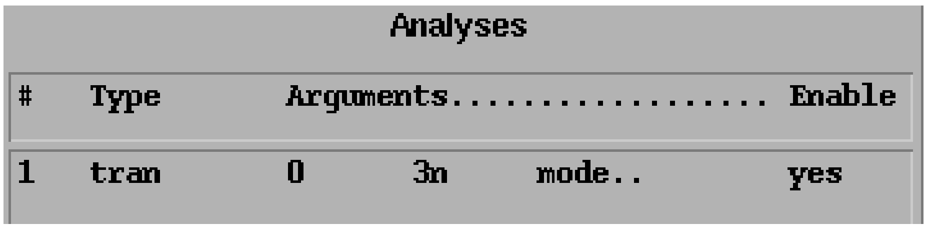
3. In the *Stop Time* field, type 3n.
4. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.
5. Verify that *Enabled* is highlighted.

   The Choosing Analyses form appears below.



6. Click *Apply* to check for setup errors for the Transient analysis.
7. Click *OK*.

The `Transient` analysis options you choose appear in the *Analyses* list box in the Simulation window.



Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Coplanar Waveguide Signals

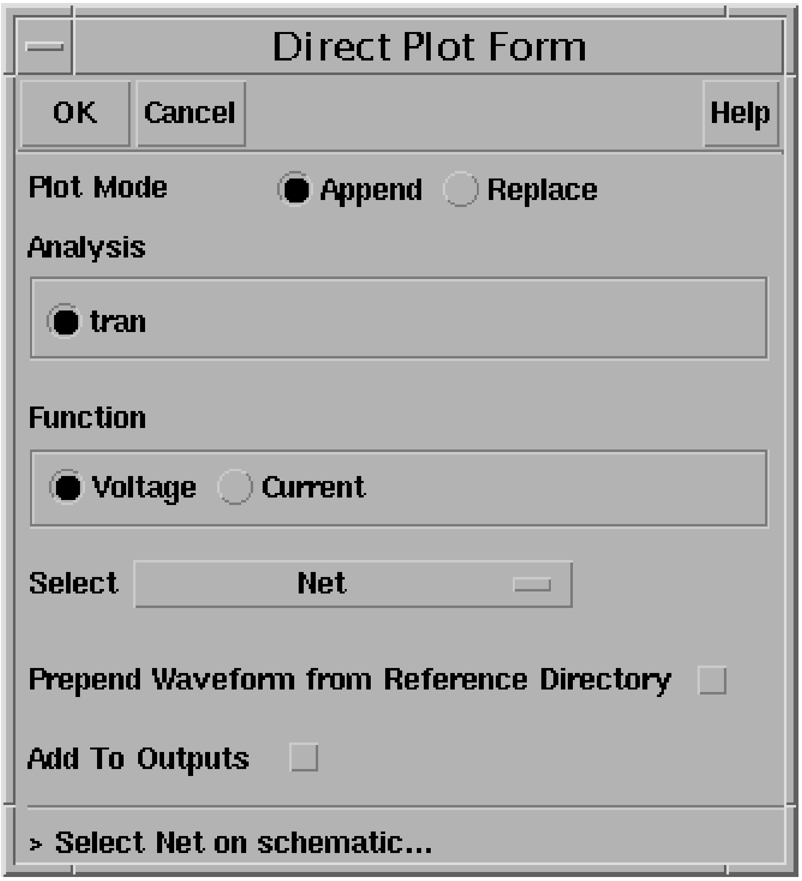
➤ To open the Direct Plot form, choose *Results – Direct Plot – Direct Plot* in the Simulation window.

The Waveform window and the Direct Plot form appear.

In the Direct Plot form do the following.

1. Highlight *Append* for *Plot Mode*.
2. Highlight *Tran* for *Analysis*.
3. Highlight *Voltage* for *Function*.

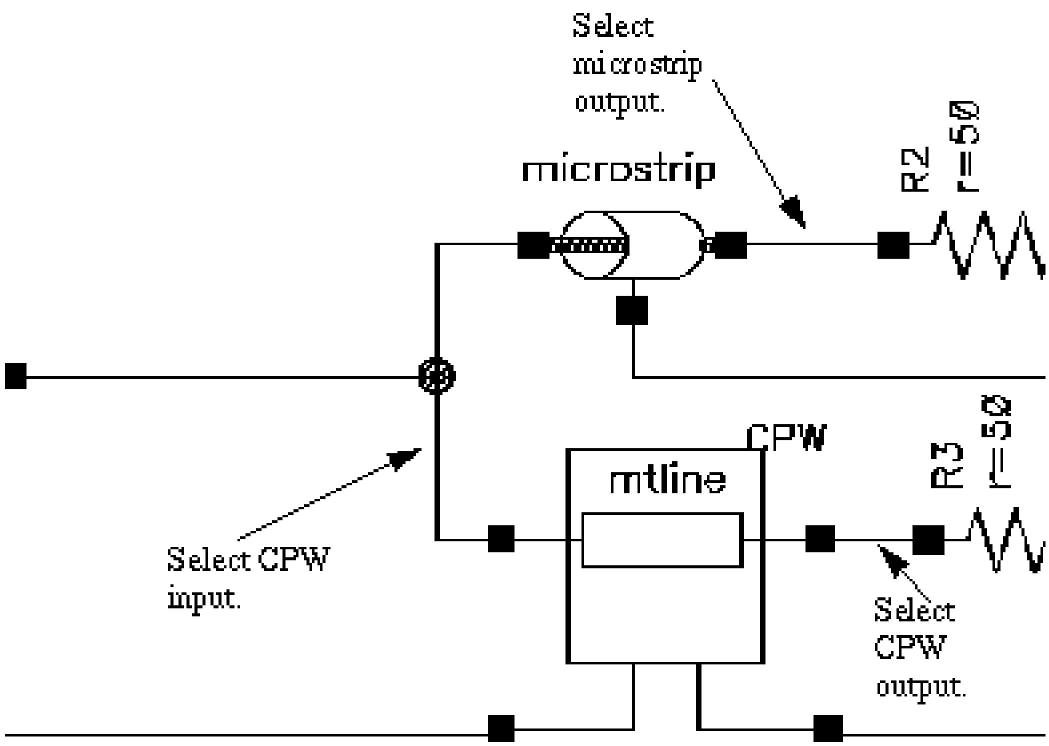
The filled-in form looks like this.



4. Follow the prompt at the bottom of the Direct Plot form.

```
> Select Net on Schematic...
```

Select the following nets in the CPW_tlineSimple schematic.



In the Schematic window, select the following nets.

a. Click on the CPW input.

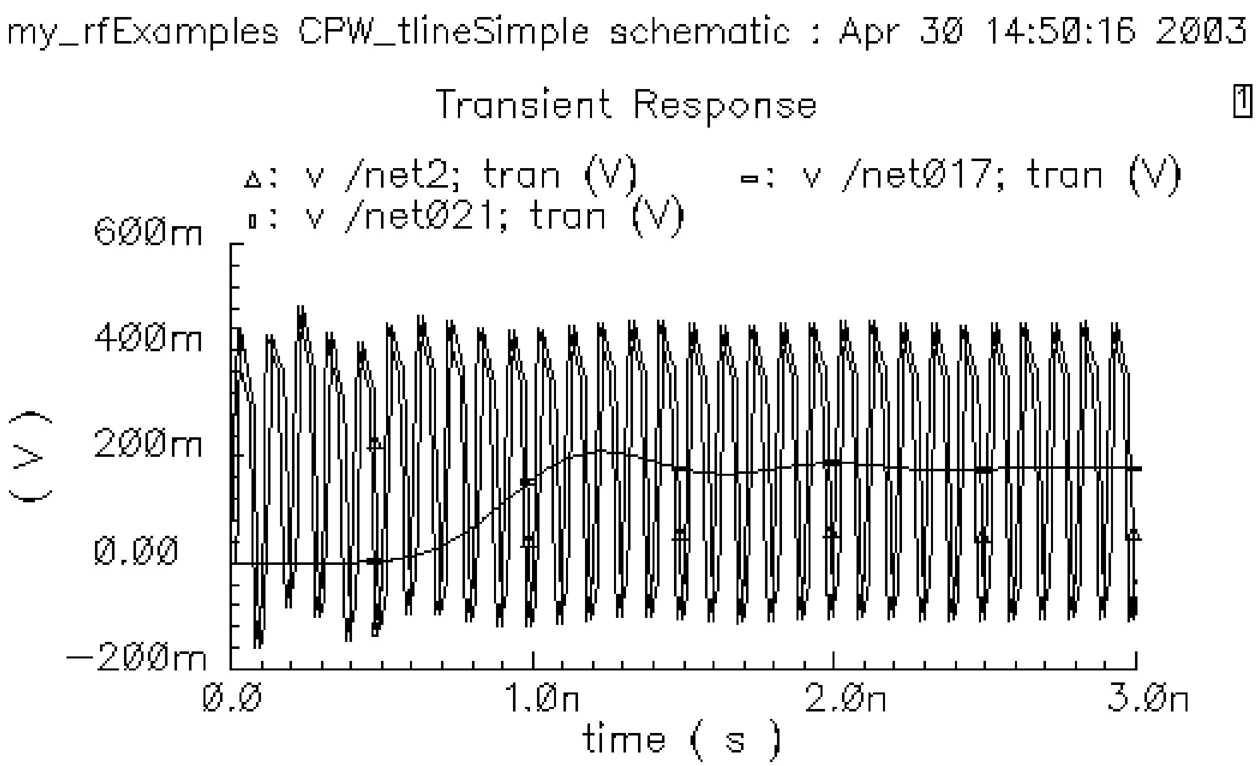
b. Click on the microstrip output.

c. Click on the CPW output.

d. Press Esc to stop selecting nets.

The plot in Figure 7-5 appears in the Waveform window.

Figure 7-7 Signals Passing Through the CPW and the 100000u Microstrip Line with a 3n Stop Time



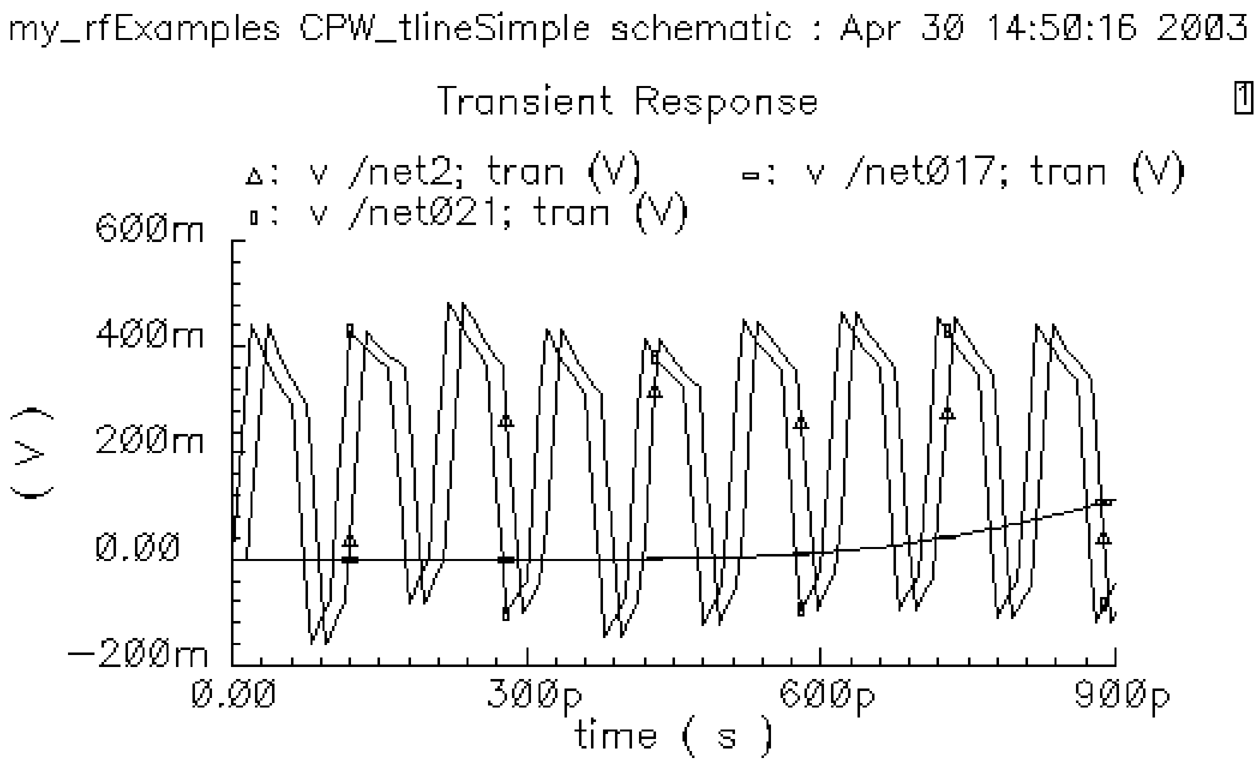
From Figure 7-7, you can see that now that the microstrip is longer (100mm) its output also has a longer time delay. Also due to the longer microstrip, its resistance is greater and its output amplitude is less. The input signal is corrupt due to reflection of the *CPW* and *microstrip* signals, but the *CPW* output is almost identical to its input except for some phase delay.

You can se this more clearly by zooming in on an area of Figure 7-7. In the Waveform window, choose *Zoom – Zoom In.*

5. Select an area to enlarge by clicking at the top left corner of the area to enlarge. Than draw a rectangle around the area and click at the bottom right corner.

The enlarged area displays in Figure 7-8.

Figure 7-8 Magnified View of Transient Response



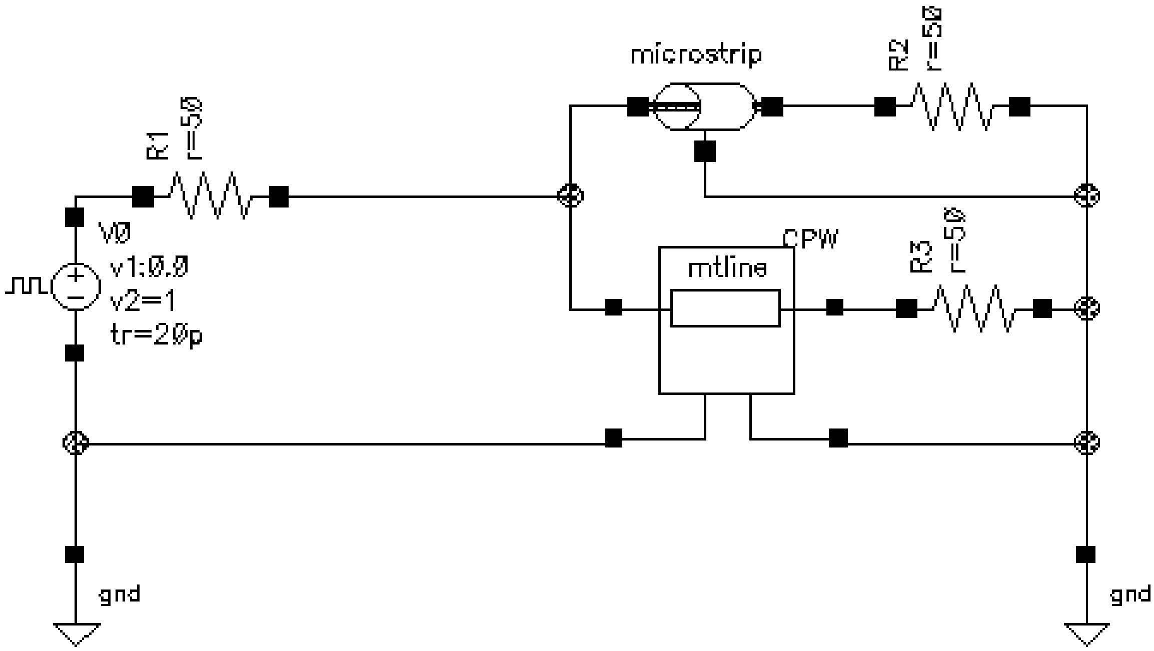
Simulating Coplanar Waveguides with the Generated Subcircuit Macromodel File

Simulate a Coplanar Waveguide, or CPW, using the macromodel subcircuit file, `cpw1.scs`, you created in Section “Using LMG to Obtain Subcircuit Macromodel and LRCG Files” on page 550. Add the `cpw1.scs` file to the CPW (`mtline`) component in the `CPW_tlineSimple` schematic. To do this you will need the absolute path to the `cpw1.scs` file you noted in Section “Generating the Subcircuit and LRCG Files” on page 554.

This example uses the circuit `CPW_tlineSimple` from your editable copy of the `rfExamples` library (*my_rfExamples* here). This library also includes other sample circuits used in this manual. If you need assistance setting up or accessing your editable copy of the `rfExamples` library, refer to Chapter 3, “Setting Up for the Examples.”

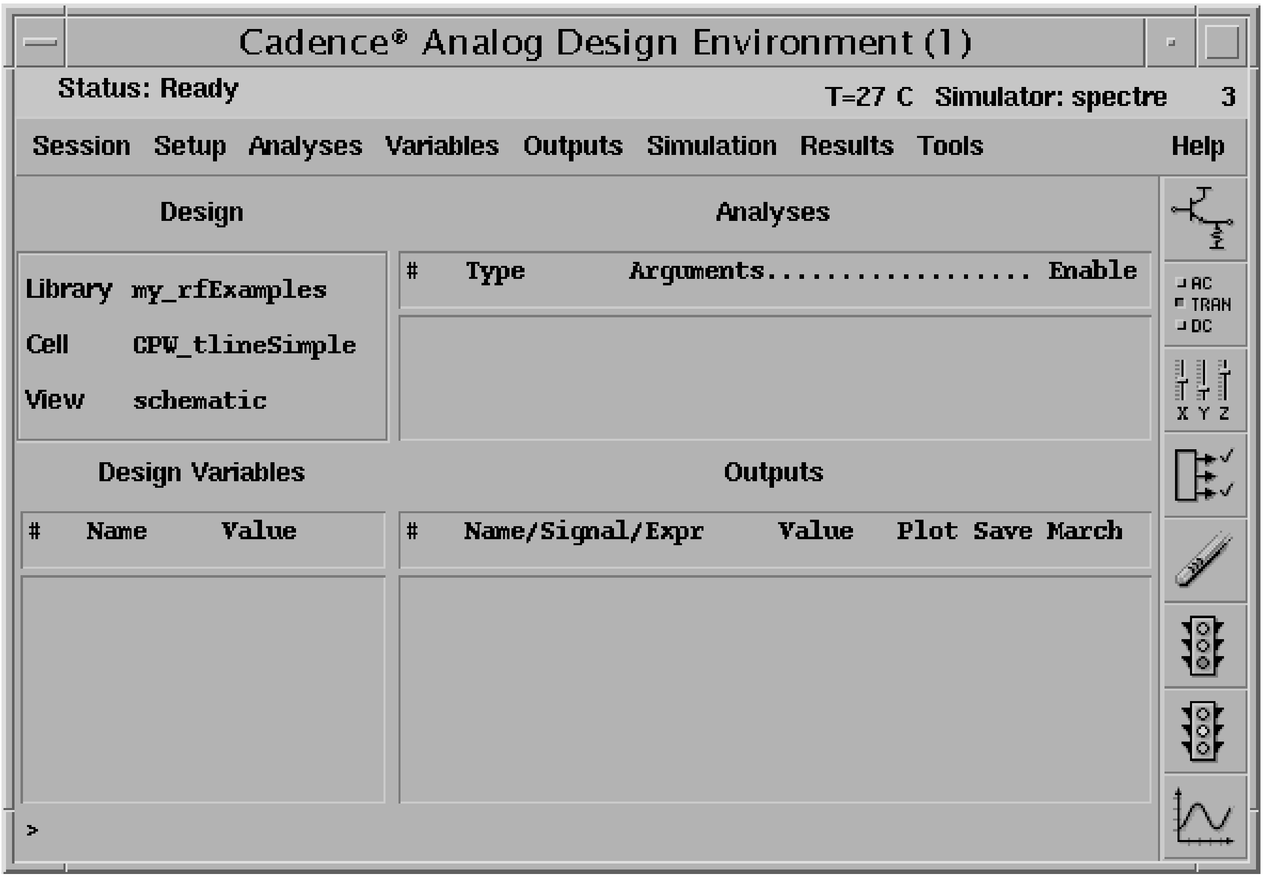
1. Open the `CPW_tlineSimple` circuit in the Schematic window as described in “Opening the CPW_tlineSimple Circuit in the Schematic Window” on page 559.

The Schematic window for the `CPW_tlineSimple` circuit appears.



2. In the Schematic window, choose *Tools—Analog Environment*.

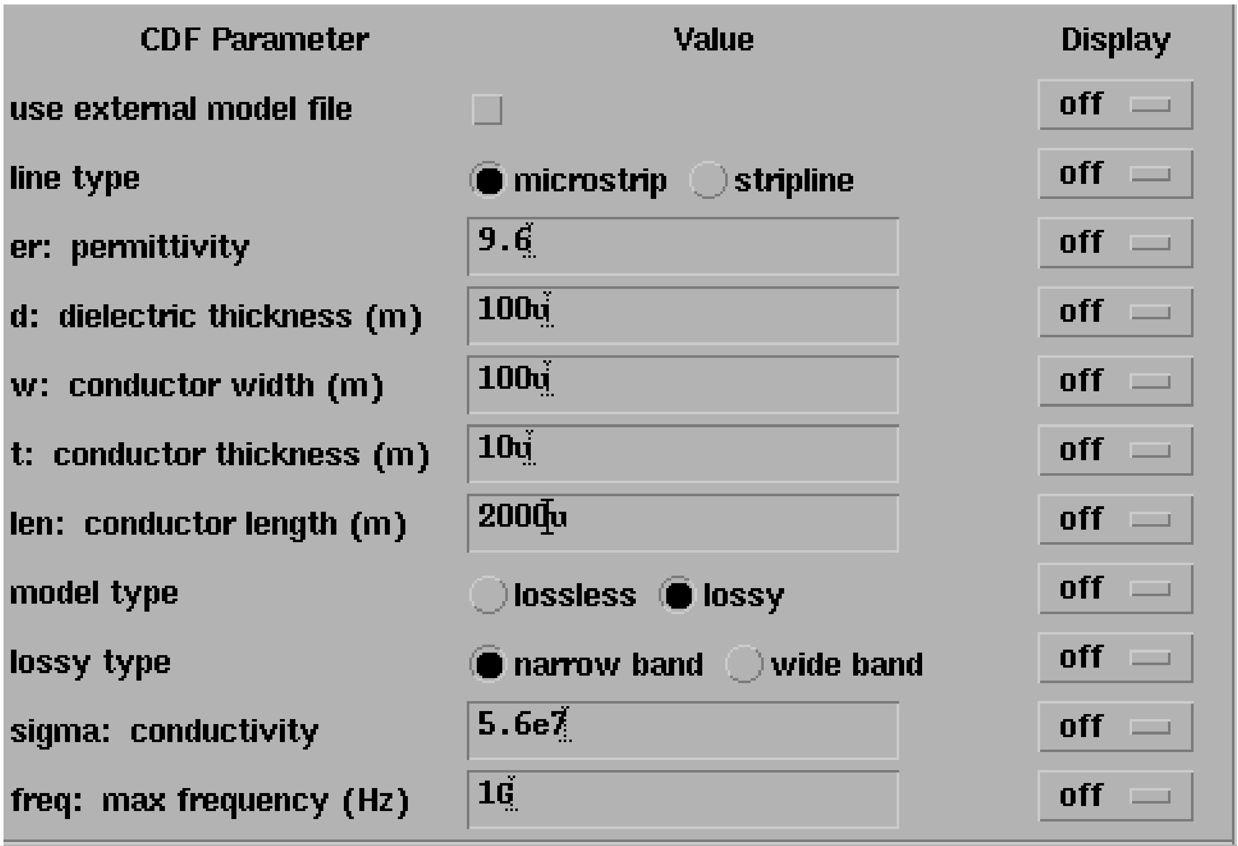
The Simulation window for the `CPW_tlineSimple` circuit opens.



3. In the `CPW_tlineSimple` schematic, verify that the length of the microstrip component is 2000 um, as described in, "Viewing the Properties of the Microstrip Component" on page 562. If necessary, edit the conductor length CDF parameter.

   `len: conductor length (m) 2000u`

The *CDF Parameter* section of the Edit Objects Properties form for the *microstrip* appears similar to the following.

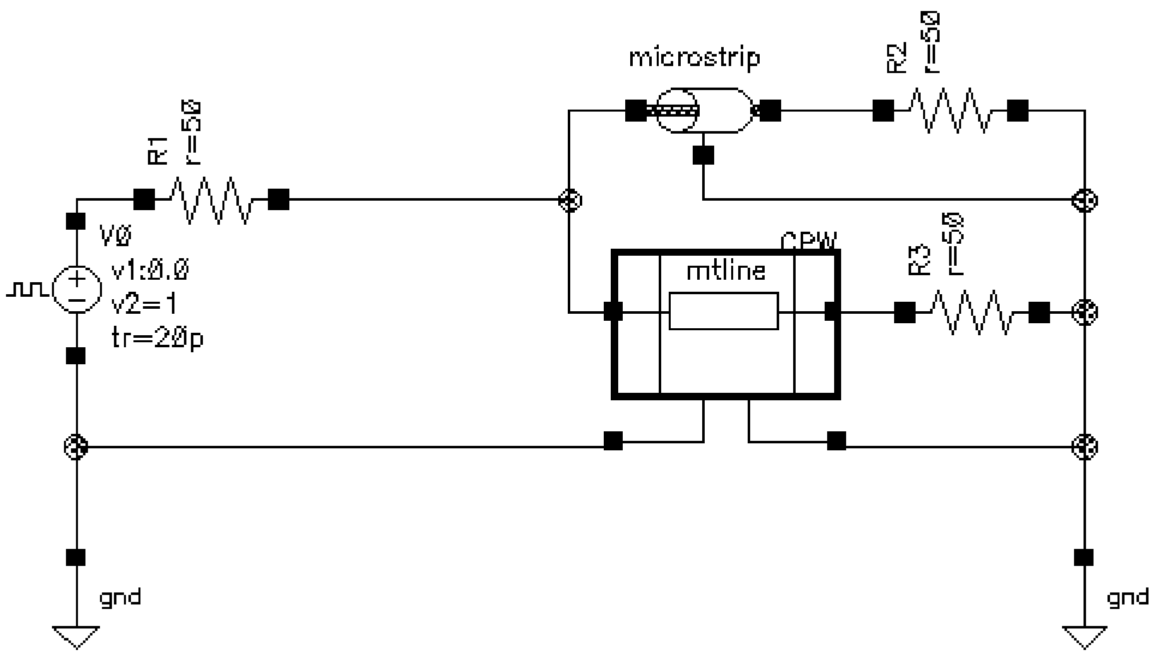


You can now set up and run a simulation in the analog design environment using the new `tline3` macromodel line length for the *microstrip* component.

Associating the Lumped Model File with the `Mtine` Component

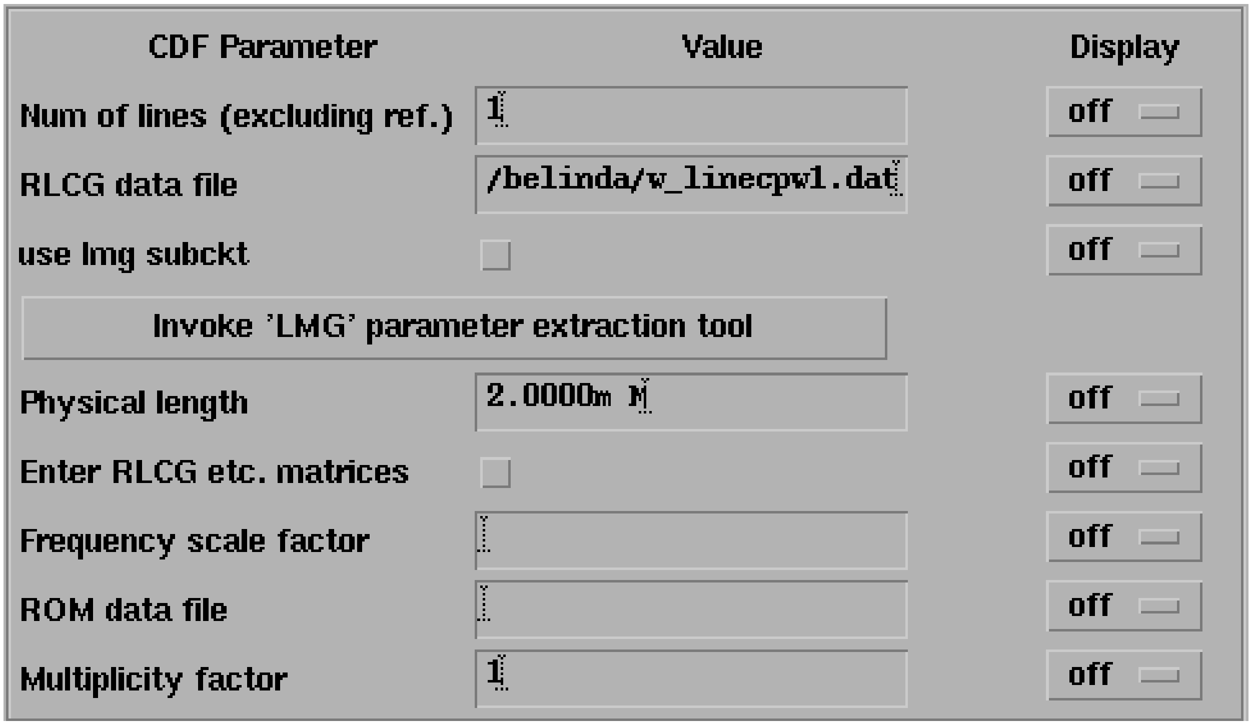
1. In the Schematic window, select the `mtline` transmission line component at the center of the schematic. This `mtline` component represents the *CPW*.

The `mtline` transmission line symbol is now selected.



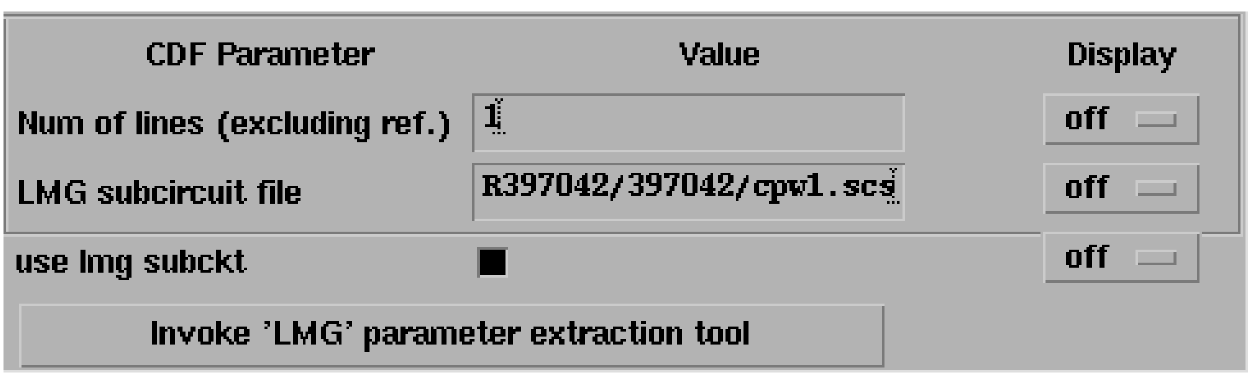
2. With the `mtline` symbol selected, choose *Edit—Properties—Objects* to open the Edit Object Properties form for the *CPW*.

The CFD parameter section of the Edit Object Properties form appears similar to the following.



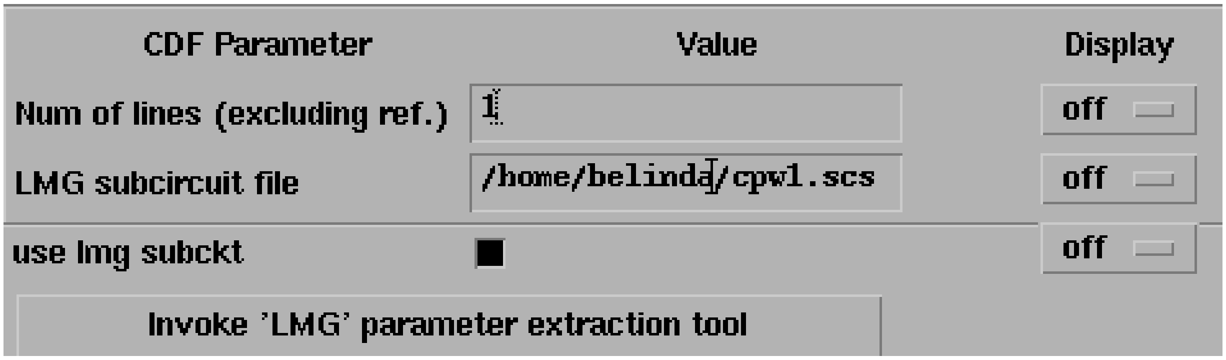
3. Select *use lmg subckt.*

The CFD parameter section changes to allow you to enter a lumped model file name.



4. In the *LMG subcircuit file* field type the full, absolute path to the lumped model file you generated in "Generating the Subcircuit and LRCG Files" on page 554.

For example, enter `/home/belinda/cpw1.scs.`



Note: If you receive an error message later when you simulate with this model, see "Resolving a Possible Error Message in the mtline Model File" on page 556 for information on correcting this error.

5. In the *Edit – Properties – Objects* form, click *OK*.

   When you click *OK*, the form closes and the `cpw1.scs` file is saved in the schematic editor. When you simulate with the *mtline* component, the `cpw1.scs` lumped model file is used.

6. In the schematic window, choose *Design—Check and Save* check and save for the `CPW_tlineSimple` schematic.

You can now set up and run a simulation using the new `mtline` macromodel in the analog design environment.

Simulating the CPW

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click on *tran*.

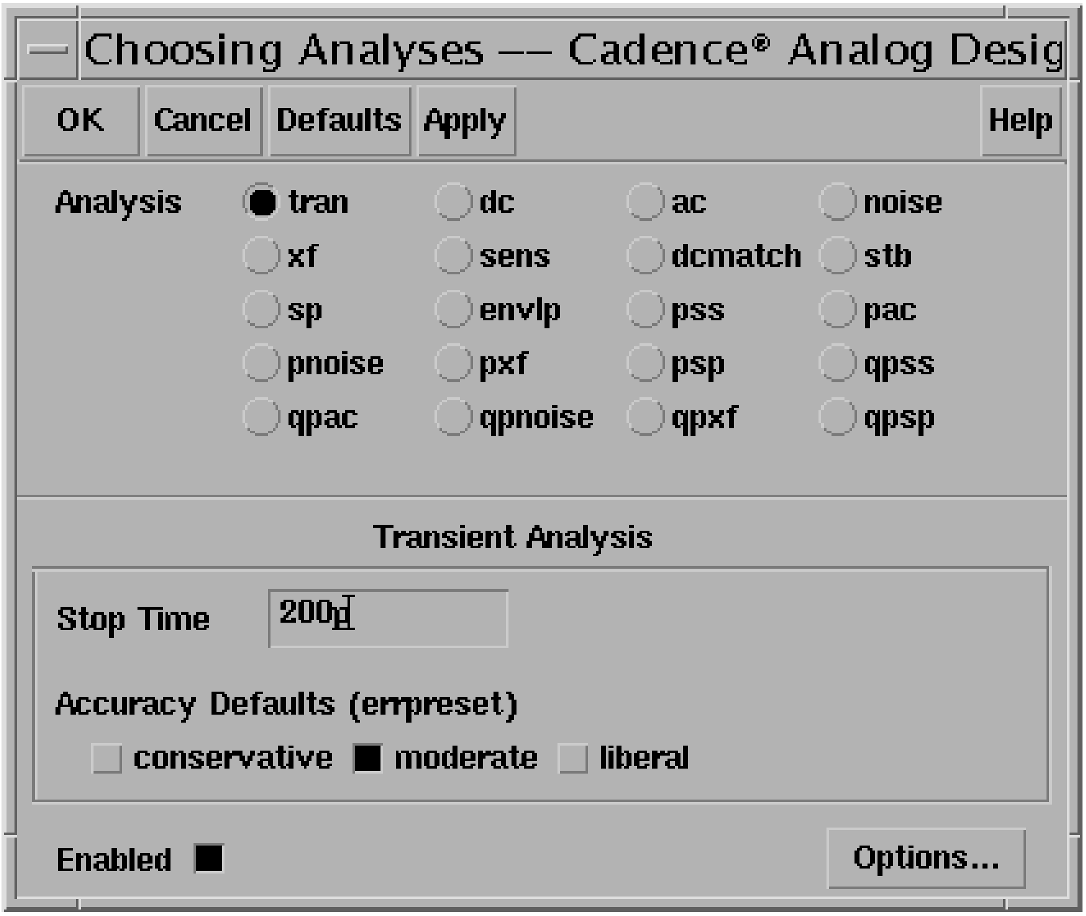
   The form displays options needed for `tran` analysis.

3. In the *Stop Time* field, type `200p`.

4. Highlight *moderate* for the *Accuracy Defaults (errpreset)*.

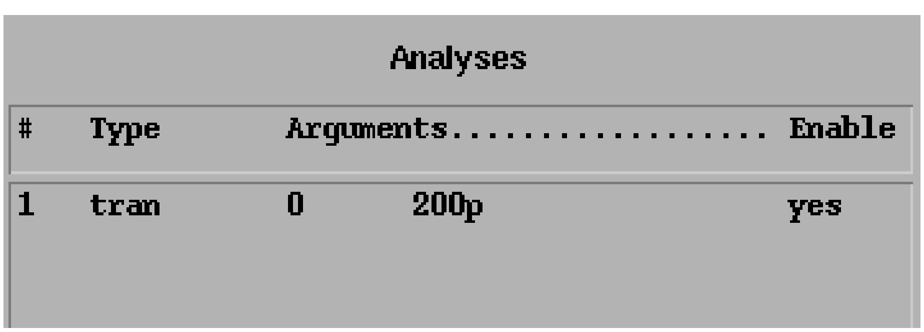
5. Verify that *Enabled* is highlighted.

The Choosing Analyses form appears below.



6. Click *Apply* to check for setup errors for the Transient analysis.
7. Click *OK*.

   The `Transient` analysis options you choose appear in the *Analyses* list box in the Simulation window.



Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Coplanar Waveguide Signals

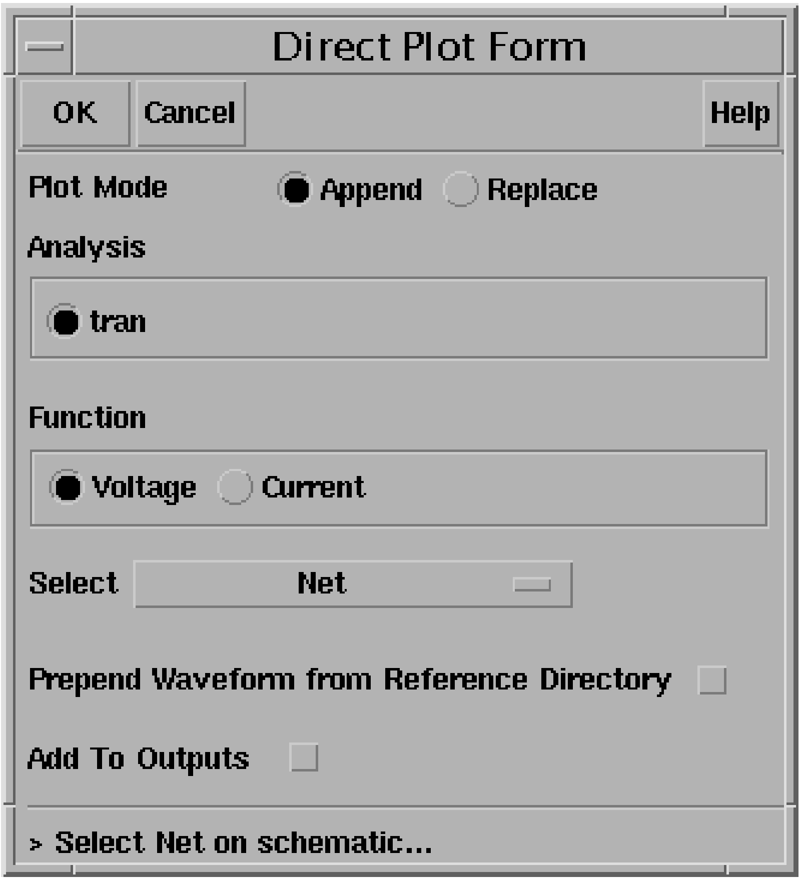
➤ To open the Direct Plot form, choose *Results – Direct Plot – Direct Plot* in the Simulation window.

   The Waveform window and the Direct Plot form appear.

In the Direct Plot form do the following.

1. Highlight *Append* for *Plot Mode*.
2. Highlight *Tran* for *Analysis*.
3. Highlight *Voltage* for *Function*.

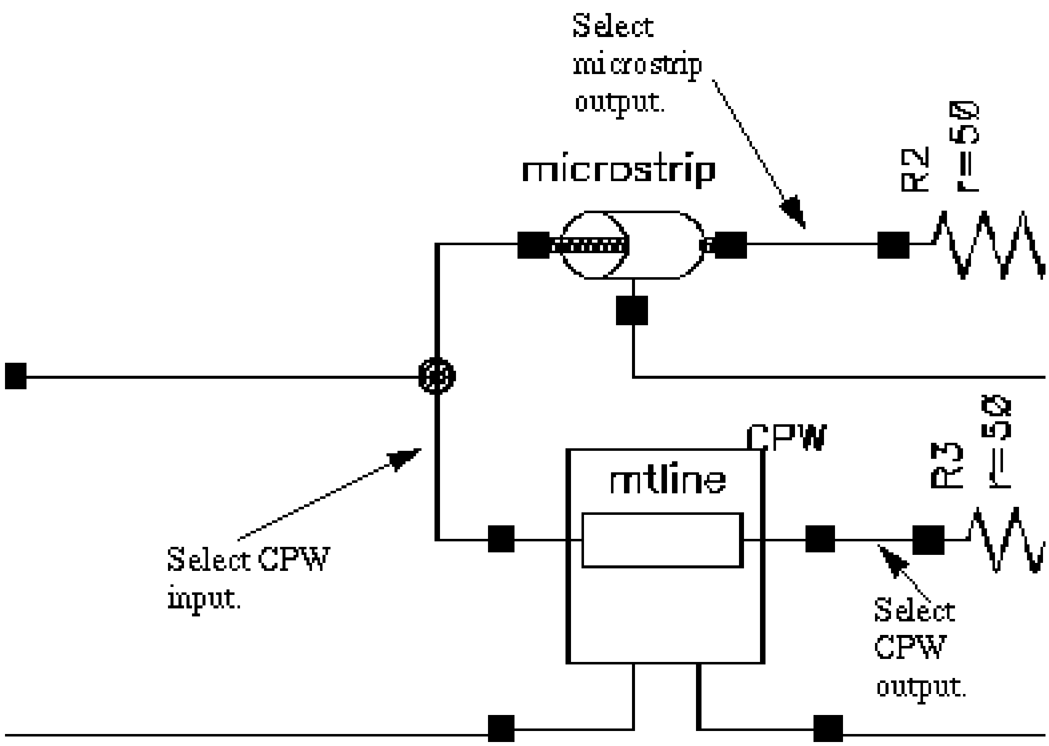
The filled-in form looks like this.



4. Follow the prompt at the bottom of the Direct Plot form.

```
> Select Net on Schematic...
```

Select the following nets in the CPW_tlineSimple schematic.



In the Schematic window, select the following nets.

a. Click on the CPW input.

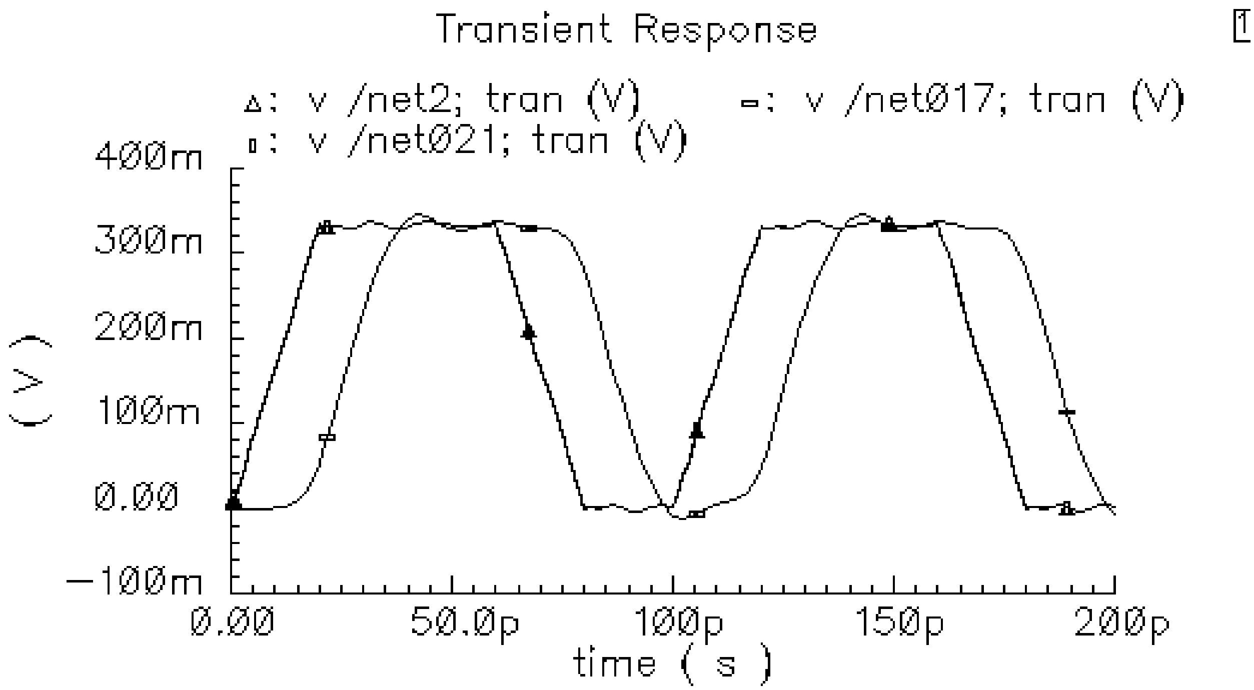
b. Click on the microstrip output.

c. Click on the CPW output.

d. Press Esc to stop selecting nets.

The plot in Figure 7-5 appears in the Waveform window.

Figure 7-9 Signals Passing Through the CPW and the Microstrip



Comparing Figure 7-9 to Figure 7-6, you can see that the corruption of the input signal is mainly due to the reflection come from *microstrip*. Because the length of CPW is now very short, 20 um, the output signal from the CPW is almost overlapping its input signal. From the Figure 7-9 you can also see that the lumped CPW model LMG has created is very accurate.

Internal Techniques in LMG and Model Accuracy

In the current release of LMG, both the LMG GUI and it's engine have been rewritten. Both the boundary element method (BEM) and the boundary spectral method (BSM) provide efficient parameter extraction of per-unit length LRCG matrixes for multi-layer multi-conductor transmission lines. Special discretization and basis functions efficiently capture both the skin effect and the proximity effect. The substrate coupling effects that impact on the CG matrixes are extracted by enforcing continuity conditions across different dielectric interfaces and the voltage condition on conductor surfaces. After frequency-dependent LRCG matrixes are extracted, a robust rational function fitting algorithm generates lumped interconnect models via Gaussian quadrature.

For the frequency-dependent per-unit length LRCG matrixes, we have made many comparisons between the current LMG tool and other measurements, other commercial tools and other publications. On both IC and PCB levels, the parameter extraction of the current LMG is efficient and accurate.

The LMG Gaussian quadrature lumped model is a simple yet accurate model. But since we also provide the *mtline* transmission line model and *mtline* focuses on simulation, we suggest that for long transmission lines, you use LMG to do parameter extraction and to provide the LRCG file to *mtline* to do the simulation. The current LMG combined with the *mtline* transmission line model provides a complete solution for transmission line modeling.

The LMG GUI Reference

In General, LMG does the following

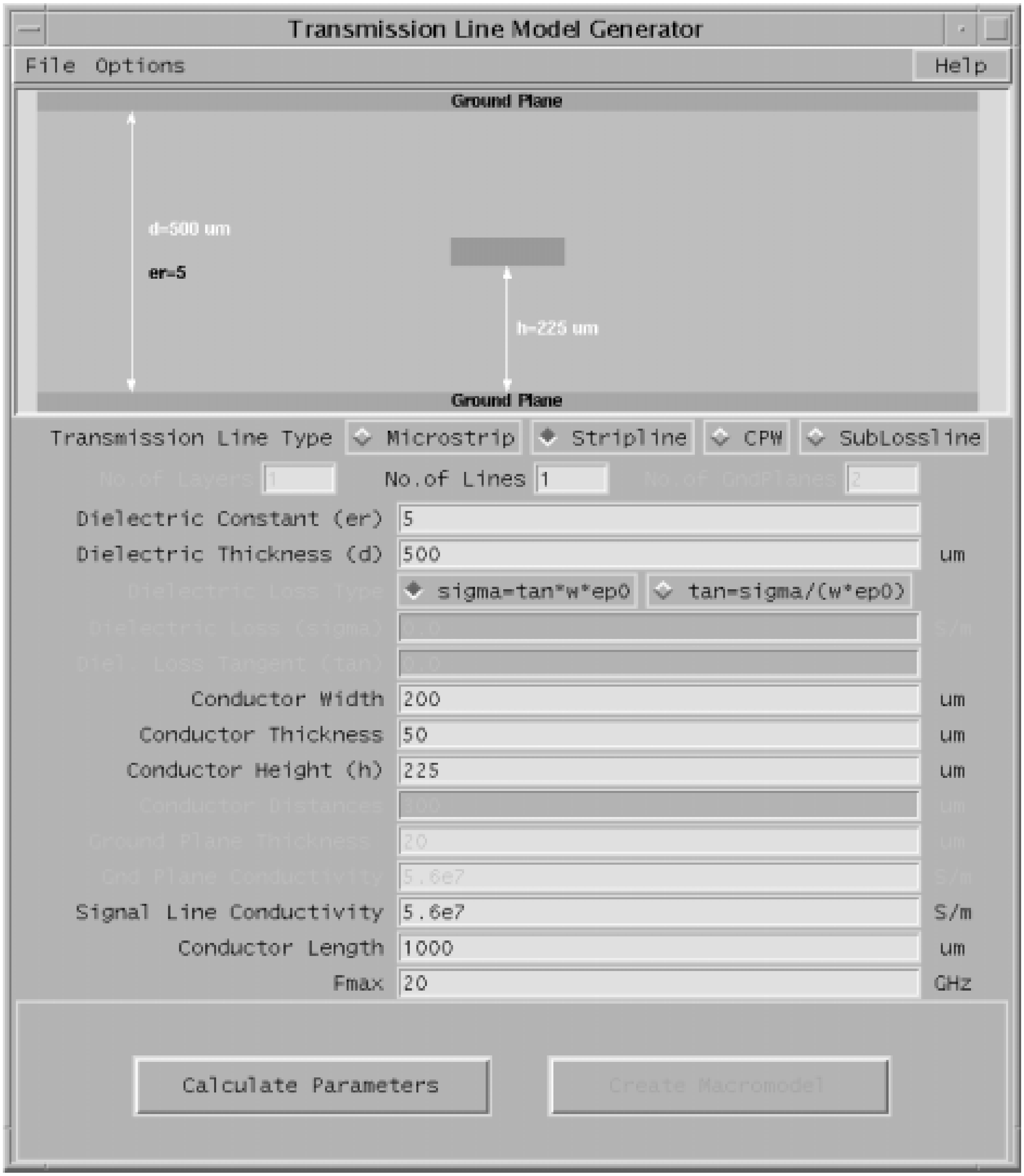
- LMG models 4 transmission line types:
  - Microstrip
  - Stripline
  - CPW (The coplanar wave guide line is new in 5.0.0.)
  - SubLossline (Includes stripline, microstrip line, lossy multilayer and multiconductor lines.)
- LMG generates 3 model types:
  - Lossless (All metal layers are PEC with no internal L.)
  - Lossy Narrow Band (Model operates at 1 frequency set with Fmax.)
  - Lossy Wide Band (Operates from DC to the frequency set with Fmax.)
- You can invoke LMG in 4 ways:
  - As a standalone GUI, type `lmg` at the operating system prompt.
  - In the analog design environment schematic flow, edit the mtline model's object properties during simulation setup.
  - From the RF Submenu, open the LMG GUI.
  - From the command line without the LMG GUI, type `lmg - help` at the operating system prompt to display the LMG command and options.

LMG extracts parameters, produces per-unit-length LRCG matrixes, and generates models and subcircuits. You can input existing, frequency-dependent LRCG matrixes directly into the mtline model by editing the model's object properties during simulation setup.

New in 5.0.0, LMG models transmission lines without ground planes. (Input 0 in the LMG GUI's "No. of GndPlanes" field.) LMG uses the last line as the reference return for both current and charge conservation. LMG performs size reduction for the lumped model and RLCG file.

The Transmission Line Generator (LMG) GUI is described in the following sections.

Figure 7-10 LMG GUI



The LMG GUI, is divided into four sections.

- Menus
- Display Section
- Data Entry Section
- Function Buttons

Menus

The two pull-down menus at the top of the GUI.

Use the two menus, *File* and *Option*, to set up LMG and to begin creating and saving models.

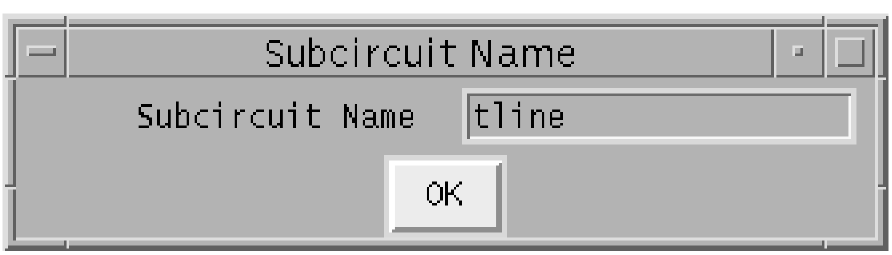
File Menu

The *File* menu has the following four options:

- *Subcircuit Name*
- *LRCG File Name*
- *Save Current Settings*
- *Quit*

Subcircuit Name

Choose *File—Subcircuit Name* to bring up the Subcircuit Name form in which you type the name of the output subcircuit. In this case, the default subcircuit name is `tline`. LMG creates the output subcircuit file name by appending `.scs` to the name you enter.
.



Enter a name that describes the model you are creating.

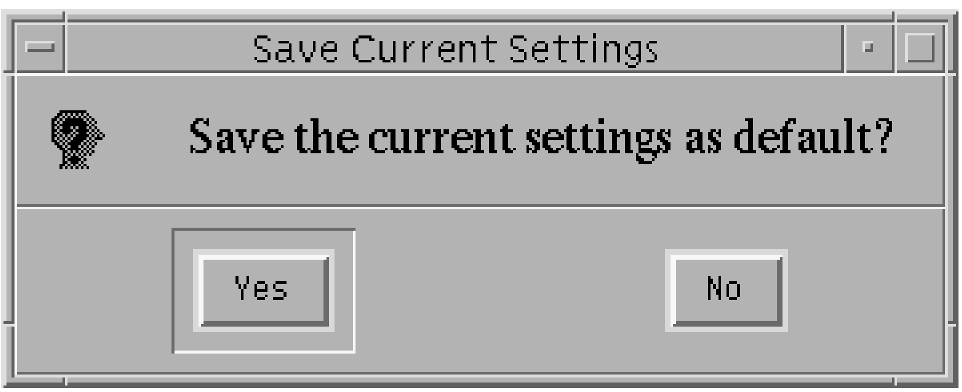
LRCG Filet Name

Choose *File—LRCG File Name* to bring up the LRCG File Name form in which you type the name of the output LRGC file name where the frequency dependent per-unit length LRCG matrixes are stored in the format consistent with the mtline macromodel. In the case illustrated here, the default file name is `w_line1.dat`.

.

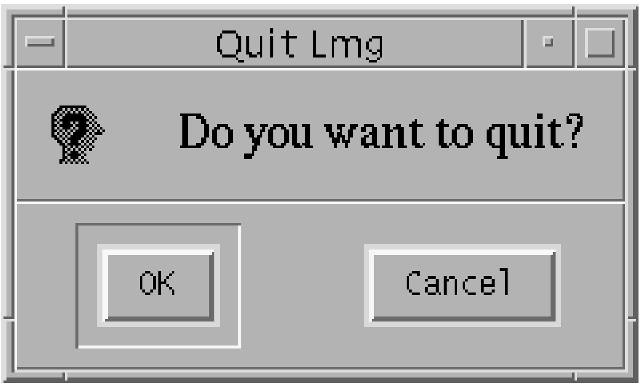
Save Current Settings

Saves the current transmission line information in the file ./`.initlmg` in the current directory as a defaults for the next LMG session if you click *Yes*.



Quit

Exits `lmg` if you click *OK*.



Options Menu

The *Options* menu has five main options.

- *Model Type*
- *Subckt Format*
- *Length Format*
- *Freq. Unit*
- *Output file Control*

Each option has a submenu in which you choose a value. After you choose a value, the geometry of the diagram is updated automatically to implement your choice.

Model Type

The three Model Type options you choose from are

- *Lossy, Wide Band*

  Parameter extraction includes metal loss from both signal lines and ground planes. The frequency range is from DC to Fmax.

- *Lossy, Narrow Band*

  Parameter extraction includes metal loss from both signal lines and ground planes. The working frequency is set to one frequency: Fmax.

- *Lossless*

  Parameter extraction does not include the metal loss from signal lines and ground planes.This means that all the metals are regarded as perfect electrical conductors (PEC), and, at this point, Fmax is meaningless, for the code is internally setup to work at the frequency of 50 GHz to force the current to flow on the surface of the metals.

Subckt Format

Select the simulator to use one of two subcircuit formats.

- *Spectre*
- *SpectreS*

Length Unit

Selects the length unit to use.

- *um*
- *mm*
- *m*
- *mil*

The currently selected length unit is displayed to the right of the data entry fields in the LMG GUI. All the dimension quantities use the same unit for consistency.

Freq. Unit

Selects the frequency unit to use.

- *Hz*
- *MHz*
- *GHz*

The currently selected frequency unit is displayed to the right of the data entry fields in the LMG GUI.

Output file Control

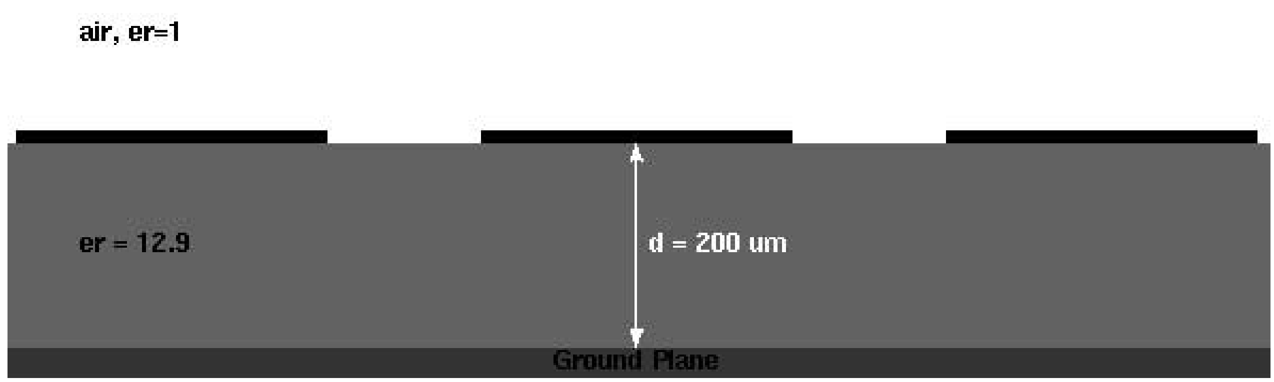
Selects which output files to create.

- *RLCG file*
- *Lumped Model*
- *Both*

The selected files are created when you press the *Create Macromodel* button.

Display Section

Displays the two dimensional cross section of the transmission line. Inside the cross section also displays the dielectric thickness, the dielectric constant, and the conductor height.The display is to scale and applies the same scale for both the horizontal and vertical dimensions.



Any time you specify new parameter information in the data entry section, the display changes immediately to reflect your change.

Data Entry Section

In the data entry section, you type information describing the transmission line you are modeling. If a field has a label, you can leave the default value unmodified, but you cannot leave a field blank.

Do not put values in the fields that are grayed out. Grayed out fields represent quantities that do not accept data for the model you are using. The meanings of the data entries are explained in the following sections.

Some data entry fields accept more than one data entry. To enter multiple data items, separate them with a single space.

Important

When you enter data incorrectly, the field is flooded in red until you correct your mistake.

Transmission Line Type: Microstrip, Stripline, CPW, SubLossline

The 4 radio buttons at the top of the data entry section let you choose between the *Microstrip*, *Stripline*, *CPW* or *SubLossline* transmission line types.

CPW models a coplanar waveguide.

SubLossline includes transmission line systems with multi-layer multi-conductors. It also includes cases with or without substrate loss. In fact, *SubLossline* includes both *Microstrip* and *Stripline.* For reasons of speed we keep *Microstrip* and *Stripline* as transmission line types.

Number of Layers

Type an integer number of layers for the transmission line up to 12.

Number of Lines

Type an integer number of conductors for the transmission line up to 30.

Number of GndPlanes

Type an integer number of ground planes for the transmission line up to two.

Dielectric Constant (er)

For *Microstrip* transmission lines, *er* is the relative dielectric permittivity of the substrate with air above the substrate.

For *Stripline* transmission lines, *er* is the relative permittivity of the dielectric between the upper and lower ground planes.

For *SubLossline* transmission lines, *er* is accounted for from bottom to top, each layer's *er* is separated by blank space. If the number of *er* values is less than the number of layers, then the last *er* value is automatically repeated.

Dielectric Thickness (d)

For *Microstrip* transmission lines, *d* is the thickness of the substrate.

For *Stripline* transmission lines, *d* is the distance between the upper and lower ground planes.

For *SubLossline* transmission lines, *d* is accounted for from bottom to top, each layer's thickness *d* data is separated by blank space. If the number *d* is less than the number of layers, the last *d* is repeated.

Dielectric Loss Type: (sigma=tan∗w∗ep0) or (tan=sigma/(w∗ep0)

The 2 radio buttons allow you to select the appropriate Dielectric loss type *sigma (for IC users)* or *loss tangent (for PCB users).*

- *sigma=tan*∗w∗*ep0* (the *Dielectric Loss (sigma)* field is activated)
- *tan=sigma/(w∗ep0)* (the *Dielectric Loss Tangent* field is activated)

For a wide band model,

- when you select tangent, tangent is constant over the frequency range.
- when you select sigma, sigma is constant over the frequency range.

Note: When you activate one field, for example, *Dielectric Loss (sigma)*, you deactivate the other field, *Dielectric Loss Tangent*.

Dielectric Loss Sigma

For the *SubLossline* transmission line, each dielectric can have substrate loss (for sigma >= $1.0e^{-6}$ S/m) or not have substrate loss (for sigma <$1.0e^{-6}$ S/m). We ignored sigma < $1.0e^{-6}$ and treat it as 0.

Conductor Width

Specifies the width of each conductor. The conductors are accounted for from left to right. Separate each conductor line width with a blank space. If the number of conductor widths is less than the number of conductor lines, then the last width is repeated.

Conductor Thickness

Specifies the thickness of each conductor. The conductors are accounted for from left to right. Each line's thickness is separated by blank space. If the number of thickness values is less than the number of conductor lines, then the last thickness is repeated.

Conductor Height (h)

Conductor Height is for the *Stripline* and *SubLossline* transmission lines only. It specifies the distance between the lower face of the conductor wire and the upper face of the bottom ground plane.

Conductor Distances

Either

- There are multiple conductors (You entered a number greater than 1 in the *No. of Lines* field), and the *Conductor Distances* field is active.
- There is a BoardSide coupled transmission line
  - For two conductors, enter the distance between the conductors.
  - For three or more conductors, enter the distances between the conductors separated by a space.
  - For a boardside coupled transmission line, enter a negative number. Conductor Distance is the only entry in the LMG GUI that you can input a negative number.

Ground Plane Thickness

Specifies ground plane thickness. You can specify one or two ground plane thicknesses. Two ground planes can have different thicknesses. The first is for the bottom ground plane, the second is for the top ground plane. They are separated by a space.

Gnd Plane Conductivity

Specifies ground plane conductivity. If you do not want extracted parameters including ground plane loss effects, input a very larger number, say the ground plane conductivity is $1.0e^{14}$.

Signal Line Conductivity

Specifies the conductivity of the conductor wire. Relative to the per-unit length R and L matrixes.

Conductor Length

Specifies the length of the transmission line in the unit shown to the right of the data field.

Fmax

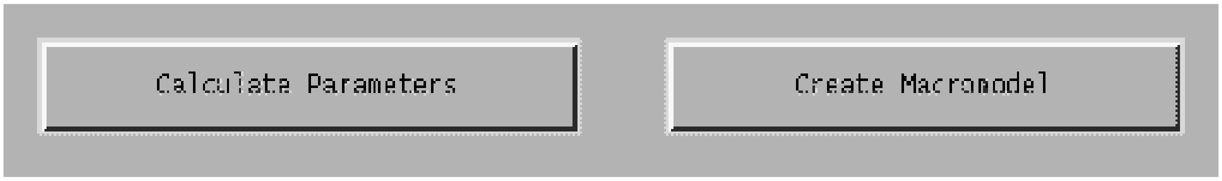
The maximum frequency at which the macromodel must work properly. You usually set *Fmax* to the frequency of the highest harmonic that has significant energy in the transmission line simulations. LMG generates a macromodel of the transmission line that is good from DC to the value chosen for *Fmax if* the model type is chosen as *Lossy, Wide Band*. If the model type is chosen as *Lossy, Narrow Band*, then LMG just do parameter extraction and model generation at frequency defined by *Fmax*. However, if model type is chosen as *Lossless*, then *Fmax* will be ignored for the tool will set a very high working frequency internally to avoid the internal inductance. Looking at the LRCG file, you can find what is the real frequency you used.

Function Buttons

The function buttons

- Update the figure display
- Calculate the transmission line parameters
- Create the macromodels.

There are two function buttons at the bottom of the Transmission Line Modeler GUI.



Calculate Parameters

After you specify the 2D transmission line geometry, click on this button to calculate the transmission line parameters. The display shows the characteristic impedance, propagation velocity, and the DC resistance per meter of the transmission line in the upper right corner of the form if applicable.

Create Macromodel

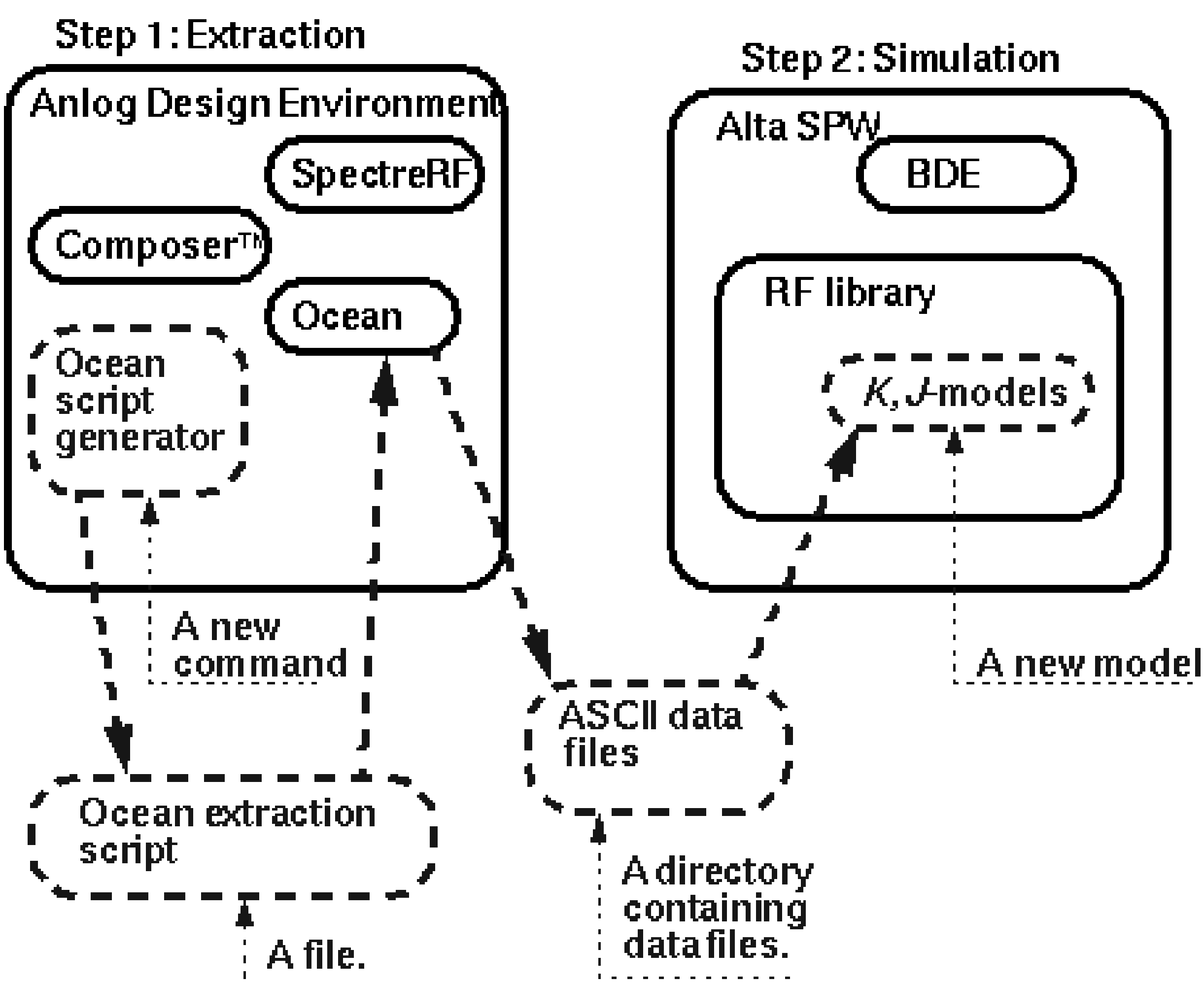
After you click on the *Calculate Parameters* button, this button is enabled. Click on this button to create the transmission *mline* model. A pop-up window tells you the name of the transmission line model file.

8

Creating and Using Receiver K-Models

The receiver K-model links SpectreRF and Alta™ SPW as shown in Figure 8-1.

Figure 8-1 Task Flow of K-Model Creation and Use



The Ocean extraction script automatically runs a set of SpectreRF analyses and writes the results to a directory you specify. The results characterize the baseband performance of the RF circuit.

There are three kinds of K-models.

- **Transmitter:** This is actually a J-model and is discussed in chapter 10.
- **Receiver (linear):** This is a linearized model of the receiver. It includes noise except for phase noise. Extraction is very similar to extraction of the non-linear K-model.
- **Receiver (non-linear):** This is a noiseless non-linear model of the receiver.

The receiver K-models only exist in the Alta SPW RF library but you extract the K-model data using SpectreRF in the analog design environment.

This chapter describes the following:

- The procedure for extracting a non-linear receiver K-model.
- The K-model’s limitations
- The procedure for handling strong out-of-band blockers

The K-model data files and their format are described at the end of the chapter.

Procedures for Simulating k_mod_extraction_example

Follow the steps below to simulate the *k_mod_extraction_example*:

➤ Copy *k_mod_extraction_example* from the Cadence *rfExamples* directory into a local directory.

Be sure the path to the local directory is defined in your *cds.lib* file.

➤ Choose *File – Open* in the Command Interpreter Window (CIW).

The Open File form appears. Filling in the Open File form lets you open the schematic.

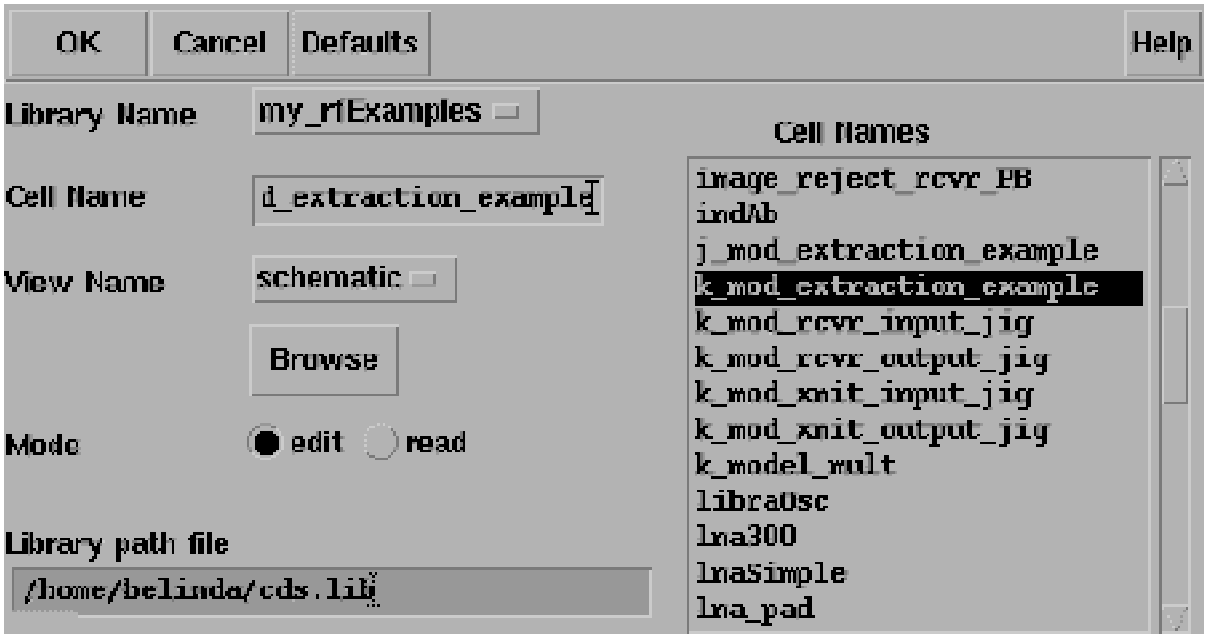
In the Open File form, do the following:

1. Choose <Path_to_local_copy> for Library Name.
2. In the *Cell Names* list box, highlight *k_mod_extraction_example*.

   k_mod_extraction_example appears in the *Cell Name* field.
3. Choose *schematic* for *View Name*.

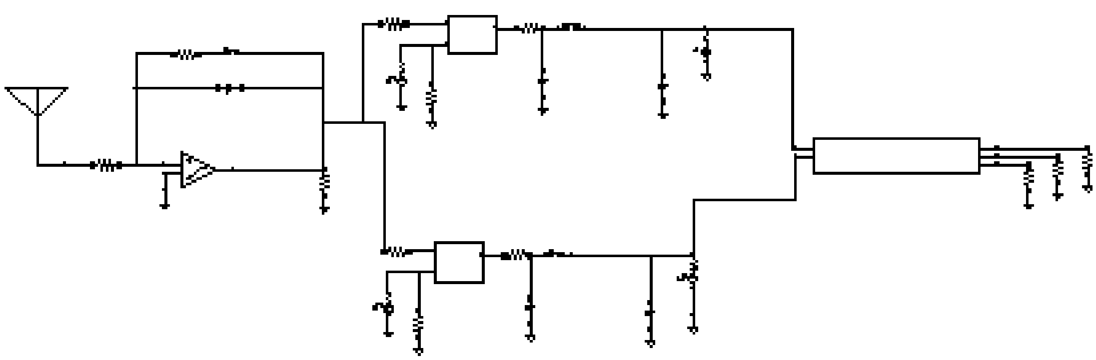
4. Highlight *edit* to select the *Mode*.

the Open File form appears as follows.



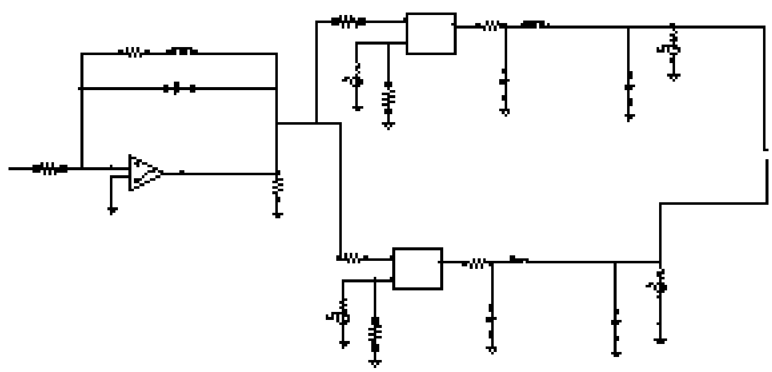
5. Click *OK* in the Open File form.

The Schematic window opens with the *k_mod_extraction_example* schematic.



Optional: Setting Up the Input and Output Jigs

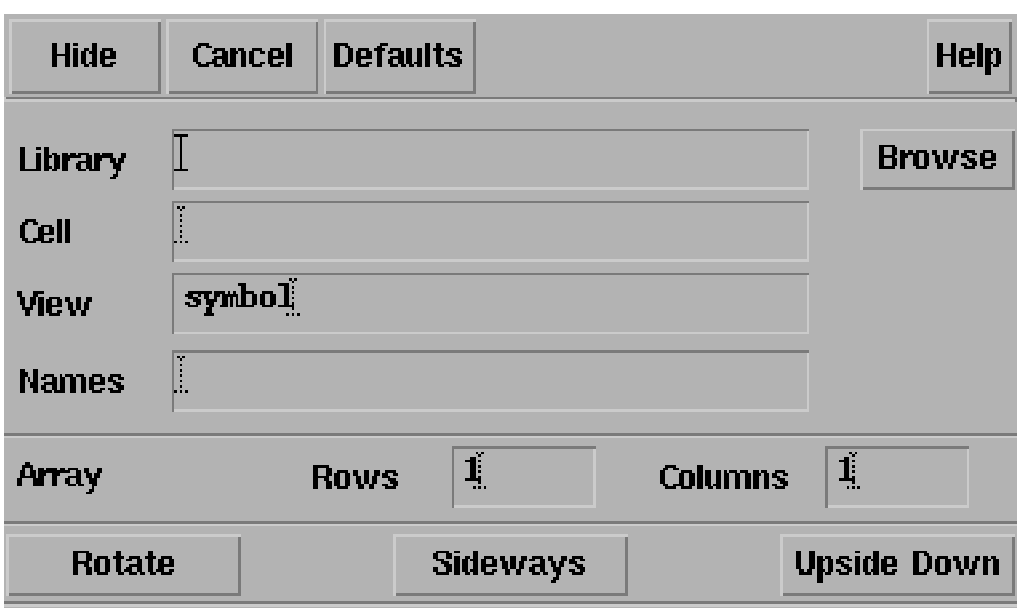
If you want to see the whole procedure for setting up the K-model, including the editing of the schematic, delete the input and output test jigs at the far left and right of the schematic. The edited schematic now looks like the one below.



From the Schematic window, do the following:

1. Select *Add – Instance.*

   The Add Instance form appears.



2. In the Add Instance form, click on *Browse*.

The Library Browser – Add Instance form appears. In the Library Browser – Add Instance form, do the following:

3. Select *rfExamples* for the *Library*.

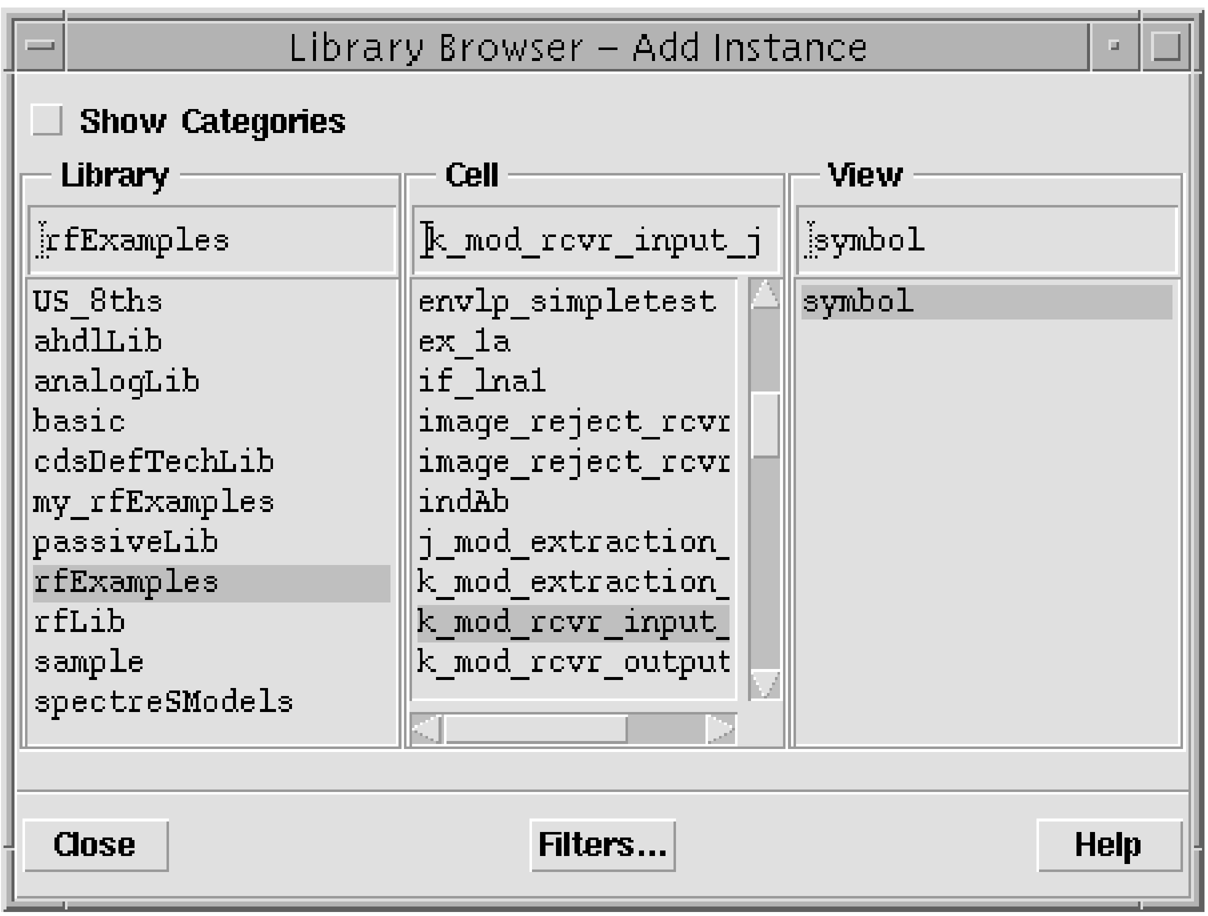
   The form changes to let you select a cell.

4. Select *k_mod_rcvr_input_jig* for the *Cell*.

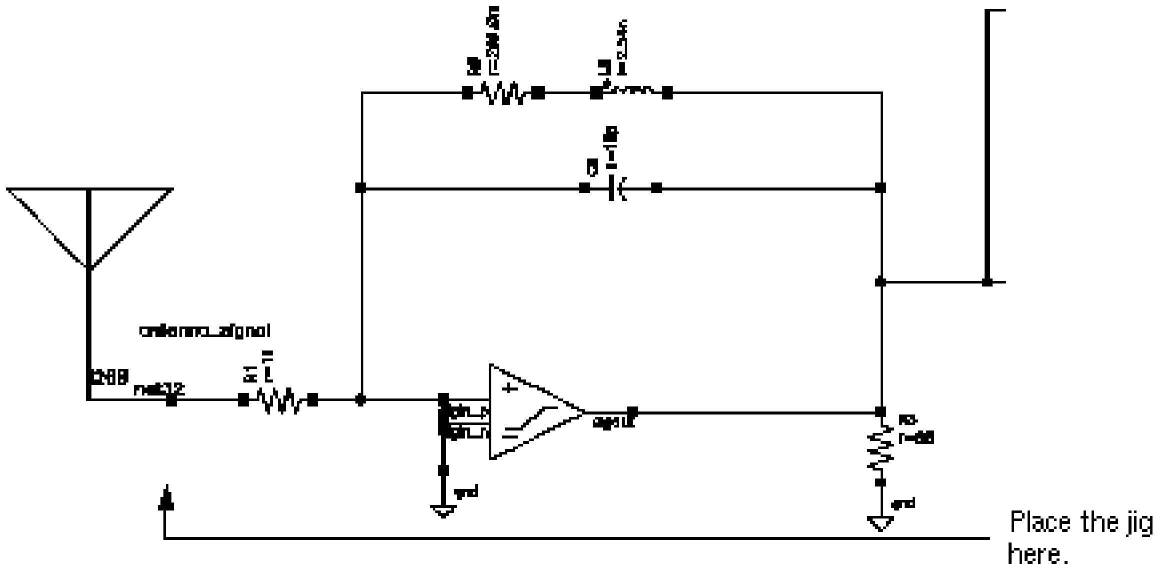
   The form changes to let you select a View.

5. Select *symbol* for the *View*.

   The completed form appears like the one below



When you place the cursor back in the Schematic window, an instance of *k_mod_rcvr_input_jig* is attached to it. Place the input jig at the appropriate place in the schematic.



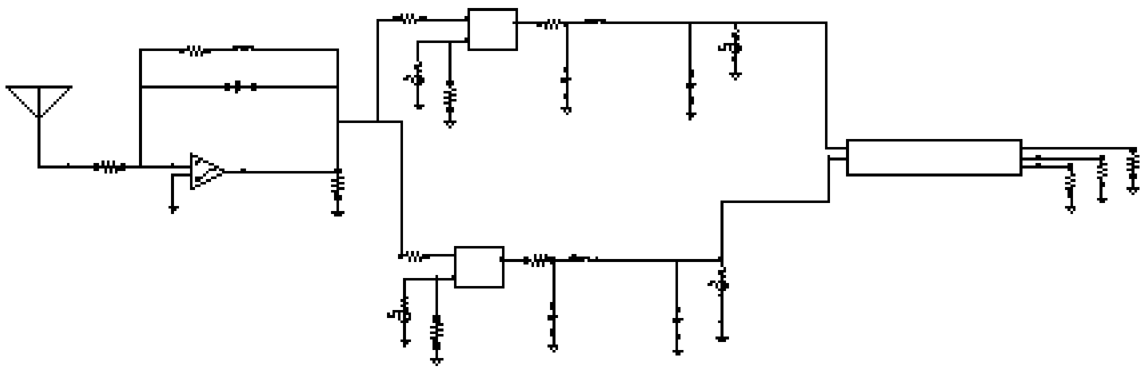
If you need assistance with the procedures for editing a schematic, see the Virtuoso® Schematic Composer™ User Guide.

In the Schematic window, again select *Add – Instance.*

Now repeat the steps you used to add the input jig to the schematic and add the output jig *k_mod_rcvr_output_jig.* Also add a resistor connected to ground to each of the three output connections of the output jig. The table below tells you the libraries, cells, and views of the components you add.

Component	**Library**	**Cell**	**View**
output jig	rfExamples	k_mod_rcvr_output_jig	symbol
resistor	analogLib	res	symbol
ground	analogLib	gnd	symbol

The edited schematic looks like this.



In the Schematic window, perform the following steps:

1. Click on the input jig to select it.
2. Select *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears.
3. In the Edit Object Properties form, type `1G` for the *carrier frequency* value.
4. In the Edit Object Properties form, click *OK*.

   The completed form appears like the one below:

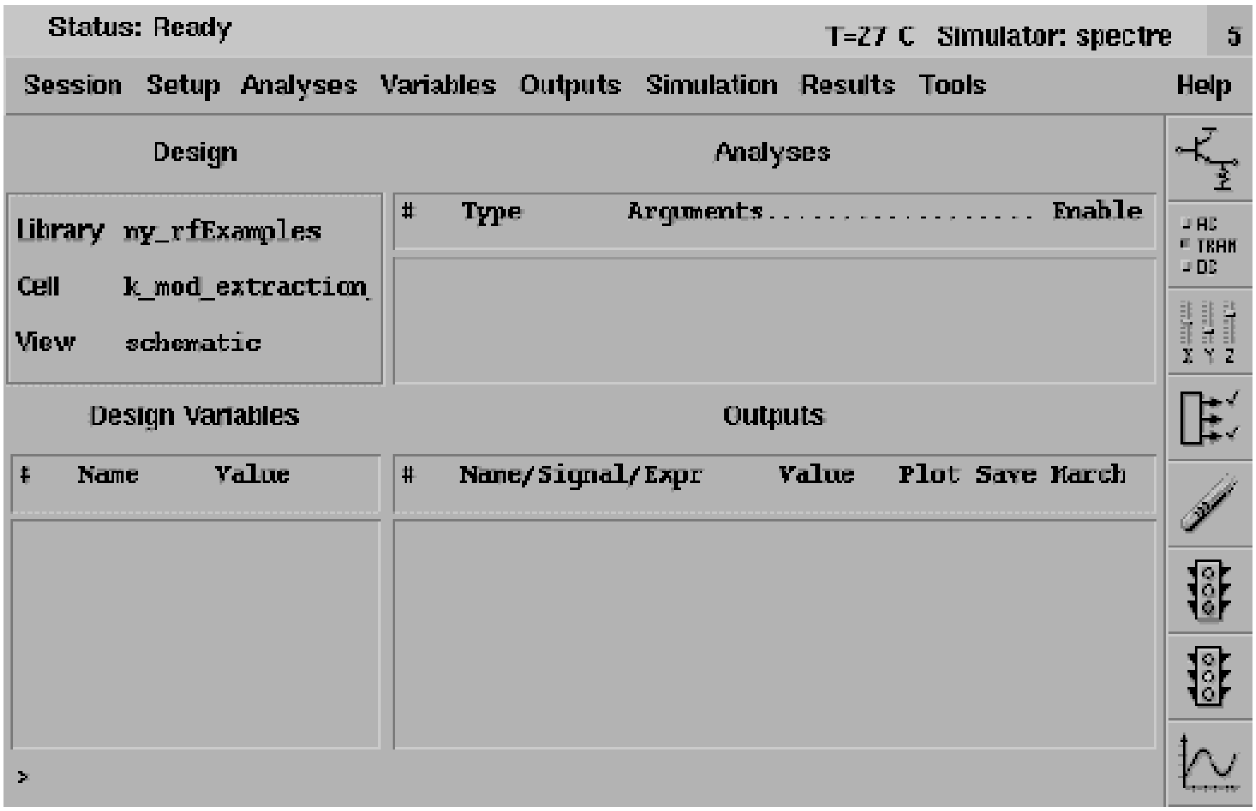
CDF Parameter	Value
carrier frequency	1G Hz
Kmodel: signal bias	sig_bias V
Kmodel: carrier phase	car_ph deg
Kmodel: signal bias end point	bias_end V
Carrier frequency name	car_freq

5. In the Schematic window, select *Design – Check and Save*.

Analysis Setup

➤ Choose *Tools – Analog Environment* in the Schematic window.

The Simulation window appears.



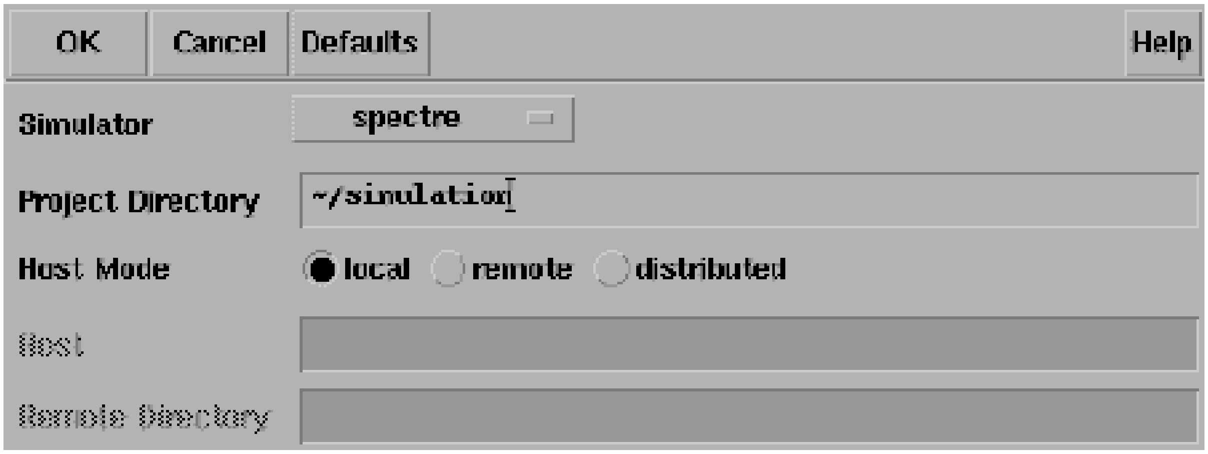
Setup the Simulator, Directory, and Host

1. Choose *Setup – Simulator/Directory/Host* in the Simulation window.

   The Choosing Simulator/Directory/Host form appears. In the Choosing Simulator/Directory/Host form, do the following:

2. Highlight *spectre* for *Simulator*.
3. Type in the name of the project directory, if necessary.
4. Highlight *local.* (If you highlight *remote Host Mode*, type in the name of the host machine and the remote directory in the appropriate type-in fields.)

The completed form appears like the one below.

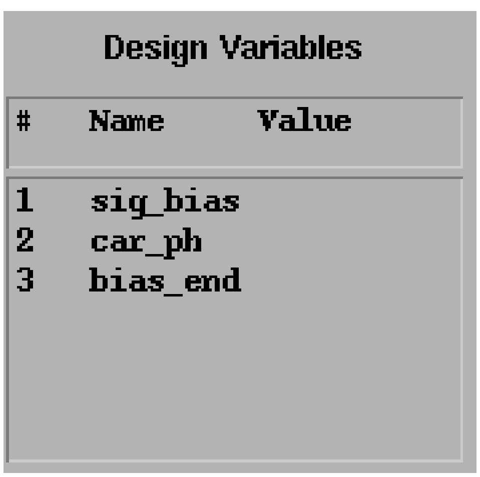


5. In the Choosing Simulator/Directory/Host form, click *OK*.

Copy and Edit Design Variables

1. In the Simulation window, choose *Variables – Copy From Cellview.*

The variables *sig_bias*, *car_ph*, and *bias_end* appear in the *Design Variables* list box of the Simulation window.

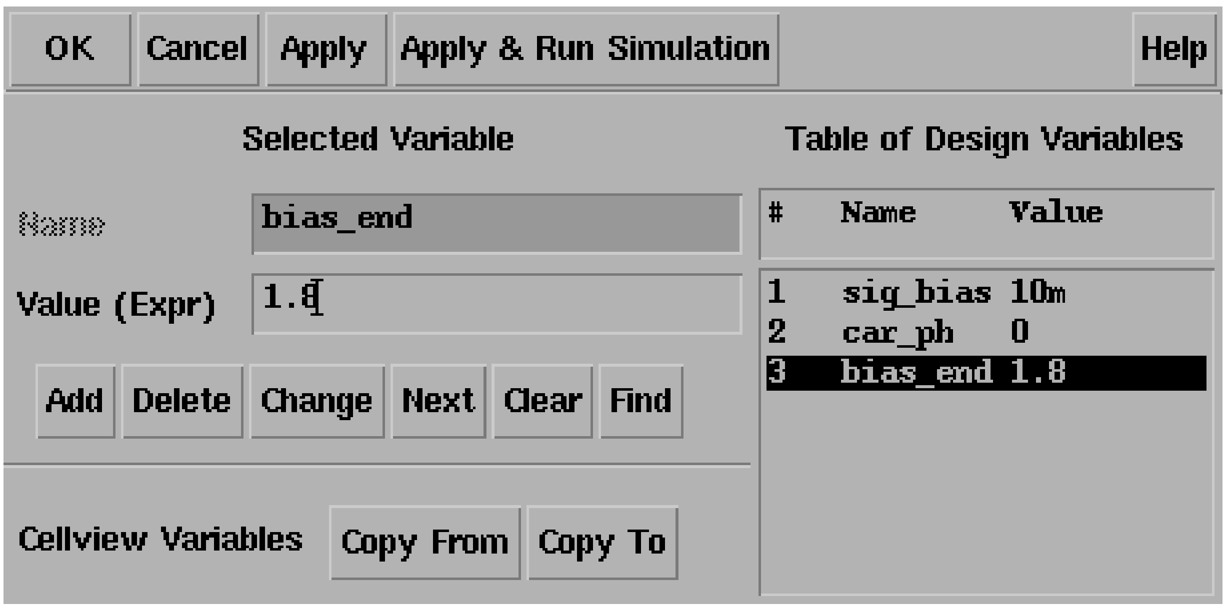


2. In the Simulation window, choose *Variables – Edit.*

The Editing Design Variables form appears. In the Editing Design Variables form, do the following:

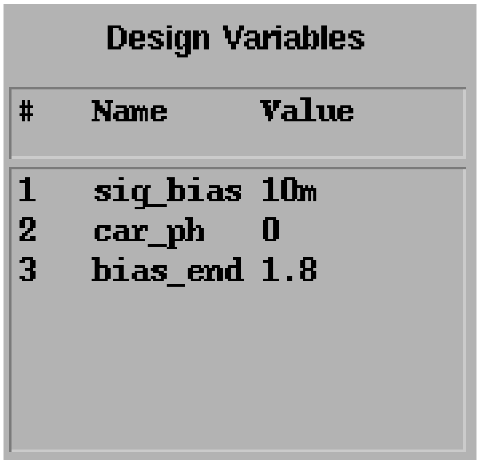
3. Click on *sig_bias* in the *Table of Design Variables* list box.
4. Type `.01` in the *Value(Expr)* field.
5. Click *Change*.
6. Repeat steps one through three for the *car_ph* and *bias_end* variables. Type `0` for *Value(Expr)* for *car_ph* and `1.8` for the *bias_end* variable.

   The completed form looks like the one below:



7. Click *OK*.

The new variable values display in the *Design Variables* list box in the Simulation window.



Setting Up the Analyses

1. Choose *Analyses – Choose* in the Simulation window.

   The Choosing Analyses form appears.

Setting Up the PSS Analysis

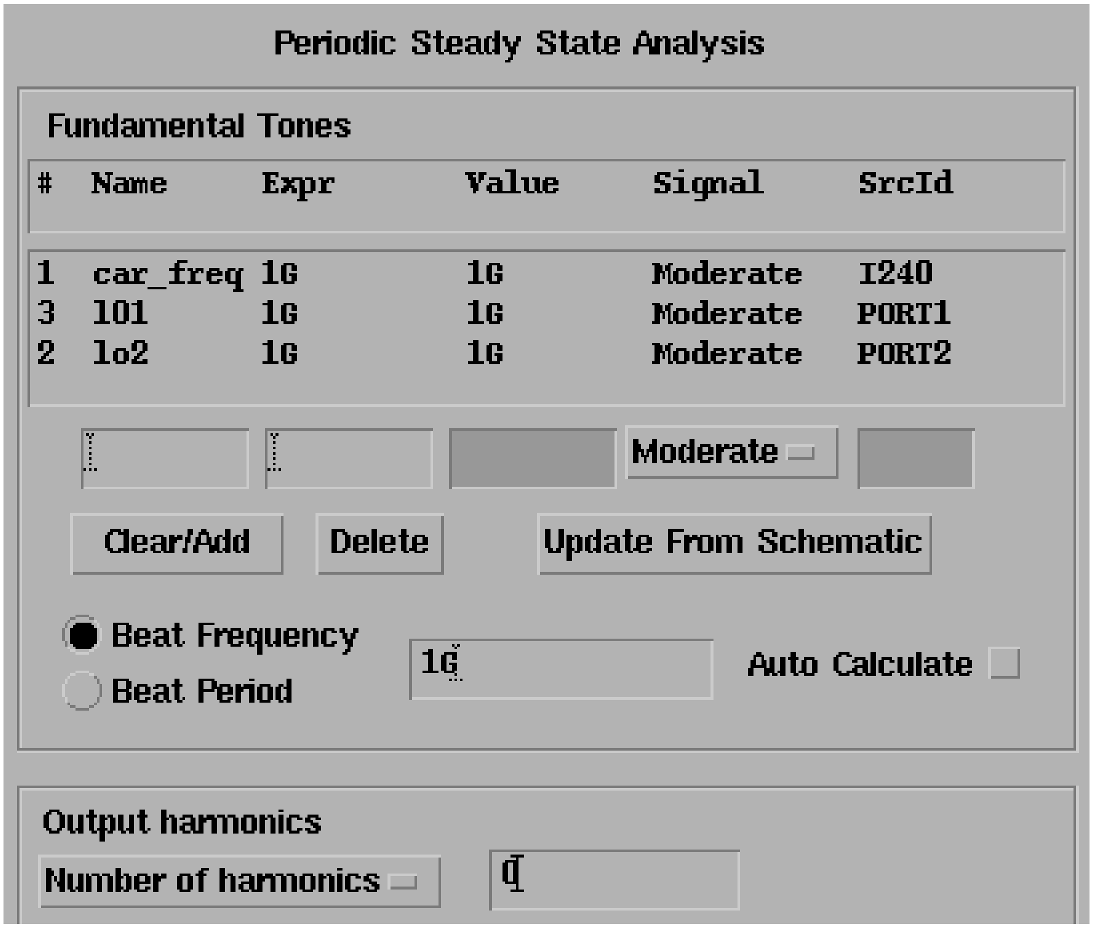
1. In the Choosing Analyses form, click on *`pss`* for the analysis.

   The form changes to display options needed for PSS. You perform PSS first because the periodic steady state solution must be determined before you can perform the small-signal PAC analysis.

2. *Beat Frequency* is highlighted by default. Type `1G` in the *Beat Frequency* field.

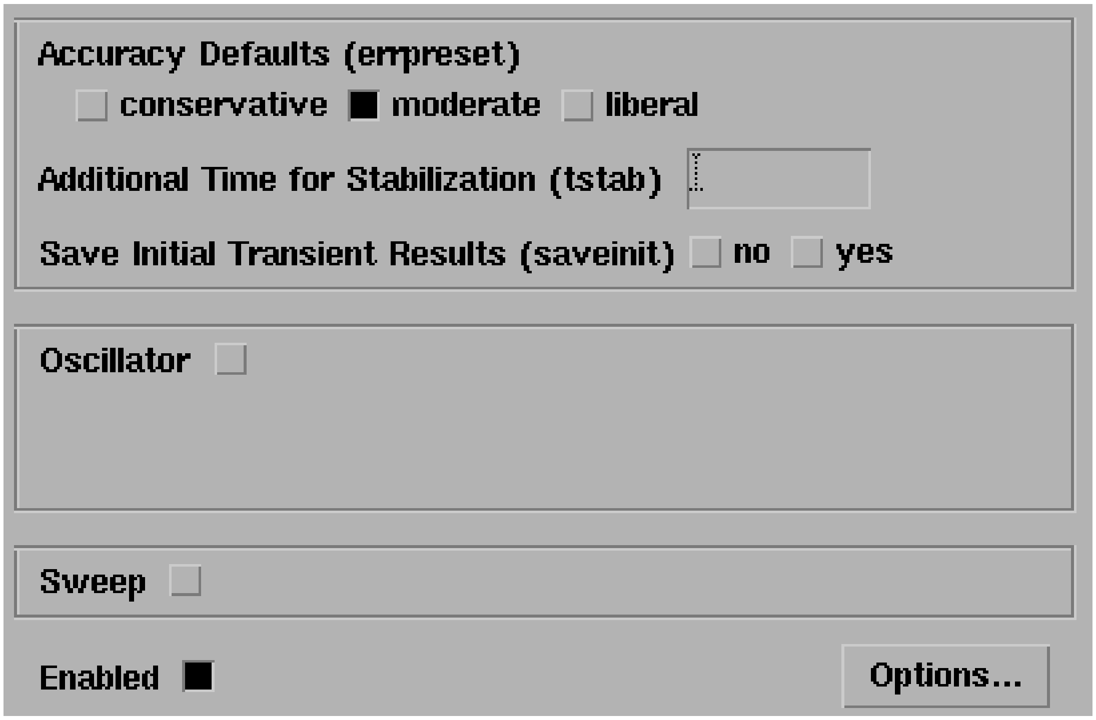
3. Choose *Number of Harmonics* in the *Output Harmonics* cyclical field, and type `0` for the number of harmonics.

The top of the PSS Analysis area looks like the following.



4. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.
5. Highlight *Enabled*, if necessary.

The bottom of the PSS Analysis area looks like the following.

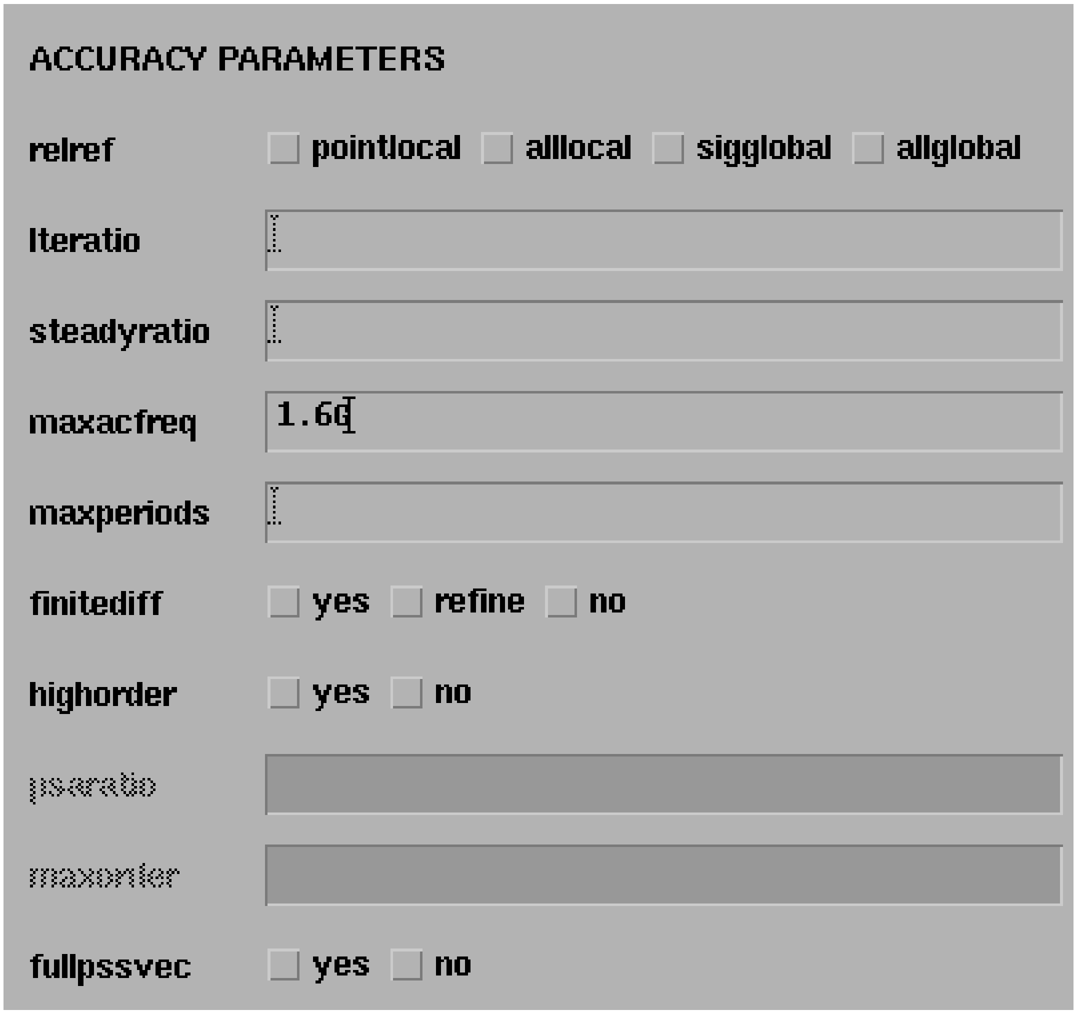


6. Click on the PSS *Options* button.

The Periodic Steady State Options form appears.

7. In the PSS Options form, type `1.6G` for the *maxacfreq* value.

the PSS Options for displays as follows.



8. Click *OK* in the PSS Options form.

   the PSS Options form closes and you return to the PSS Choosing Analysis form.

Setting Up the PAC Analysis

1. In the Choosing Analysis form, choose *pac* for the *Analysis*.

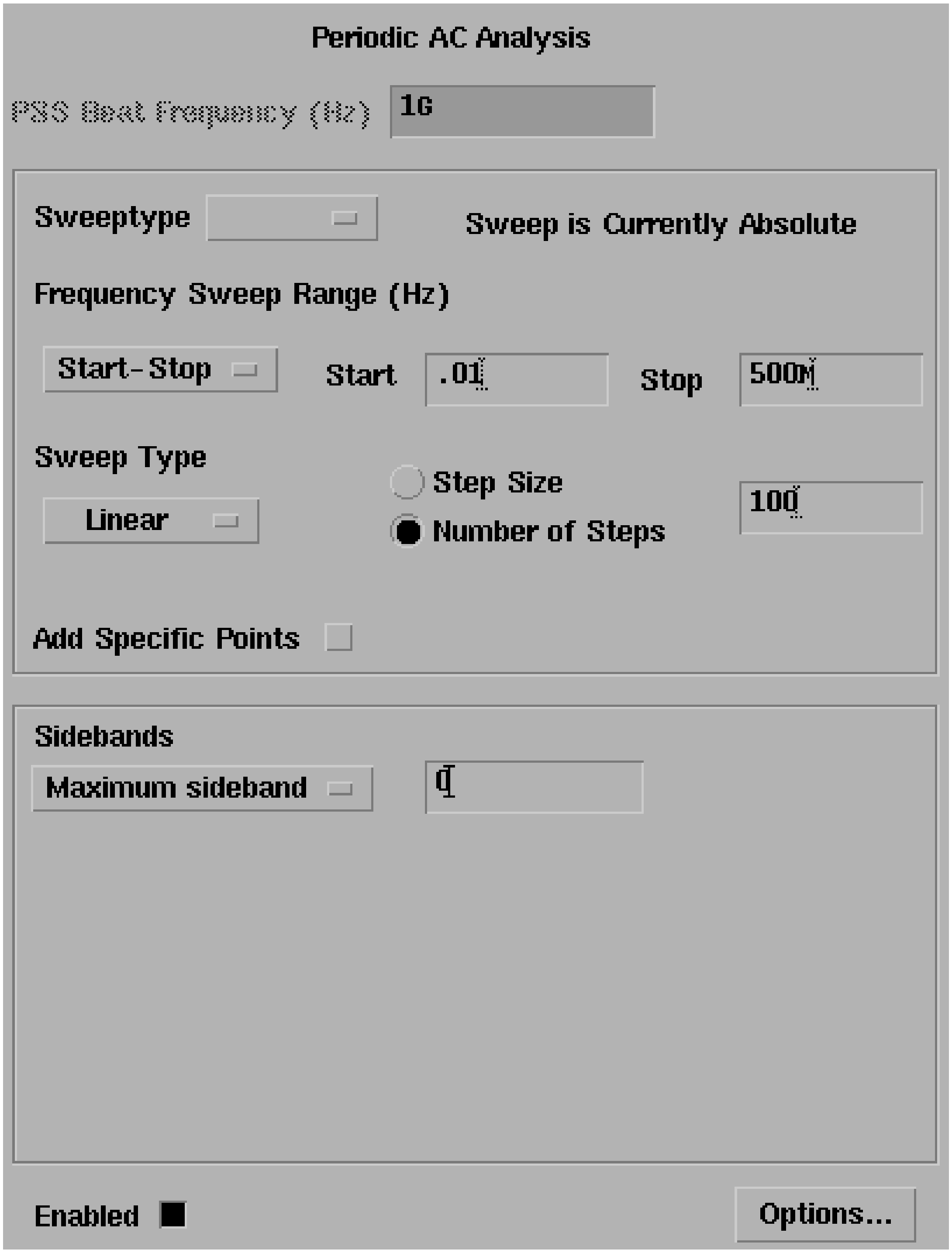
   The form changes to let you specify data for the *pac* analysis that follows the *pss* analysis.

2. Choose *Start – Stop* in the *Frequency Sweep Range (Hz)* cyclical field and type `.01` and `500M` for the *Start* and *Stop* values, respectively.

The starting frequency must be very low but not zero. If the DC values of the PAC results are not captured, the K-model can incur large errors.

3. In the *Sweep Type* cyclical field, choose *Linear*.
4. Highlight *Number of Steps,* and type `100` in the *Number of Steps* field.
5. In the *Sidebands* cyclical field, choose *Maximum Sidebands*, and type `0` for the *Maximum Sidebands* value.
6. Highlight *Enabled*, if necessary, for the *pac* analysis.

The completed *pac* section looks like this.



7. Click *OK* in the Choosing Analyses form.

Running the Simulation

1. In the Simulation window, select *Simulation – Netlist and Run* to run the simulation.

   The output log file appears and displays information about the simulation as it runs. Check the CIW for a message saying the simulation completed successfully.

Checking the Simulation Results

➤ Choose *Results – Direct Plot – Main Form* in the Simulation window.

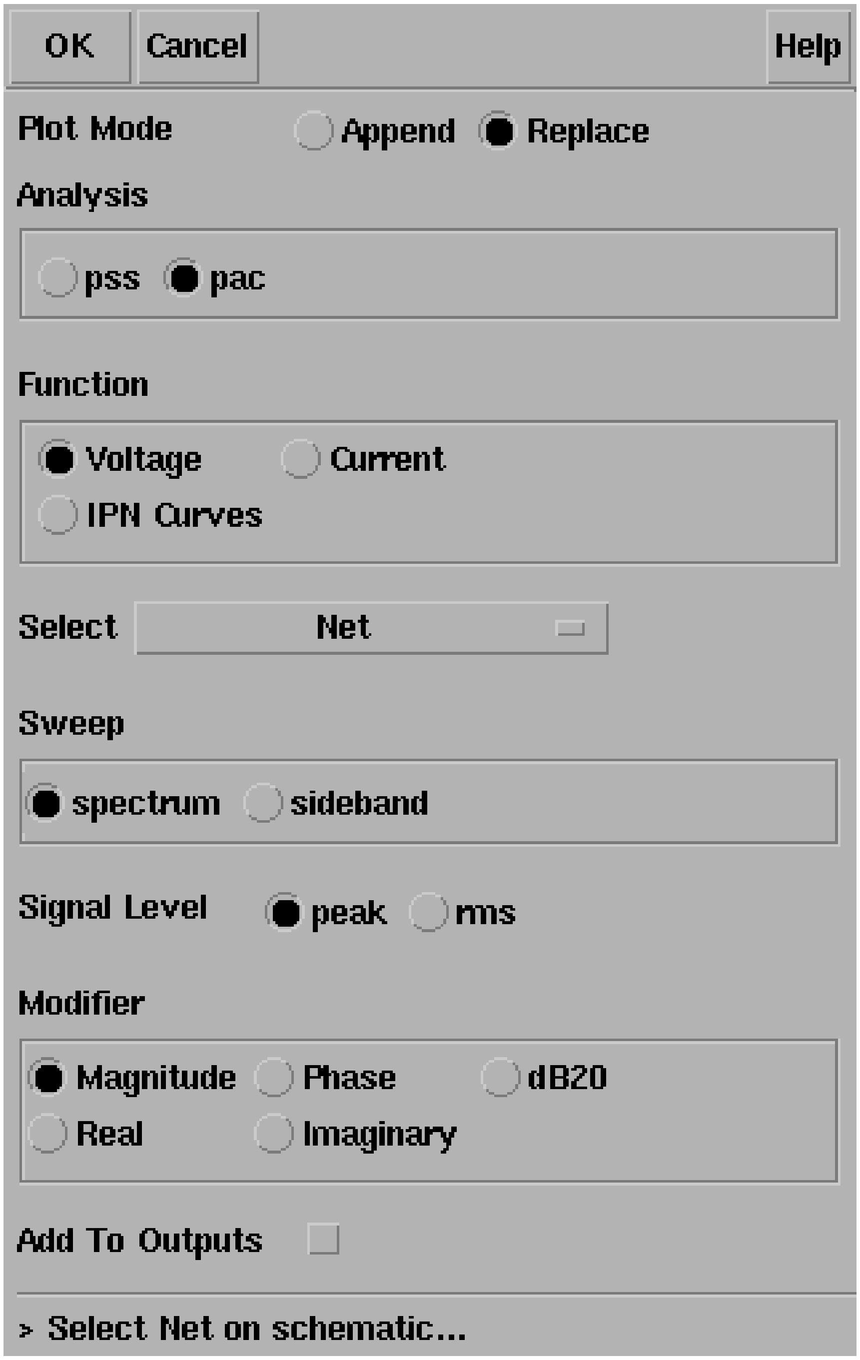
The Waveform window and the Direct Plot form appear.

In the Direct Plot form, do the following:

1. Choose *Replace* for *Plot Mode*.
2. Choose *pac* for *Analysis*.
3. Choose *Voltage* for *Function*.

   The form changes to display information for the *Voltage Function*.
4. Choose *Magnitude* for *Modifier*.
5. Note that the *Select* cyclic field displays *Net*.

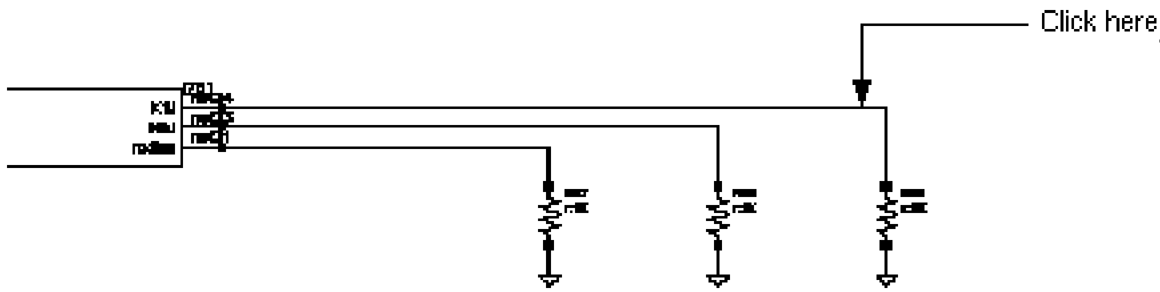
The completed form looks like this.



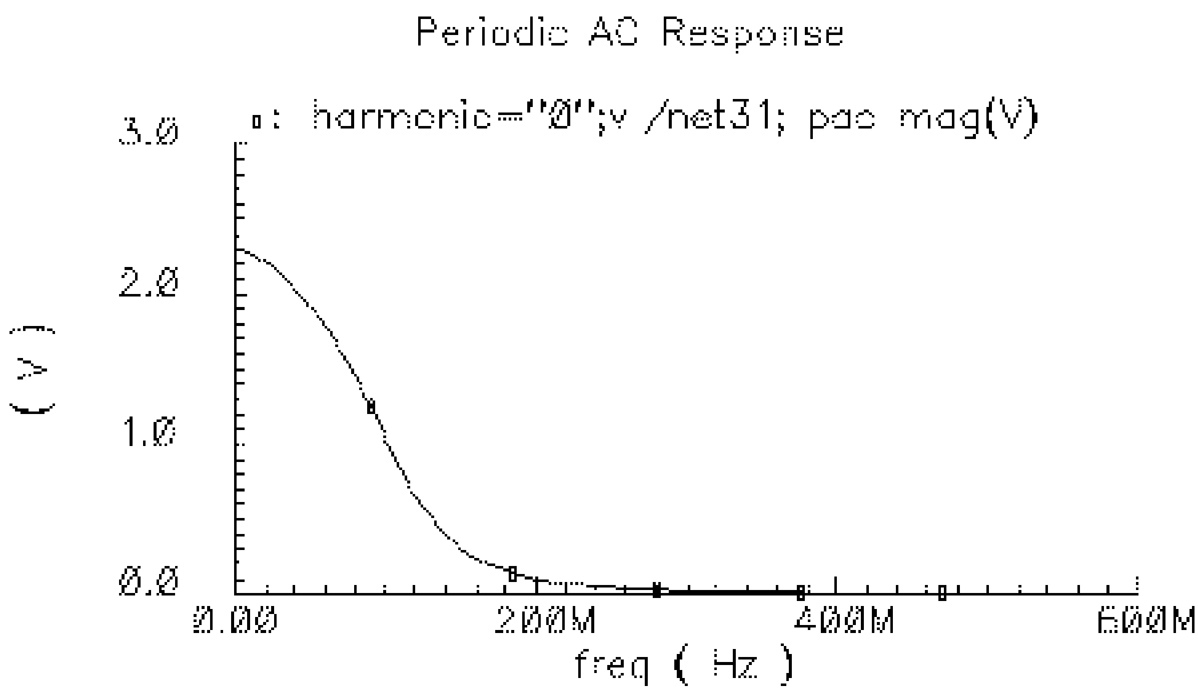
6. Following the message at the bottom of the form,

Select Net on schematic...

In the Schematic window, click on the appropriate wire.



The plot in the Waveform window looks like the one below.



Creating the K-Model

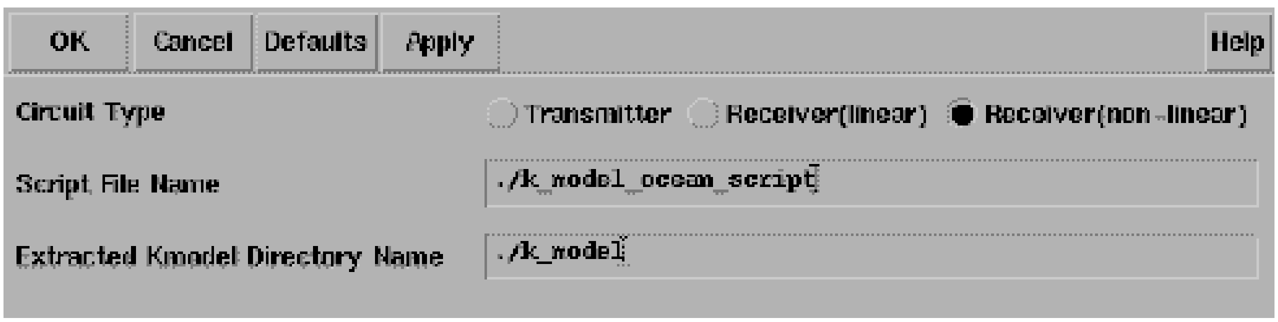
1. In the Simulation window, select *Tools – RF – Link to SPW*.

   The RFIC Modeler for SPW form appears.

2. Choose *Receiver(non-linear)* for the *Circuit Type.*

3. Edit the *Script File Name* and the *Extracted Kmodel Directory Name*, if necessary.

   The file name is the name of the Ocean script to run later, and the directory name is the output directory.



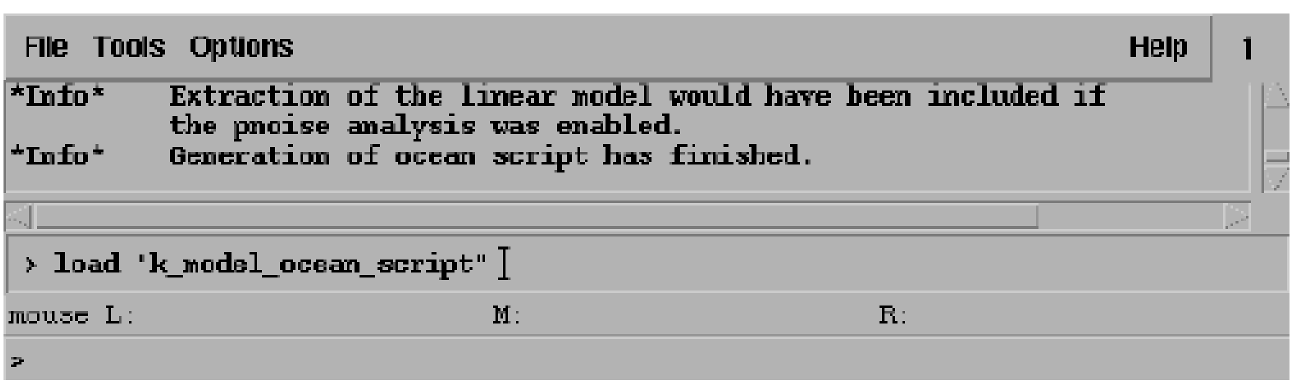
4. Click *OK*.

   In the CIW, watch for a message that the ocean script is generated.

5. In the CIW, load the Ocean script file as follows. Do not forget to include the double quotation marks around the script name.

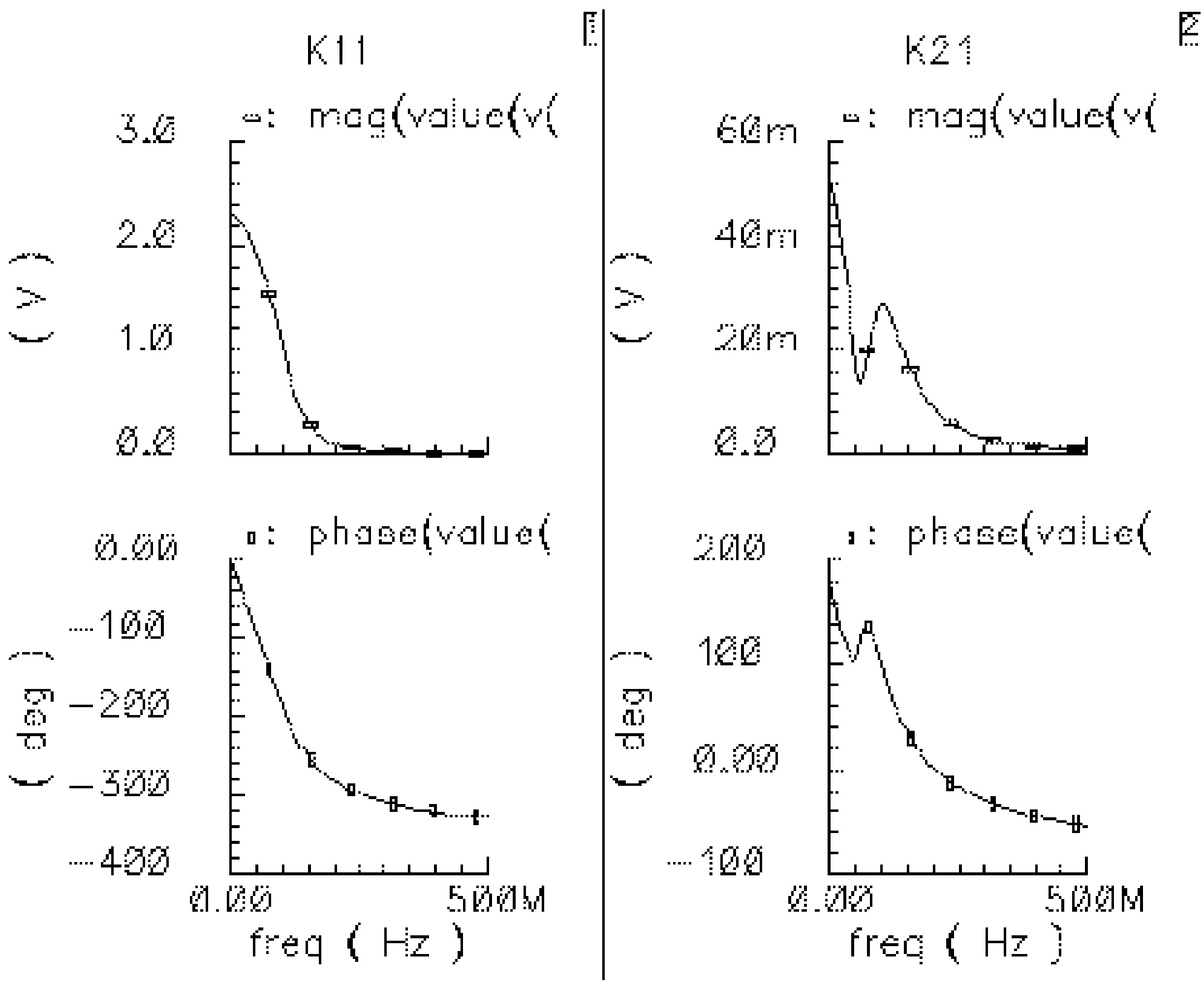
   `load “k_Model_ocean_script”`

6. Press *Return.*

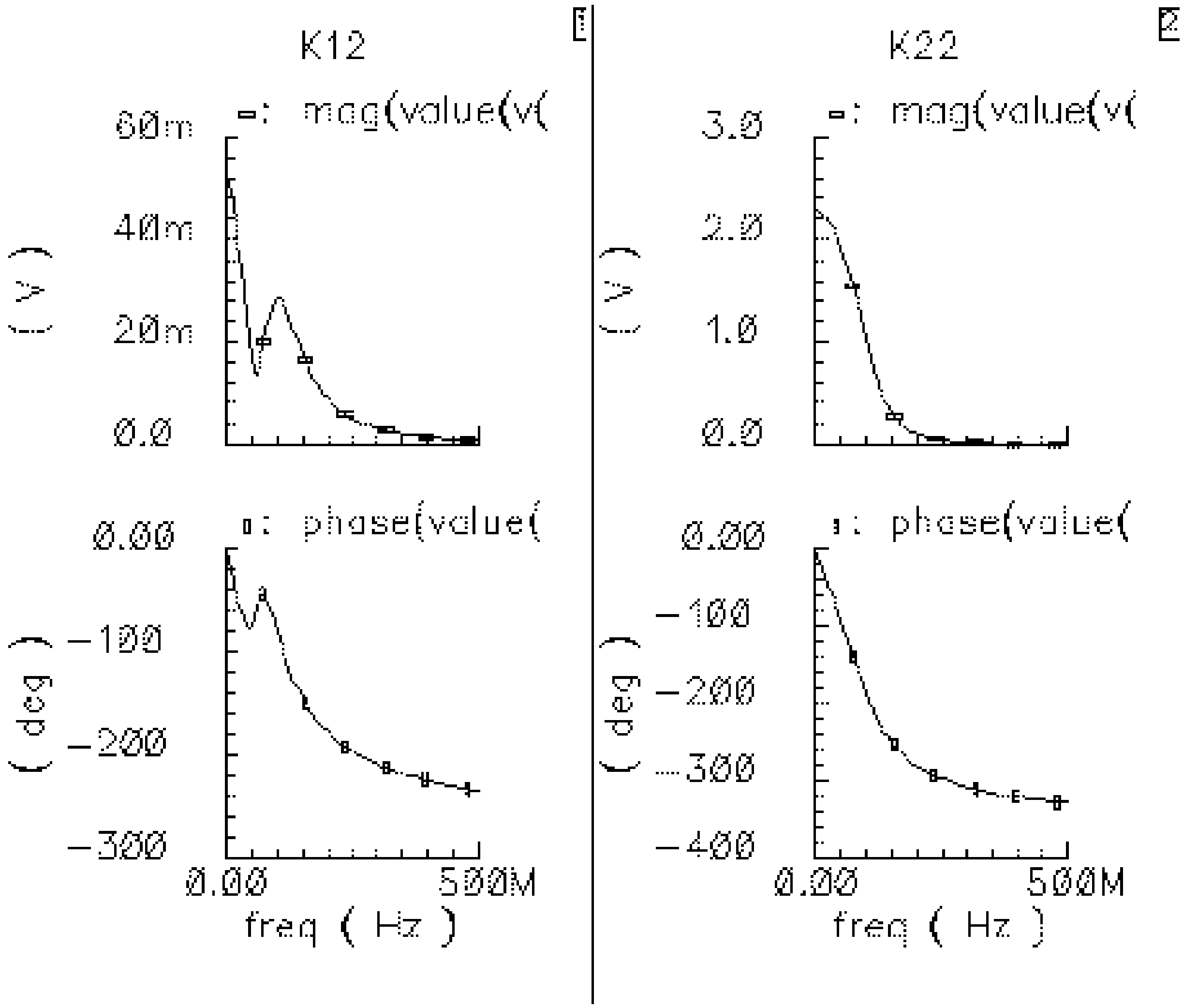


The script runs and generates the k-model.

When the script finishes, it generates the following plots. The K11, K12, K21, and K22 plots are the I-to-I, Q-to-I, I-to-Q, and Q-to-Q transfer functions in the linear K-model.



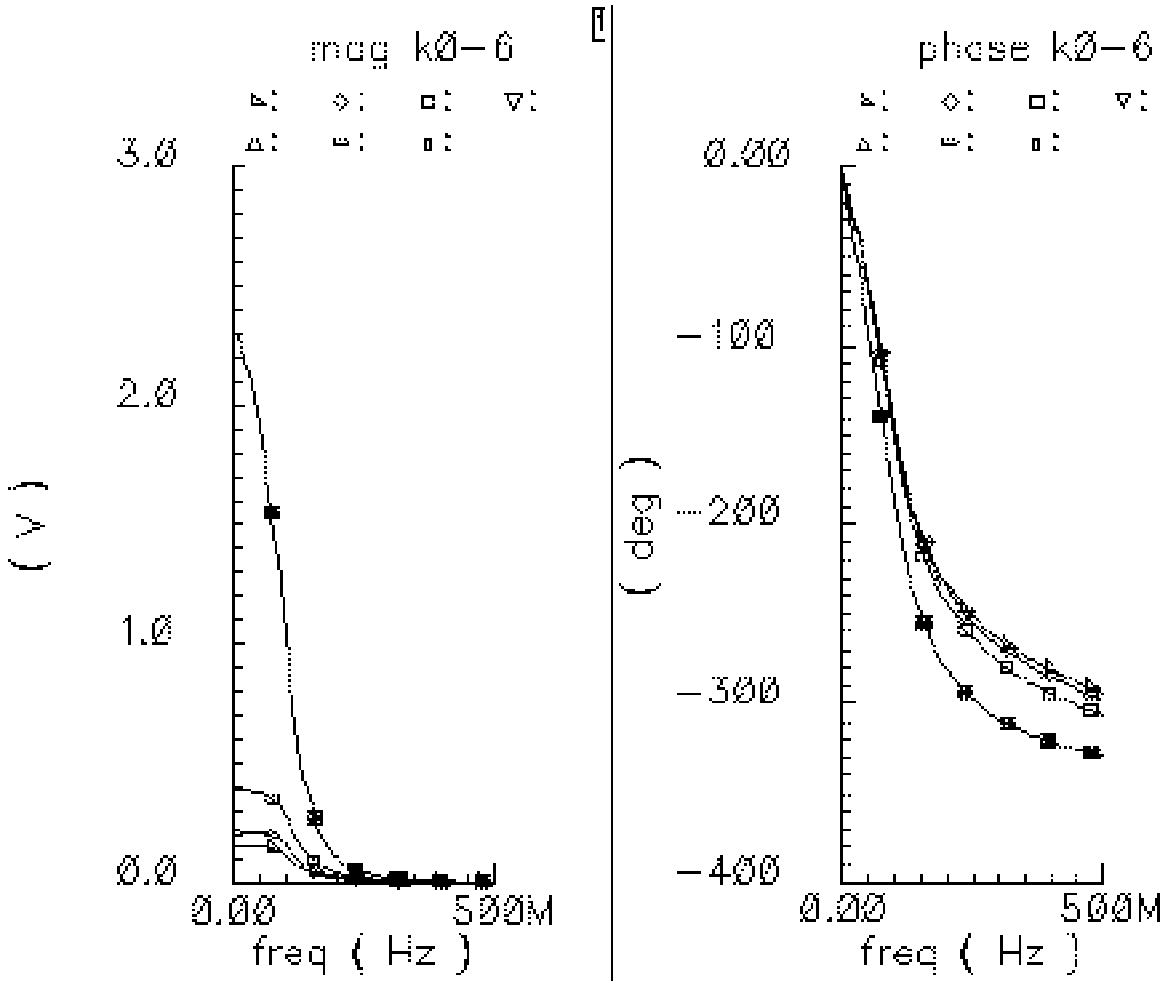
The linear K-model is extracted at the *sig_bias* signal level.



The overlaid curves are the K11 transfer functions. These transfer functions are extracted with I-input signal biases that range linearly from *sig_bias* to *bias_end*.

You can reconstruct these plots by loading the Ocean script in the file located at

<k_model>/plotDir/oceanKmodQuickPlot



Examine the curves for the following features:

- **Narrow-band dynamics:** The K-model does not handle narrow-band dynamics well because it uses FIR (Finite Impulse Response) filters. The most likely narrow-band dynamics result from AGC loops and large DC blocking capacitors. They can introduce ultra-low frequency dynamics that require many FIR filter taps. Ultra-low frequency dynamics appear in the above curves as sharp dips near DC.
- **Non-negligible response at the maximum frequency:** Be sure the data goes to a high enough frequency. If the Alta SPW sample rate, divided by two, exceeds the maximum frequency in the data, Alta SPW will linearly interpolate from the highest frequency data point down to zero at half the sample frequency. Because the receiver response probably decays exponentially rather than linearly, this interpolation can introduce errors.

Alta SPW ignores data beyond half the sample frequency. This does not cause a problem because an Alta SPW input signal probably would have no power beyond half the sampling frequency.

More About the K-Model

Like any behavioral model, the K-model has limitations and requirements. The assumptions on which the K-model is based cause the basic limitations. Other limitations are caused by practical considerations. There are also requirements that are specific to the Alta SPW environment. Finally, this section discusses some limitations of the K-model's ability to model out-of-band blockers.

Limitations and requirements resulting from linear K-model assumptions

The linear K-model makes the following two assumptions beyond the small signal assumption:

- Unless the SpectreRF model includes detailed A/D converter models, the extracted linear K-model assumes that noise at the base-band outputs does not change significantly over the A/D aperture time.

  The first K-model assumption involves noise and A/D conversion. The linear model includes noise generated within the front end. "Aperture time" is the time the A./D converter requires just to measure a voltage, not to measure and hold it. The assumption is that the noise is constant over the aperture time. This implies the noise is band limited. However, as the following discussion explains, the noise bandwidth must also not be too small.

  Note that the The linear model does not account for phase noise. However, if one knows the power spectral density of the phase noise one can easily follow the K-model with a complex multiplier that shifts the phase of the K-model output. A filter driven by white Gaussian noise would generate the random phase shift.

- The two equivalent output noise sources, one for the I output and one for the Q output, are uncorrelated.

  Frequently, much of the noisy circuitry is common to both outputs. Despite the common circuitry, the K-model assumes the two output noise signals are independent. This is a good assumption if the noise is white over the frequencies of interest. In summary, the noise must be white with respect to filters preceding the A/D conversion but band-limited with respect to 1/(aperture time).

  Why isn't there just one K-model, a noisy non-linear K-model instead of a noisy linear K-model and a noise-less non-linear K-model? Putting small signal noise in the non-linear

K-model would suggest that the model's noise varied with input power. It does not. Furthermore, noise analysis is usually of interest in bit error rate simulations. Such simulations are usually long and the desired signal is weak. The linear K-model suffices for weak desired signals and runs faster than the non-linear K-model.

Limitations and requirements resulting from non-linear K-model assumptions

The K-model requires two basic assumptions:

- Small-signal components of the base-band input can be wide-band, but large components must fall within the receiver's bandwidth.

  The signal of interest is usually within the receiver's bandwidth, and interferers usually lie outside this bandwidth. However, if the receiver has a band selection filter immediately after the antenna, and that filter can be modeled apart from the K-model, this first assumption is not a severe restriction. This is true because the filter makes the interferer small by the time it reaches the K-model.

- Non-linear behavior depends only on the input radius. Non-linear PM/PM and PM/AM effects are negligible.

  If the input signal is

  `rf(t)=a(t)cos(wct)-b(t)sin(wct)`

  the base-band signal is

  `a(t)+jb(t)`

  a trajectory in the complex plane. This is the rectangular representation.

  The polar representation is in terms of radius and angle. The radius is

  `Sqrt[a(t)a(t)+b(t)b(t)]`

  The angle is

  `ArcTan[b(t)/a(t)]`

  Requirements resulting from the Alta SPW environment

  Both K-models are part of Alta SPW, the Cierto Signal Processing Worksystem. Alta SPW is a DSP design tool. This environment creates the following two consequences you must remember:

- Model extraction requires base-band input and output ports.

Because it is a base-band tool, Alta SPW operates on the information impressed on the RF carrier rather than the instantaneous value of the RF signal. Each of two carrier phases, sine and cosine, carry information. Because Alta SPW represents base-band signals as complex numbers, the modeled RF circuit must have base-band input and output ports.

The input test jig creates the input port. The input test jig mixes the base-band input signal up to the appropriate carrier frequency, thereby synthesizing an antenna signal from a base-band input. Changing the phase of the carrier by 90 degrees switches the base-band input between the "in-phase" and "quadrature" input ports. The Ocean extraction script changes carrier phase automatically when it extracts a K-model.

If the circuitry does not contain two base-band output ports, you must synthesize them. You can synthesize them easily using ideal behavioral elements. For example, to build a K-model of only the low noise amplifier, you add ideal downconverters to the SpectreRF model and extract a K-model of the combination.

You can partition a receiver into smaller K-models only if there is no non-linear interstage loading at the break points and if the carrier fundamental is the only important quantity.

The input and output test jigs also tell the extraction script where to excite the SpectreRF model and what quantities to measure.

- The SpectreRF noise analysis must extend far enough to capture all important features, including those beyond half the Alta SPW sampling frequency.

  This is true because the power spectral density suffers aliasing in the DSP environment. The only data available for DSP is for discrete, evenly-spaced time-points. This restriction imposes a constraint on possible input signals that is usually met within the design. Specified input signals must not have significant power beyond half the sampling frequency.

  The channel and RF front end are cascaded physical entities with no sampler between them. But because they are modeled in Alta SPW as separate blocks, Alta SPW implicitly assumes an ideal sampler lies between the channel and front end. That assumption is only valid for base-band signals that have no significant spectral components near or beyond half the sampling frequency. If that assumption were not valid the overall design would have far more serious problems than K-model accuracy.

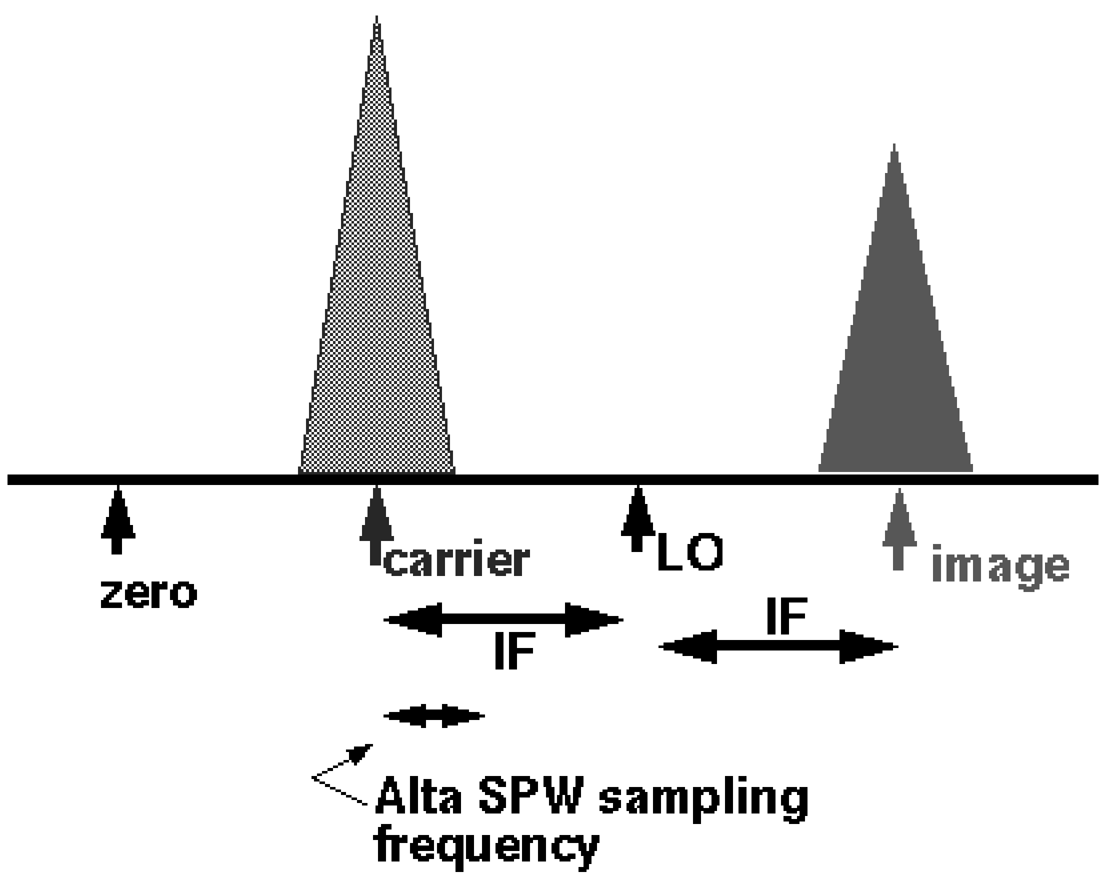
  Because the noise is not a signal you specify, it need not meet that constraint. Noise power that is well beyond half the sampling frequency can alias down to the band of interest. If the noise is characterized to a high-enough frequency, the K-model automatically aliases it.

Practical limitations of the K-model

- **Image rejection**

  Although the K-model theoretically models image rejection, such modeling is not practical for the Alta SPW implementation Figure 8-2 on page 637 shows how the carrier, local oscillator, and image frequencies can be related. The Alta SPW sampling frequency is probably approximately the bandwidth of the RF receiver, shown in blue. To simulate dynamics at the image frequency, the Alta SPW sampling frequency must span twice the difference between the carrier frequency and the image frequency. Such a high sampling frequency requires an impractical number of FIR filter taps to model the receiver dynamics. Simulation time is almost as lengthy as with an unsuppressed carrier simulation.

Figure 8-2 Carrier, Local Oscillator, and Image Frequency Relationships for the Alta SPW Implementation



- **Ultra-low frequency dynamics**

Ultra-low frequency effects such as those caused by PLLs, AGC loops, and large DC blocking capacitors again require many FIR filter taps. However, behavioral phaselock loop models or AGC loop models can precede the K-model in Alta SPW to approximate these low frequency effects.

Limitations in modeling large out-of-band blockers

The Figure 8-3 illustrates a basic K-model limitation. The basic K-model does not accurately simulate the effects of large out-of-band blockers. However, if the blocker is narrow-band enough to be considered deterministic (ideally, a single tone), you can extract a model that accurately simulates blocker-induced gain compression by extracting a K-model with the blocker present.

Figure 8-3 K Model with Out of Band Blockers

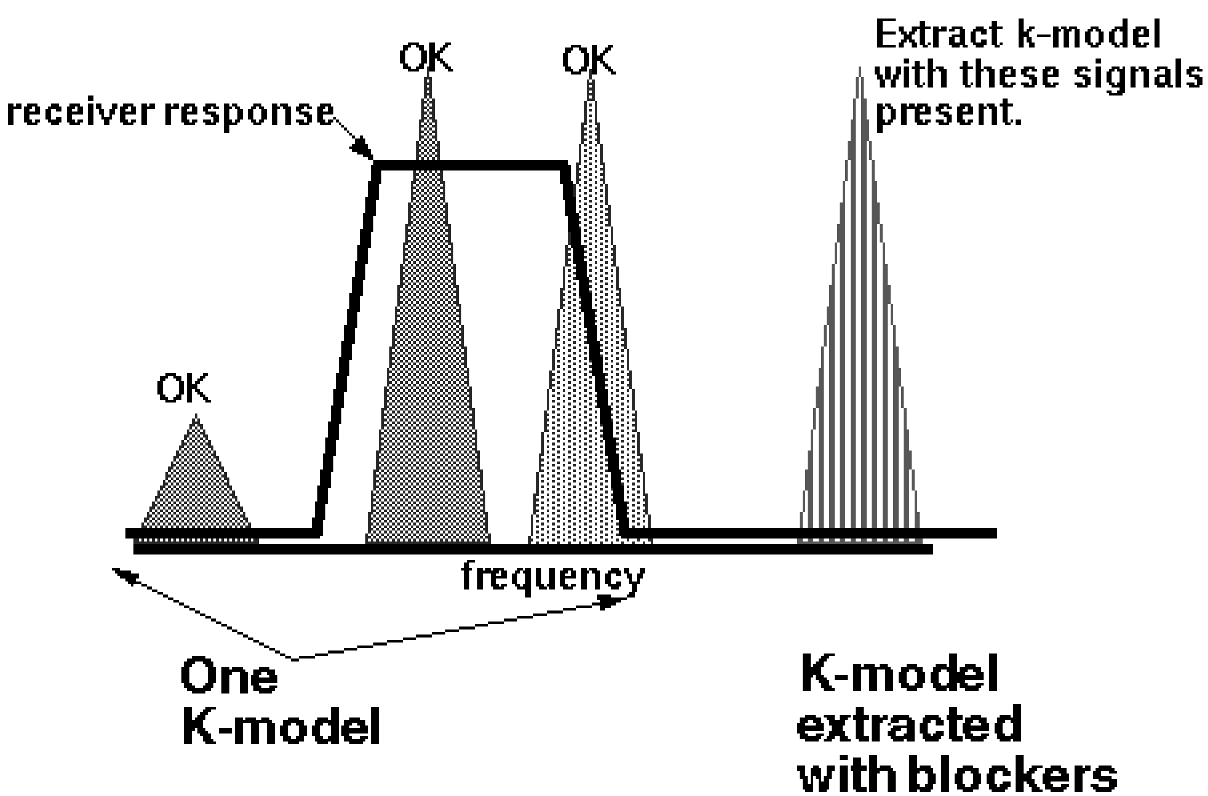
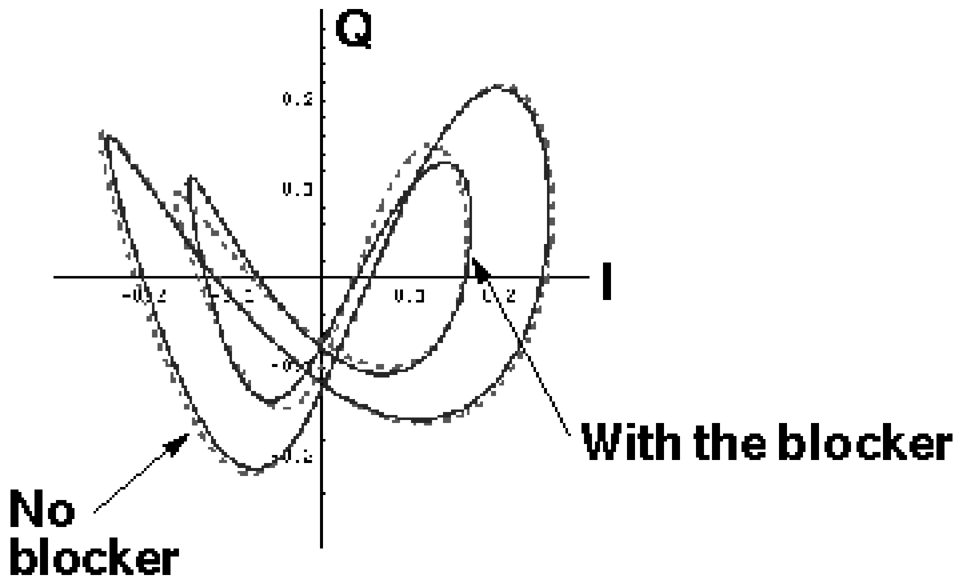


Figure 8-4compares receiver outputs computed using a transistor-level model in SpectreRF with outputs that result from using K-models in Alta SPW. One K-model includes the blocker. The other does not.

Figure 8-4 Results Calibrated with a Stronger Signal



The I-input signal is in-band and the Q-input signal is on the band edge. The blocker is more than two decades out of band. The blocker's amplitude is more than 10 times the in-band signal level that saturates the receiver.

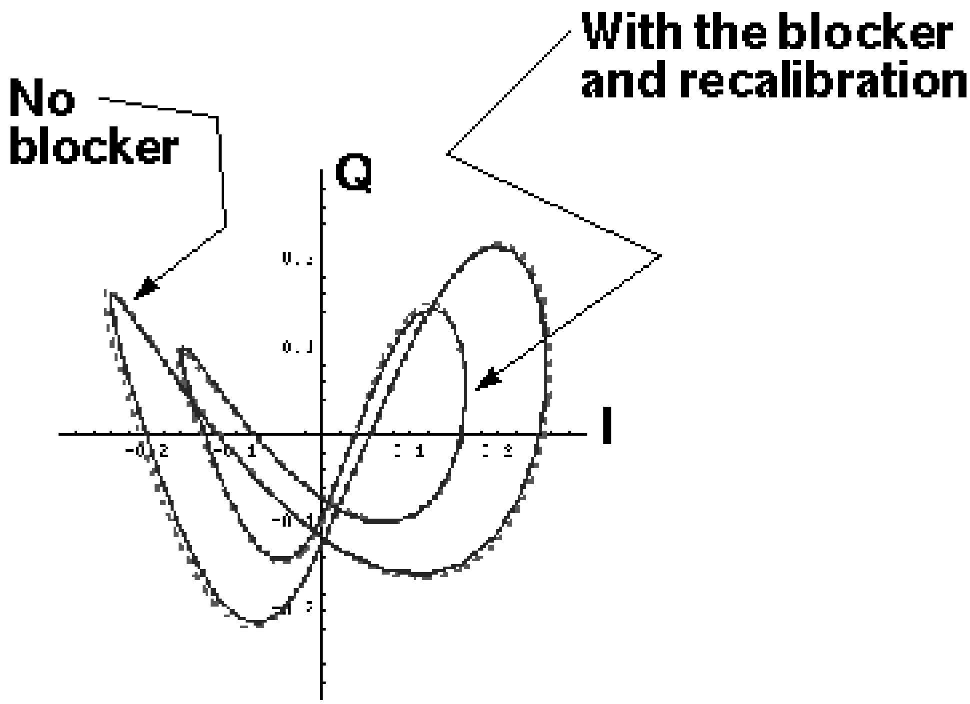
The red dotted trajectory shows direct SpectreRF simulation results. The solid blue line shows results from the K-model simulation. The larger trajectories are without the blocker.

The K-model extracted without the blocker matches device-level simulation quite well. The K-model extracted with the blocker present also matches the corresponding device-level simulation well except for a fixed rotational error. Rotating the blue trajectory by 7 degrees puts it nearly on top of the trajectory computed in SpectreRF. Such a fixed rotational error is usually irrelevant to the DSP section because the equalizer hides the error, and a fixed rotation does not affect the spectrum.

However, you can reduce the error. The K-model internally calibrates for rotational error. A blocker introduces an offset that creates calibration error. Alta SPW can modify the K-model to calibrate at a stronger input signal so that the offset has less effect.

Figure 8-5 shows the results with the K-model calibrated at the stronger signal.

Figure 8-5 K Model Calibrated at the Stronger Signal



K-model data files

This section describes the K-model data files.

I_data.ascsig and Q_data.ascsig

These files characterize the static behavior of the *I* and *Q* outputs, respectively, as the input bias increases along the input *I*-axis. The format is given below:

```
data
$
signal type = double
vector type = interlaced
vector length = 14
number of vectors = 2
number of signal points = 14
sampling frequency = 1
starting time = 0
```

```
$
0.001   0.002300
0.3008333   0.692013
0.6006667   1.381728
0.9005   2.071483
1.200333   2.568851
1.500167   2.655311
1.8   2.710871
```

dck

This file contains the lowest frequency gains of the *K11* and *K21* transfer functions. The format is shown below:

```
2.299950
-0.051965
```

sig_biases

This file lists the input signal biases. The format is shown below:

```
0.001
0.3008333
0.6006667
0.9005
1.200333
1.500167
1.8
```

psd_I.ascsig and psd_Q.ascsig

These files contain the power spectral densities of the noise observed at the baseband *I* and *Q* outputs. The first column is frequency, and the second column is power spectral density in volts/Sqrt(Hz). The format is given below:

```
# wavew4s1i1() from 10e-3 to 25e6
#             wavew4s1i1()

10e-3         1.334e-6
250e3         943.1e-9
500e3         943.1e-9
750e3         943e-9
1e6           942.7e-9
```

```
1.25e6        942.4e-9
1.5e6         941.8e-9
1.75e6        940.9e-9
2e6           939.6e-9
2.25e6        938e-9
2.5e6         935.9e-9
```

K11.ascsig, K21.ascsig, K12.ascsig, K22.ascsig, k11_1 through k11_6, and k21_1 through k21_6

The first four files contain the *I-I*, *I-Q*, *Q-I*, and *Q-Q* transfer functions at the smallest input bias. The format is identical to the *k21_j* and *k11_j* transfer functions.

The *k11_1* through *k11_6* files contain the transfer functions from the *I*-input to the *I*-output as the input bias sweeps from the *sig_bias* level to the *bias_end* level. The *k21_1* through *k21_6* files contain the transfer functions from the *I*-input to the *Q*-output for the same input bias levels. The format of all of these files is the same. The first column is the frequency, the second is the magnitude in volts/volt, and the third is the phase in degrees. You must keep exactly three lines of header. The format is shown below:

```
freq (Hz)         mag(harmonic(v    phase(harmonic

10e-3                 2.300           -18.860e-9
1e6                   2.300            -1.886
2e6                   2.299            -3.771
3e6                   2.298            -5.656
4e6                   2.297            -7.540
5e6                   2.295            -9.424
6e6                   2.293           -11.310
7e6                   2.290           -13.190
```

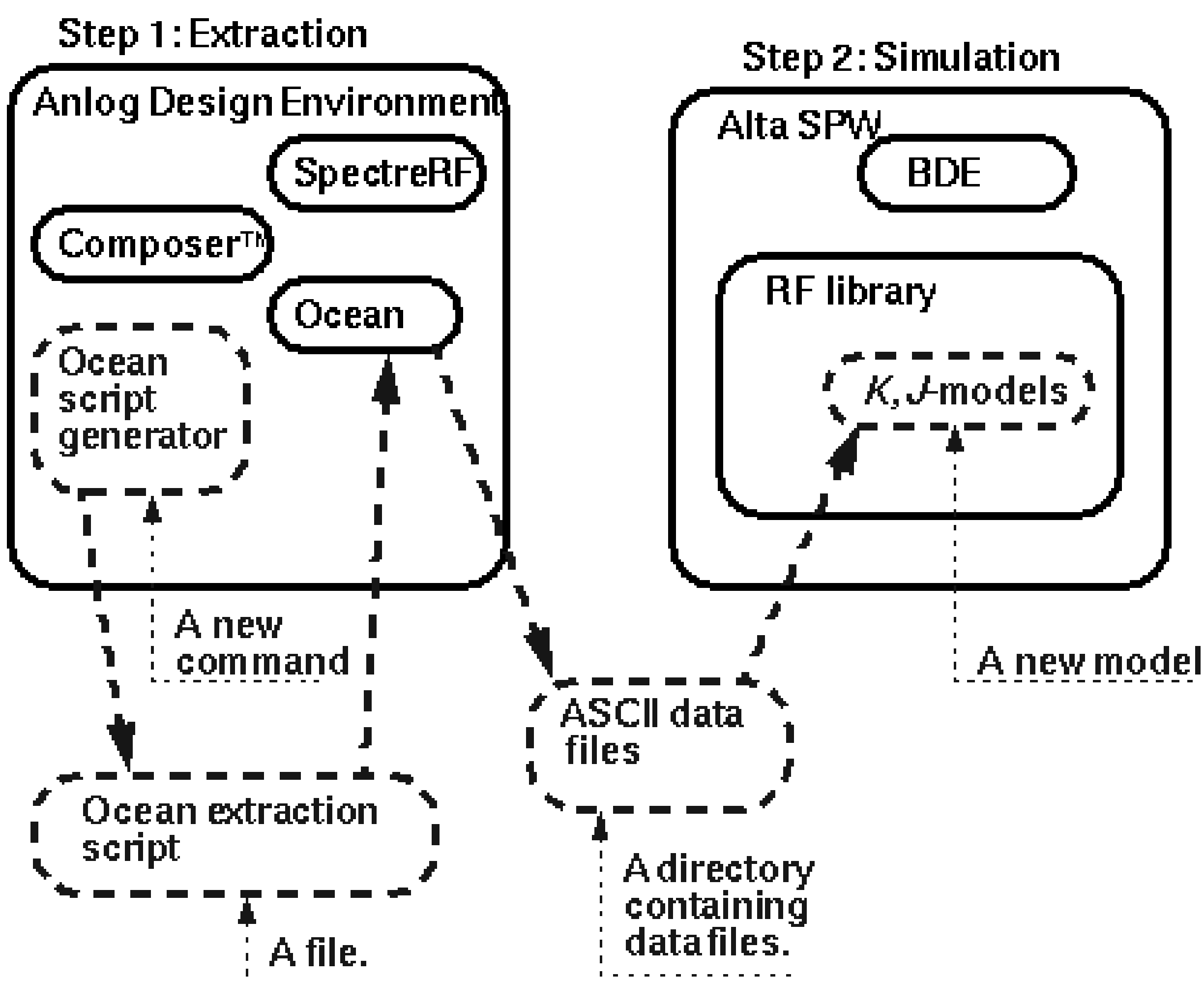
The rest of the files are obsolete binary files. They formerly supported the Alta SPW beta version of the K-model. The released version uses only the ascii files.

9

Creating and Using Transmitter J-Models

The J-model, also known as a transmitter K-model, links SpectreRF and Alta™ SPW as shown in Figure 9-1.

Figure 9-1 Task Flow of the J-Model Creation and Use



The J-model, also known as a transmitter K-model, has two primary uses.

- The J-model facilitates Adjacent Channel Power Ratio (ACPR) estimates
- The J-model imports transmitter impairments into the Alta™ SPW environment.

You can extract a J-model and then feed it directly into SPW by instantiating a J-model from the Alta SPW library and specifying the directory containing the extracted J-model data. An Alta SPW user can use the J-model to examine how a direct conversion transmitter affects the end-to-end bit error rate.

Only the model extraction part of the SPW flow is described in this chapter. The analog design environment *rfLib* includes Verilog-A versions of the J-model that read the same data files. The *j_mod_extraction_example* simulation procedures in this chapter describe how to use the Verilog-A version of the J-model to compute ACPR in the analog design environment. The last part of this chapter briefly describes the structure of the J-model and files created by the extraction script.

The example in this chapter is highly ideal but is still a good example because it simulates quickly yet contains the main distortion mechanisms captured by the J-model.

Procedures for Simulating j_mod_extraction_example

Follow the steps below to simulate the *j_mod_extraction_example*:

➤ Copy *j_mod_extraction_example* from the Cadence *rfExamples* directory into a local directory.

  Be sure the path to the local directory is defined in your *cds.lib* file.

➤ Choose *File – Open* in the CIW.

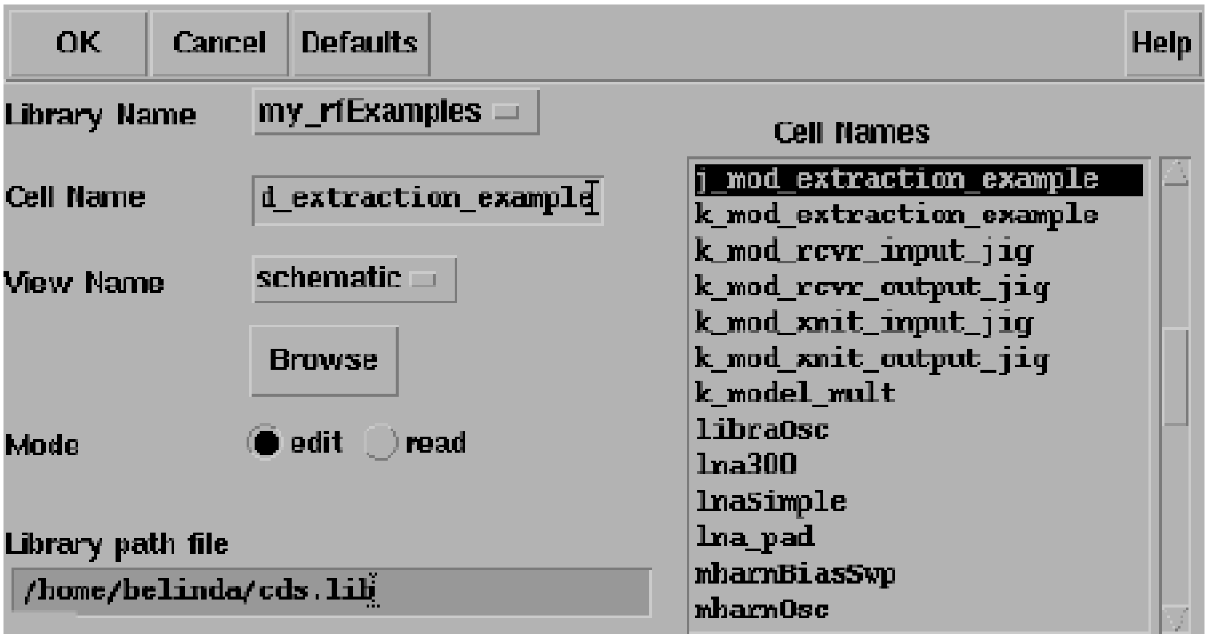
  The Open File form appears. Filling in the Open File form lets you open the schematic.

In the Open File form, do the following:

1. Choose *<Path_to_local_copy>* for *Library Name*.
2. Choose *schematic* for *View Name*.
3. In the *Cell Names* list box, highlight *j_mod_extraction_example*.

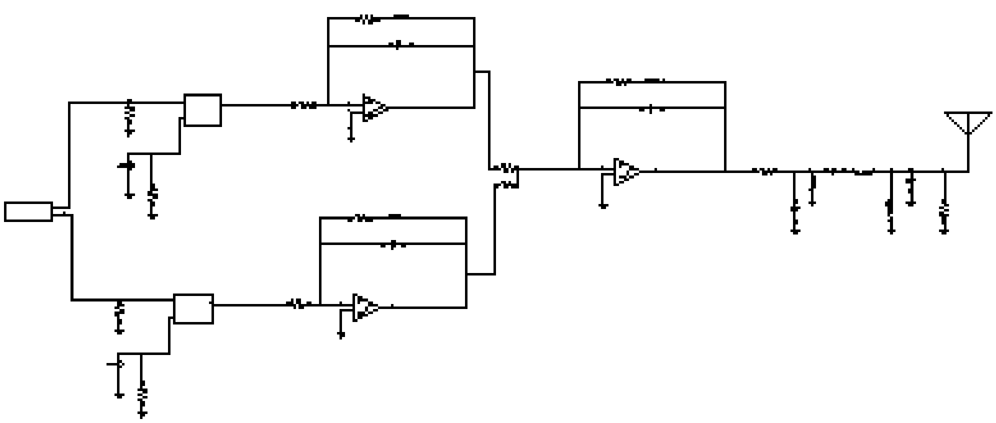
   j_mod_extraction_example appears in the *Cell Name* field. (The j-model is another name for a transmitter k-model.)

4. Highlight *edit* to choose the *Mode*.



5. Click *OK* in the Open File form.

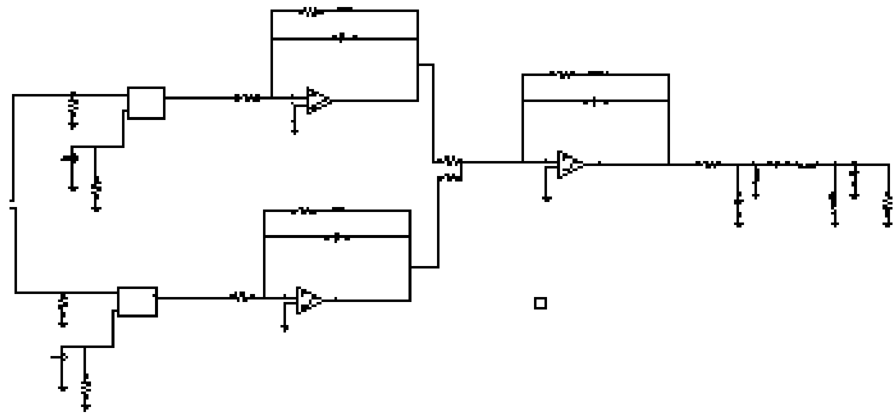
The Schematic window opens and displays the schematic.



Optional: Setting Up the Input and Output Jigs

If you want to see the whole procedure for setting up the J-model, including the editing of the schematic, delete the input and output test jigs at the far left and right of the schematic. The edited schematic now looks like the one below.

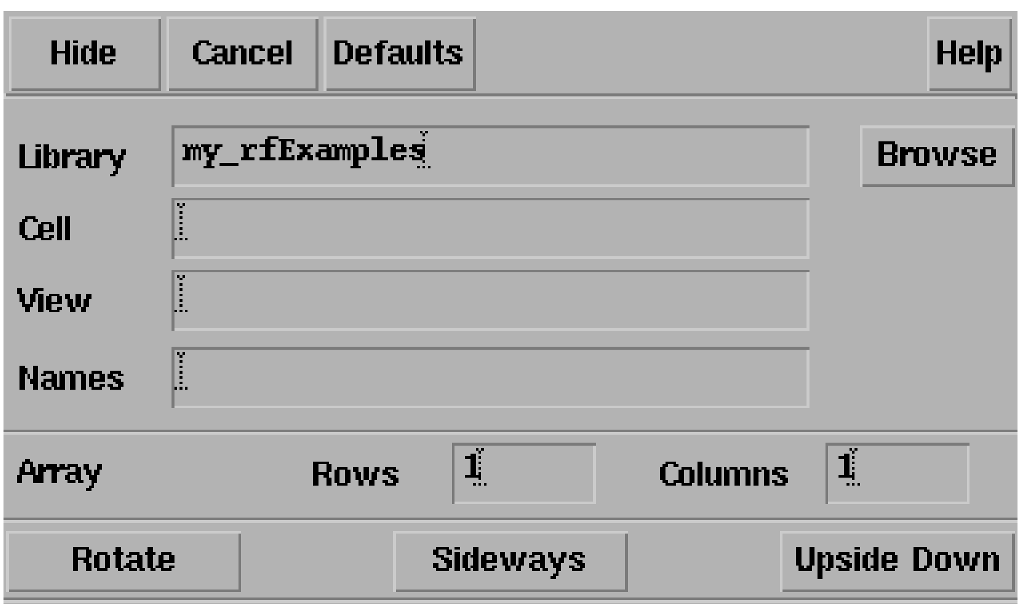
If you do not want to edit the schematic but only want to set up and run the simulation, skip to the Analysis Setup section.



From the Schematic window, do the following:

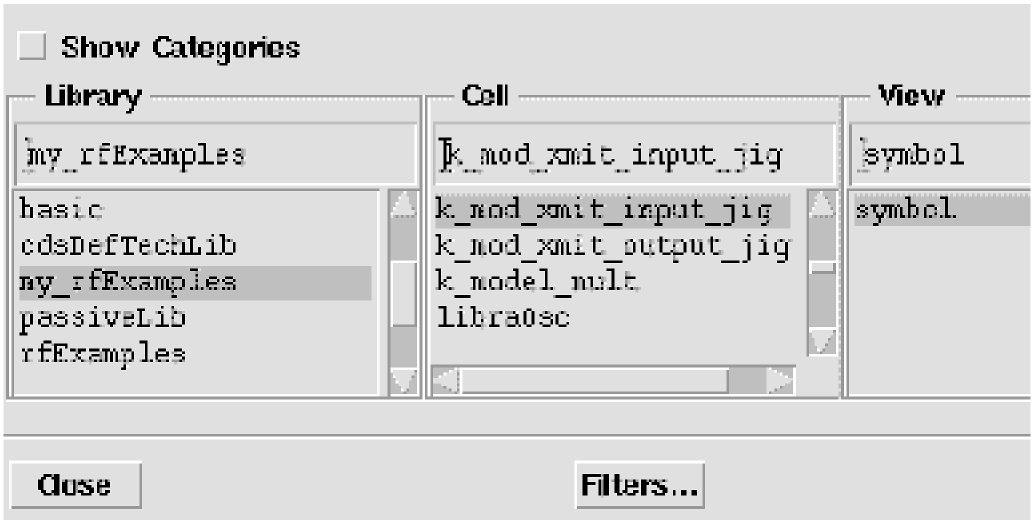
1. Choose Add – *Instance.*

   The Add Instance form appears.



2. In the Add Instance form, click on *Browse*.

The Library Browser – Add Instance form appears.



In the Library Browser – Add Instance form, do the following:

1. Choose *my_rfExamples* for the *Library*. (Choose the name of the editable copy of the *rfExamples* library that you created.)

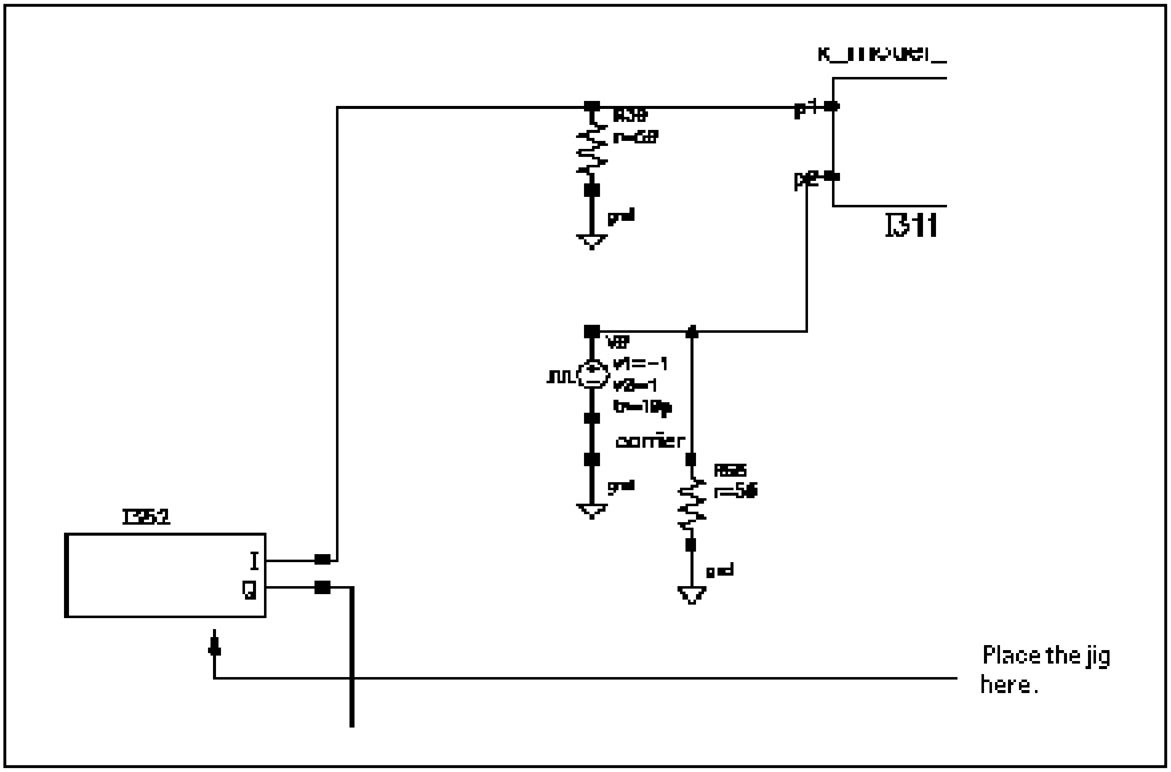
   The form changes to let you choose a cell.

2. Choose *k_mod_xmit_input_jig* for the *Cell*.

   The form changes to let you choose a *View*.

3. Choose *symbol* for the *View*.

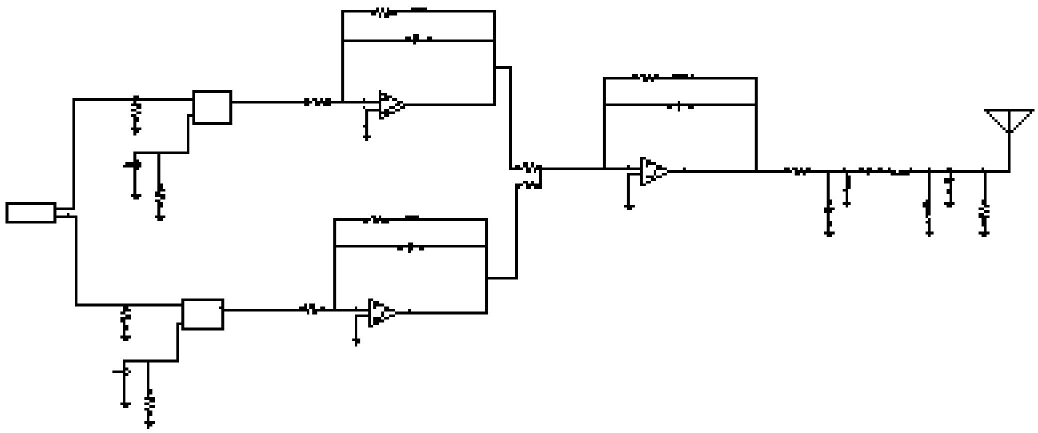
When you place the cursor back in the Schematic window, an instance of *k_mod_xmit_input_jig* is attached to it. Place the input jig at the appropriate place in the schematic.



If you need assistance with the procedures for editing a schematic, see the Virtuoso® Schematic Composer™ User Guide.

In the Schematic window, again choose *Add – Instance.*

Now repeat the steps you used to add the input jig to the schematic and add the output jig *k_mod_xmit_output_jig.* The edited schematic looks like this.



In the Schematic window, perform the following steps:

1. Click on the input jig to select it.
2. Choose *Edit – Properties – Objects* in the Schematic window menu choices.

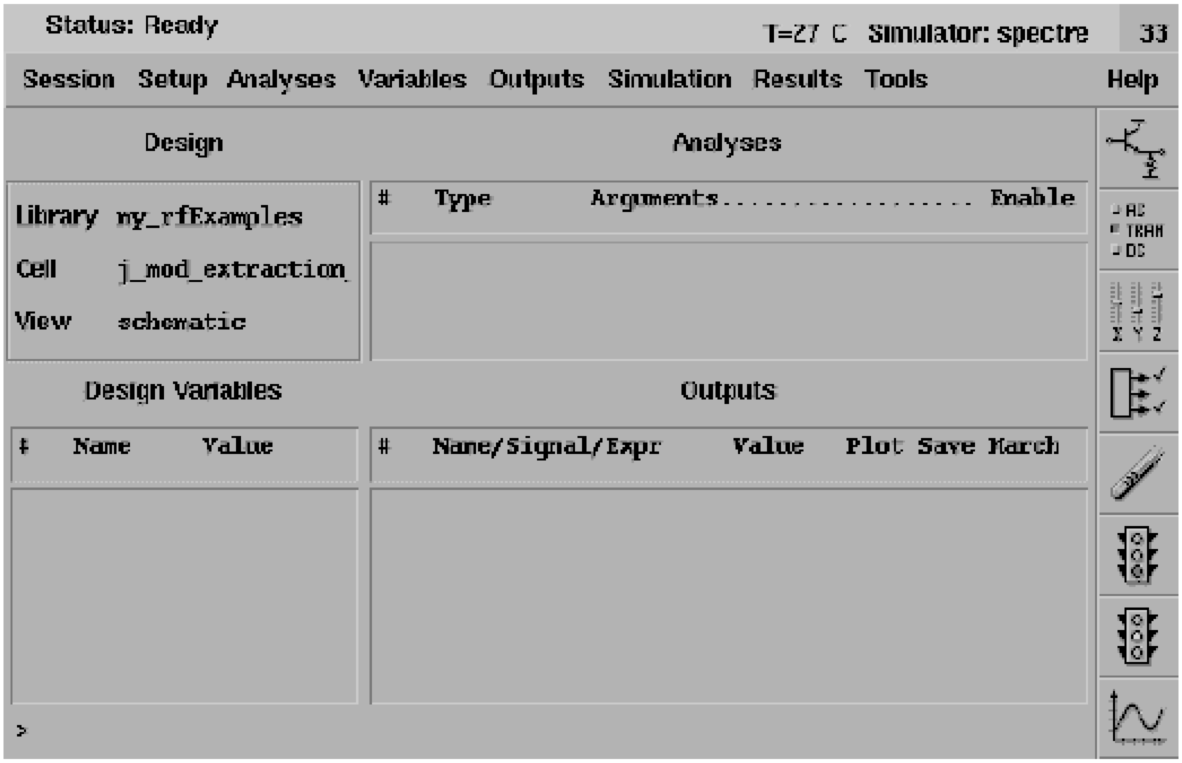
   The Edit Object Properties form appears.

   a. In the Edit Object Properties form, type `1M` in the *Base band frequency* field and click *OK*.
3. In the Schematic window, choose *Design – Check and Save*.

Analysis Setup

1. Choose *Tools – Analog Environment* in the Schematic window.

The Simulation window appears.



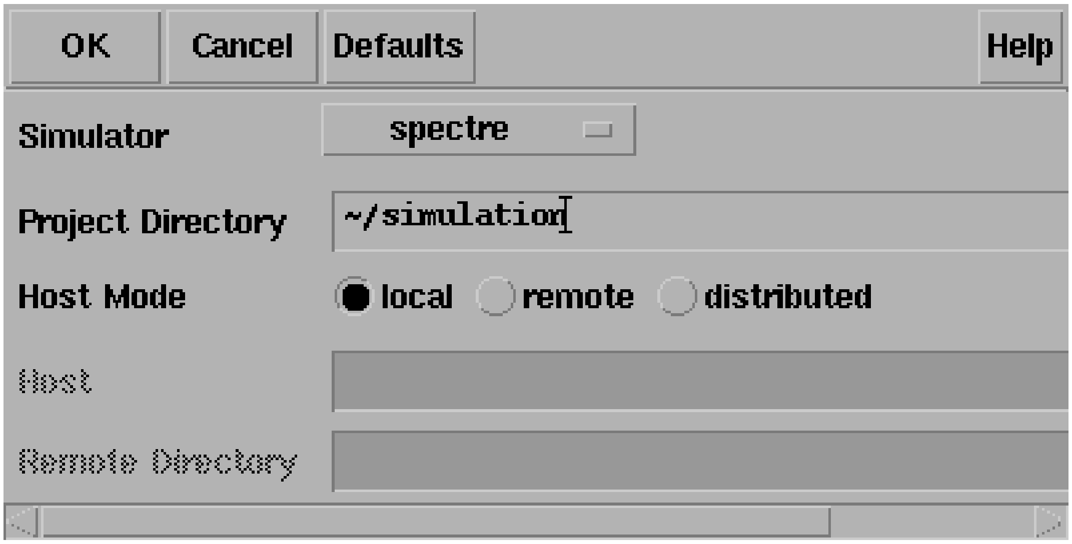
2. Choose *Setup – Simulator/Directory/Host* in the Simulation window.

   The Choosing Simulator/Directory/Host form appears.

In the Choosing Simulator/Directory/Host form, do the following:

1. Choose *spectre* for *Simulator*.
2. Type in the name of the project directory, if necessary.
3. Highlight *local.* (If you highlight *remote,* specify the *Host Mode*. For remote simulation, type in the name of the host machine and the remote directory in the appropriate fields.)

The completed form appears like the one below.

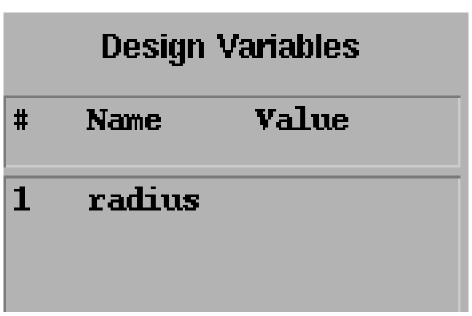


4. In the Choosing Simulator/Directory/Host form, click *OK.*

Copy and Edit Design Variables

1. In the Simulation window, choose *Variables – Copy From Cellview.*

   The variable *radius* appears in the *Design Variables* list box of the Simulation window.



2. Choose *Variables – Edit.*

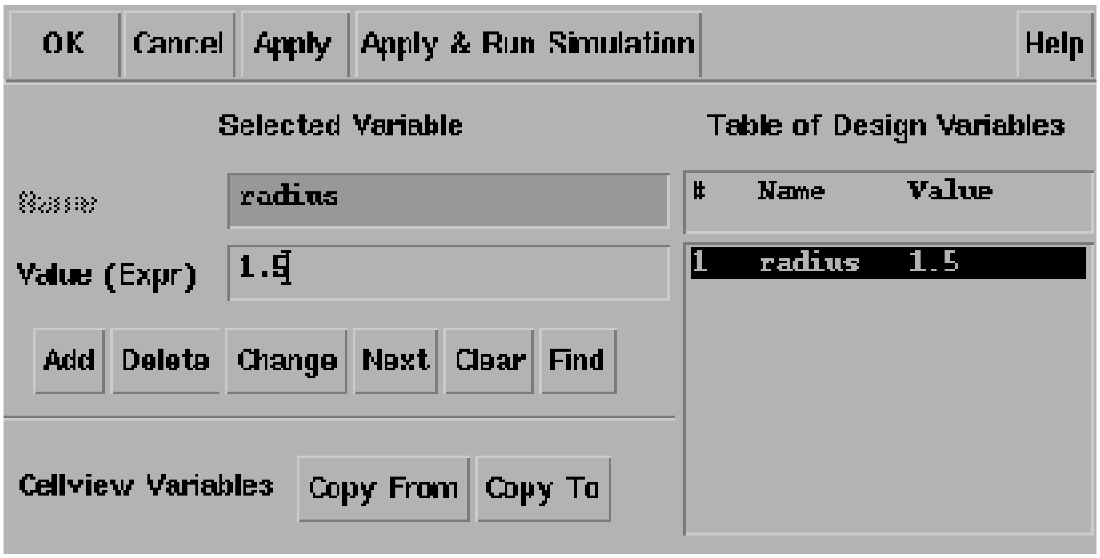
   The Editing Design Variables form appears.

In the Editing Design Variables form, do the following:

1. In the Table of Design Variables list box, click on *radius*.

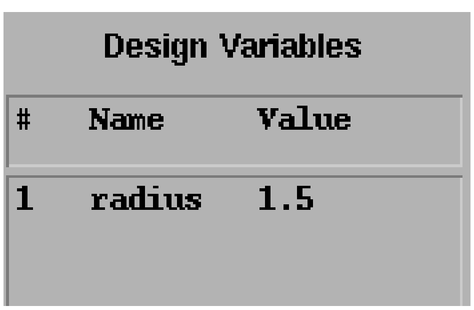
2. Type `1.5` in the *Value(Expr)* field.
3. Click *Apply*.

   The completed form looks like the one below:



4. Click *OK*.

   The new variable value displays in the Simulation window.



Setting Up the Analyses

1. Choose *Analyses – Choose* in the Simulation window.

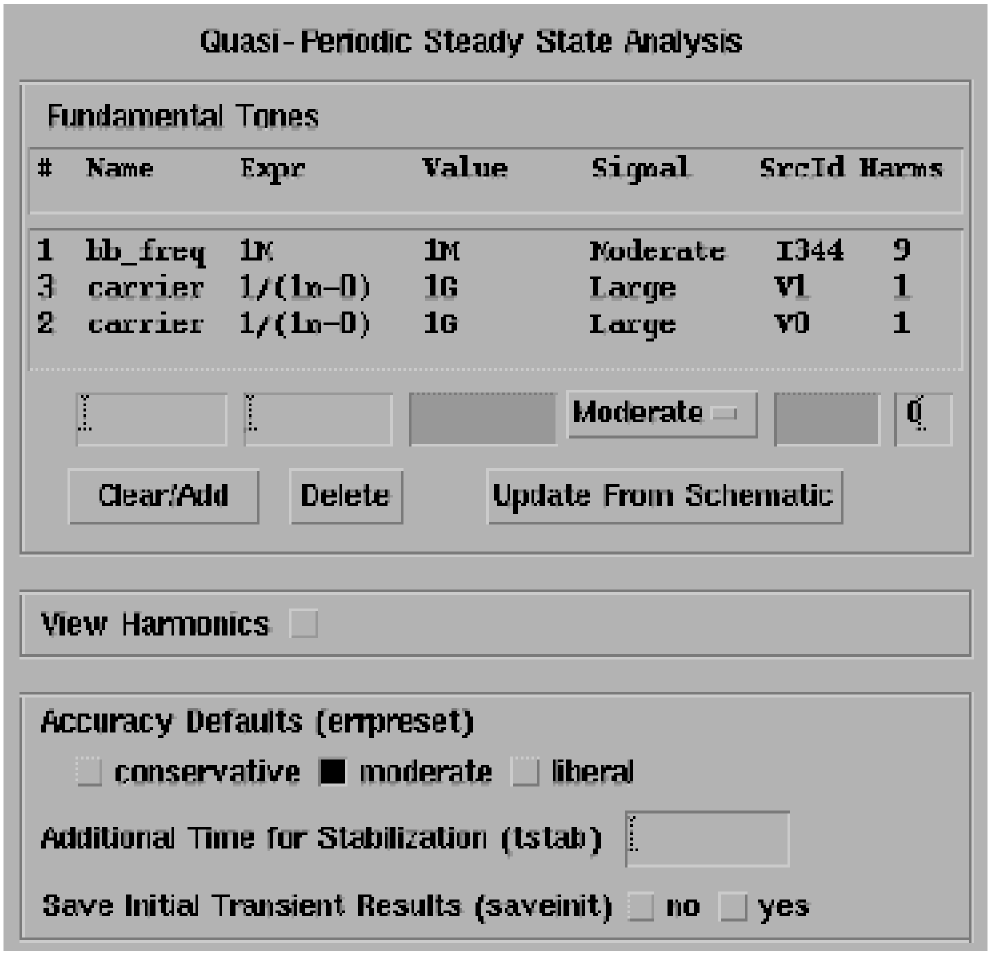
   The Choosing Analyses form appears.
2. In the Choosing Analyses form, click on *qpss* for the analysis.

The form changes to display options needed for QPSS. In the Choosing Analyses form, do the following:

3. In the *Fundamental Tones* list box, choose one of the tones labeled *carrier*.
4. In the fields immediately below the *Fundamental Tones* list box, do the following:
    a. Choose *Large* in the *Signal* cyclic field.
    b. Type 1 for the *Harms* field value.
    c. Choose *bb_freq*.
    d. Type 9 for the *Harms* field value for *bb_freq*.

    Since this is a down converting situation, a *Harms* value of 1 for the *Large* tone is sufficient here. For the *Moderate* tone, the higher *Harms* value of 9 guarantees higher order intermodulation terms. However, for *Moderate* tones, increasing the *Harms* value increases the simulation run time.
5. Highlight *moderate* for the *Accuracy Defaults (errpreset)* value.

The top of the QPSS form looks as follows.



6. Highlight *sweep*.

The form expands to let you specify sweep data. The defaults are *Variable* in the *Sweep* cyclic field and *no* for *Frequency Variable?*.

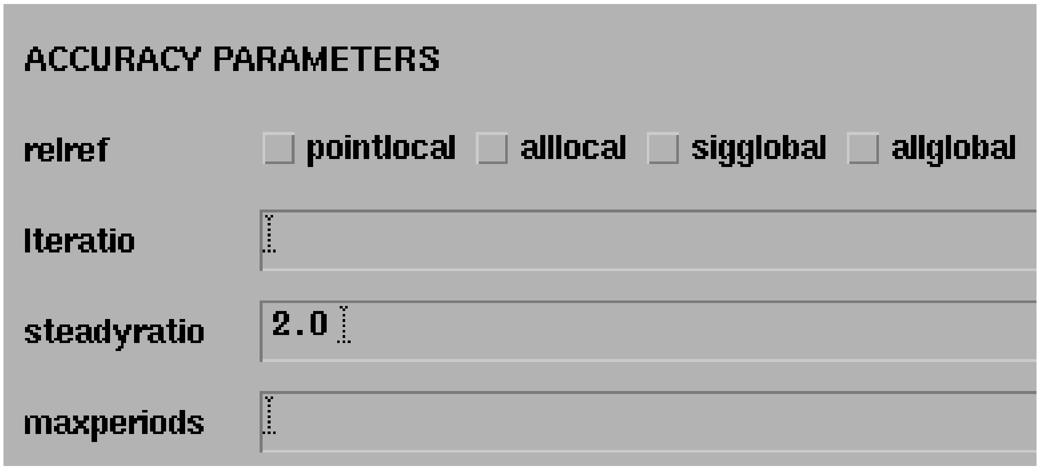
7. Click on *Select Design Variable*.

The Select Design Variable form appears.

8. In the Select Design Variable form, highlight *radius* and click *OK*.
9. In the QPSS form, highlight *Start – Stop* for *Sweep Range*.
10. Type `.001` in the *Start* field and `1.5` in the *Stop* field.
11. Highlight *Linear* for *Sweep Type*, and highlight *Number of Steps*. Type `20` in the *Number of Steps* field.
12. Choose *Options*.

    The QPSS Options form appears.
13. In the *Accuracy Parameters* section of the QPSS Options form, type `2.0` in the *steadyratio* field.

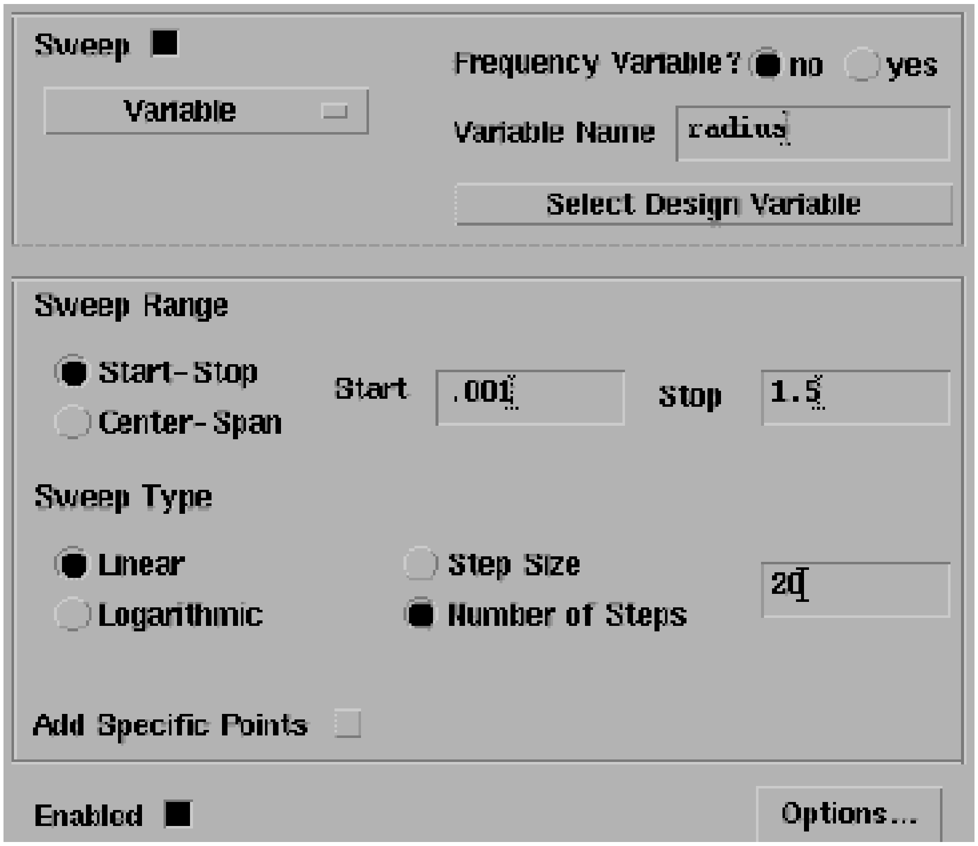
    Note: This circuit can cause convergence problems because the behavioral amplifiers have hard saturation curves. The `steadyratio` of 2 should work on all platforms but if you still experience convergence problems at any point in the QPSS sweep, increase `steadyratio` and decrease `reltol` by the same factor. For example, if you double `steadyratio`, half `reltol`.



14. In the QPSS Options form, click *OK*.

15. Be sure *Enabled* is highlighted.

The bottom of the QPSS form looks as follows.



16. In the QPSS Choosing Analyses form, click *OK*.

Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

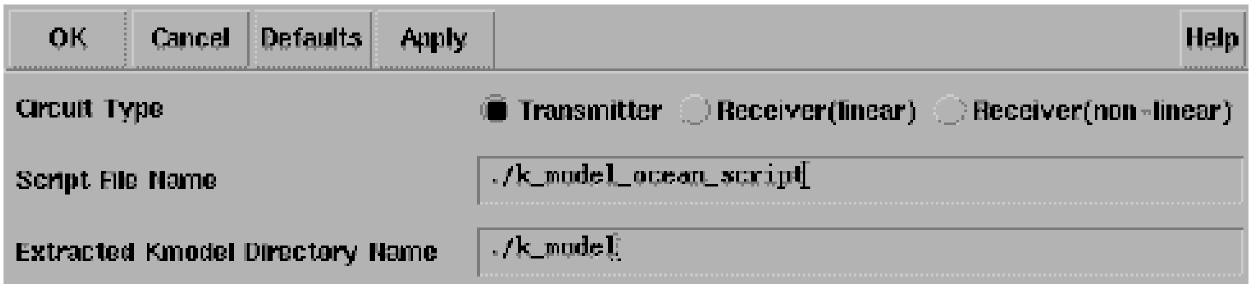
   The output log file appears and displays information about the simulation as it runs.

2. Check the CIW for a message saying the simulation completed successfully.

Creating the J-Model

1. In the Simulation window, choose *Tools—RF—Link to SPW*.

The RFIC Modeler for SPW form appears.

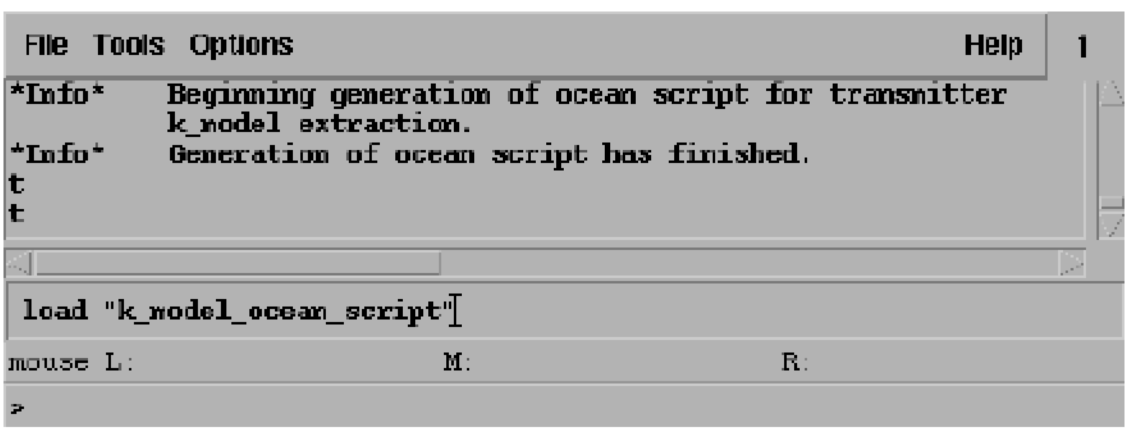


2. Highlight *Transmitter* for the *Circuit Type.*
3. Edit the *Script File Name* and the *Extracted Kmodel Directory Name*, if necessary.

   The file name is the name of the Ocean script to run later, and the directory name is the output directory.
4. Click *OK*.

   In the CIW, watch for a message that the Ocean script is generated.
5. In the CIW, load the Ocean script as follows. Remember to put the script name in double quotes.

   ```
   load "k_model_ocean_script"
   ```
6. Press *Return*.



You can follow the progress of the script in the CIW. When it finishes, the j-model is created in the directory you specified.

Using the J-model in a Circuit

In this section you create a circuit that contains the j-model you created and run an appropriate simulation.

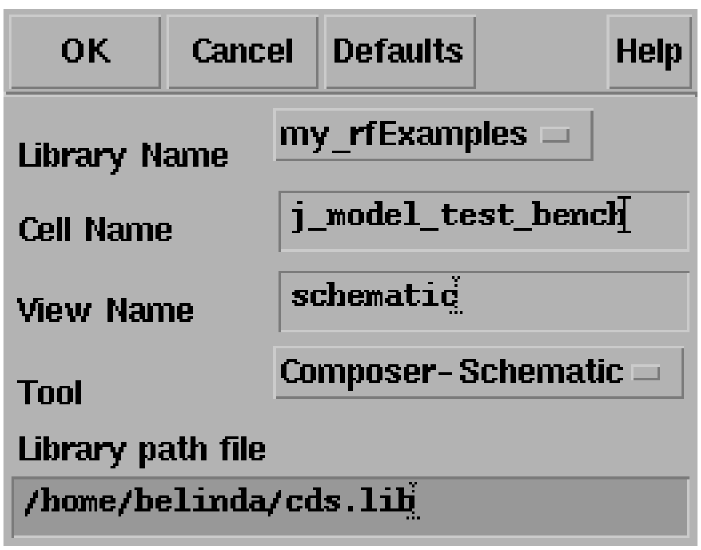
Setting up the Circuit for Simulation

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

2. Choose a library name from the *Library Name* cyclic field.
3. Type a name for the *Cell Name*.
4. Type a name for the *View Name*.
5. Click *OK*.

   The completed form looks like this.



   An empty Schematic window appears.

6. In the new Schematic window, choose *Add – Instance*.

   The Add Instance form appears.

7. In the Add Instance form, click on *Browse*.

   The Library Browser – Add Instance form appears.

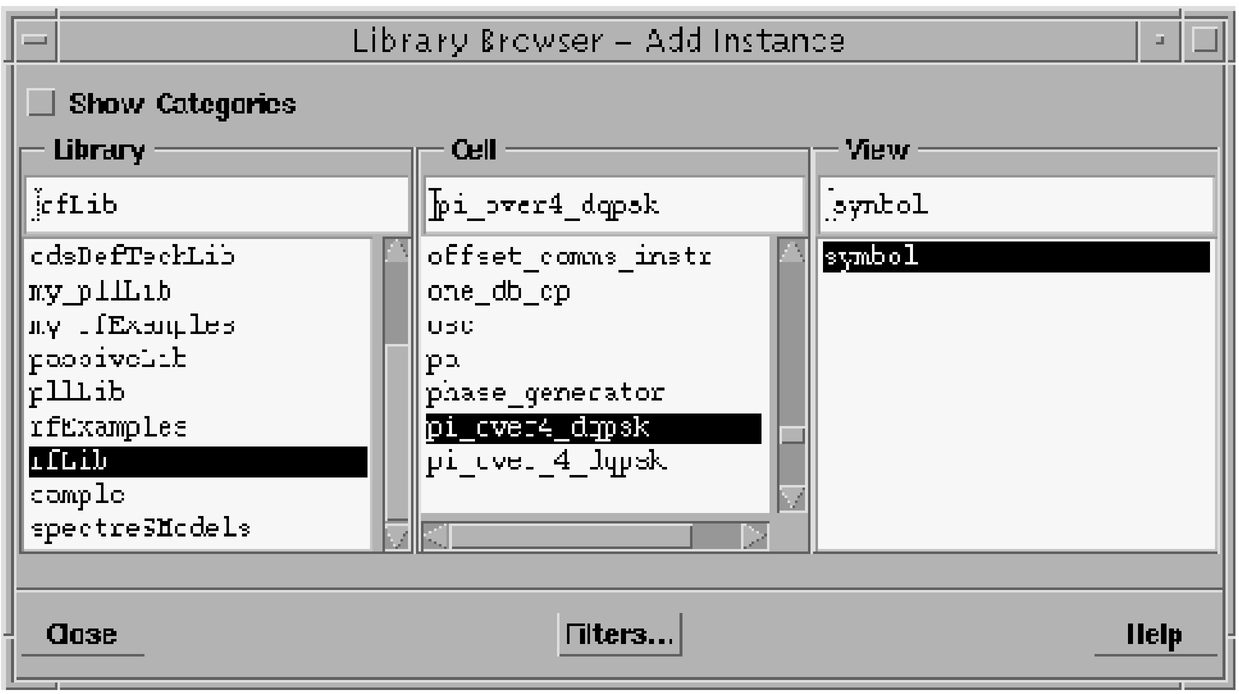
8. In the Library Browser – Add Instance form, do the following:

   a. Choose *rfLib* for *Library*.

   b. Choose *pi_over4_dqpsk* for *Cell*.

   c. Choose *symbol* for *View*.

   The completed form looks like this



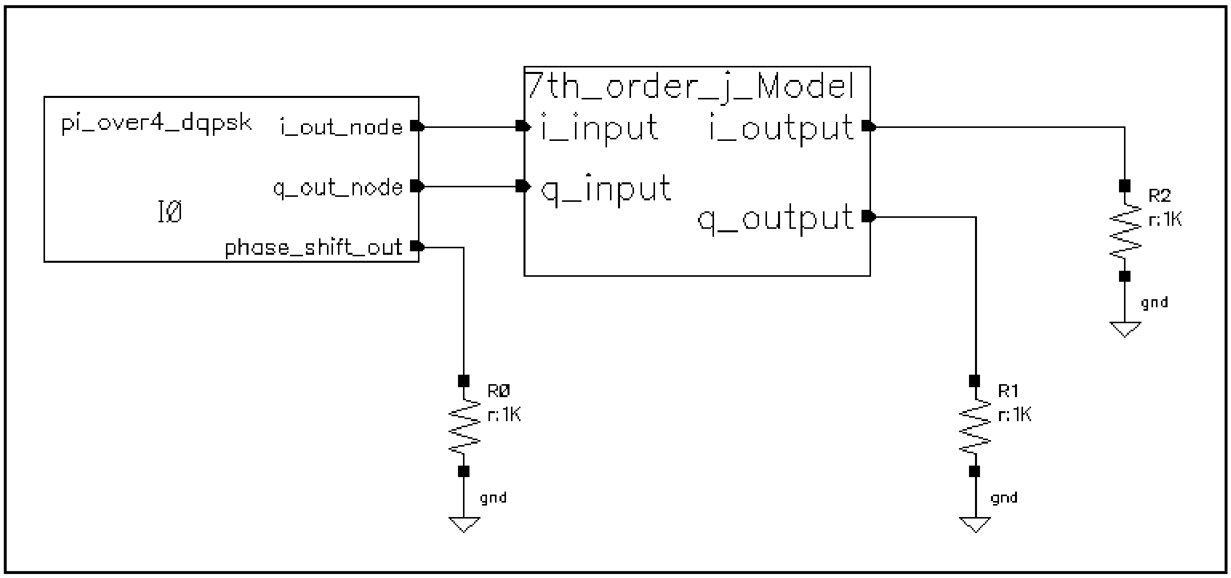
9. Place the cursorat the left end of the Schematic window and left click to place an instance of the *pi_over4_dqpsk* in the schematic. Press *Esc* after placing the instance.

   The appropriate instance is attached to the cursor when you place the cursor in the schematic. If you need assistance for working in the Schematic window, see Virtuoso® Schematic Composer™ User Guide.

10. Repeat the procedure you used to place the *pi_over4_dqpsk* instance in the schematic to add a *7th_order_j_model* instance from the *rfLib* to the right of the *pi_over4_dqpsk* along with three resistors and three grounds attached to the resistors. You create multiple copies of an instance with repeated left mouse clicks. The table below tells you the *Library*, *Cell*, and *View* values to choose.

Library	**Cell**	**View**
rfLib	7th_order_j_model	symbol
analogLib	res	symbol
analogLib	gnd	symbol

11. In the Schematic window, choose *Add – Wire (narrow)*. Click on beginning and ending locations in the schematic to add wires so the final schematic looks like the one below. Press *Esc* after you finish adding the wires.



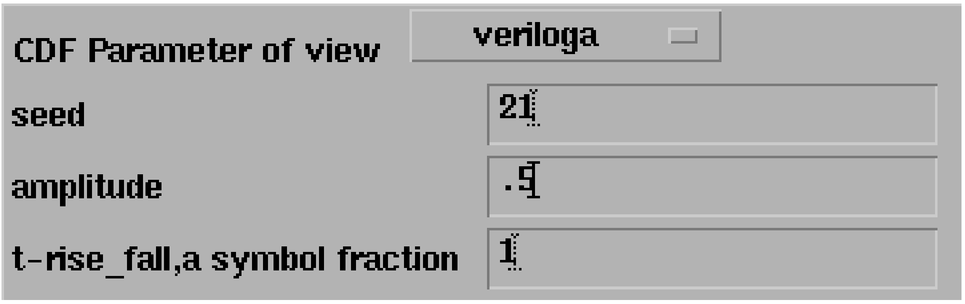
12. In the Schematic window, highlight the *pi_over4_dqpsk* instance and choose *Edit – Properties – Objects*.

Specify the following in the Edit Object Properties form:

a. Choose *veriloga* for *CDF Parameter of view.*

b. Type `.5` for *amplitude*.

The completed CDF parameter edits look like this



c. Click *OK*.

13. In the Schematic window, highlight the *7th_order_j_model* instance, and repeat the previous steps to bring up the Edit Object Properties form. Edit the form as follows:

    a. Choose *ahdl* for *DCF Parameter of view*.

    b. Type the full path to the j_model for the *data_directory* value.

    The completed CDF parameter section of the form looks like this.

    c. Click *OK*.

14. In the Schematic window, choose *Design – Check and Save*.

15. In the Schematic window, choose *Tools – Analog Environment*.

    The Simulation window appears.

16. In the Simulation window, choose *Analyses – Choose*.

    The Choosing Analyses form appears.

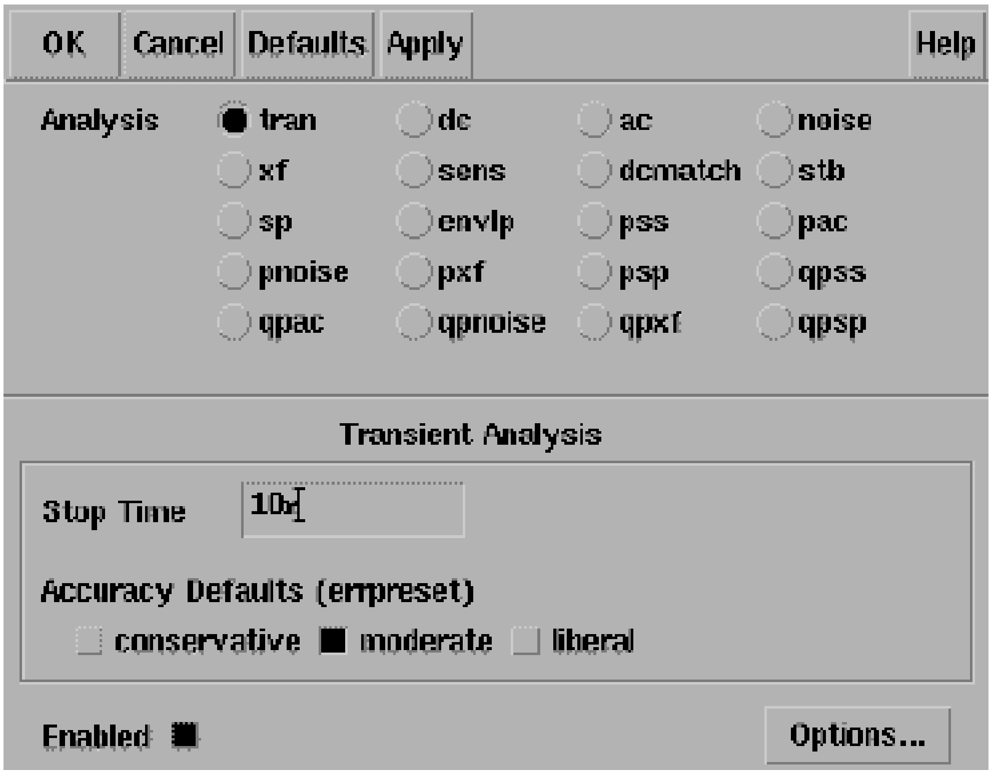
17. Specify the following in the Choosing Analyses form:

    a. Type `10m` for the *Stop* time.

    b. Highlight *moderate* for *Accuracy Defaults (errpreset)*.

You can also leave this value unselected because *moderate* is the default.

c. Be sure the *Enabled* button is highlighted.



d. Click *OK*.

18. In the Simulation window, choose *Simulation – Netlist and Run* to run the simulation.

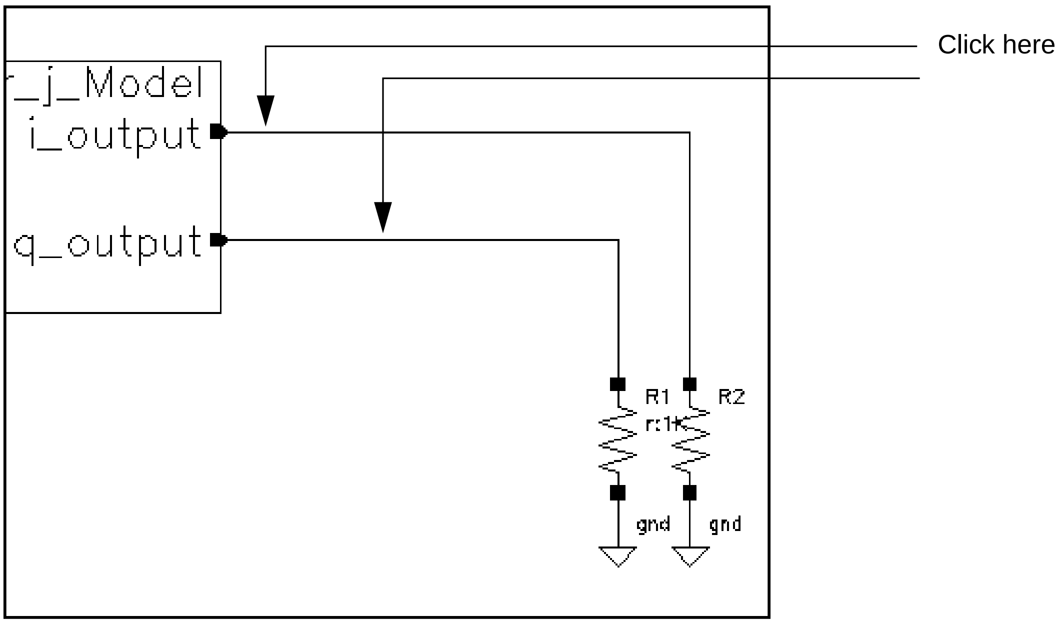
Check the CIW for a message saying the simulation completed successfully.

Working with the Simulation Results

1. In the Simulation window, choose *Results – Direct Plot – Transient Signal*.

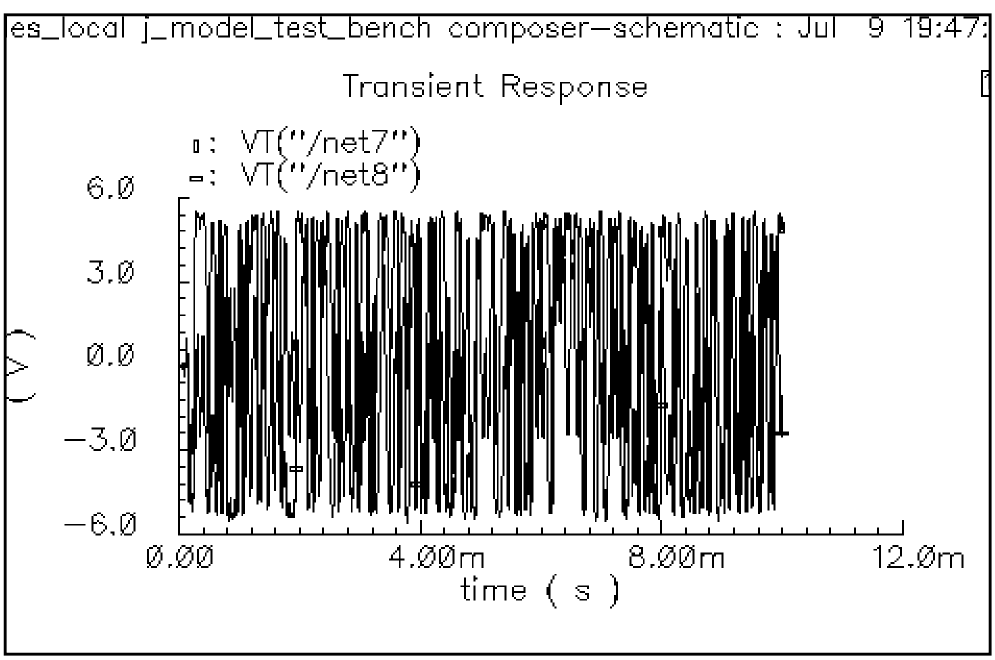
An empty Waveform window appears.

2. In the Schematic window, click on the *i_output* net and then on the *q_output* net



3. Press the *Esc* key.

The plots for *i_output* and *q_output* appear in the Waveform window.



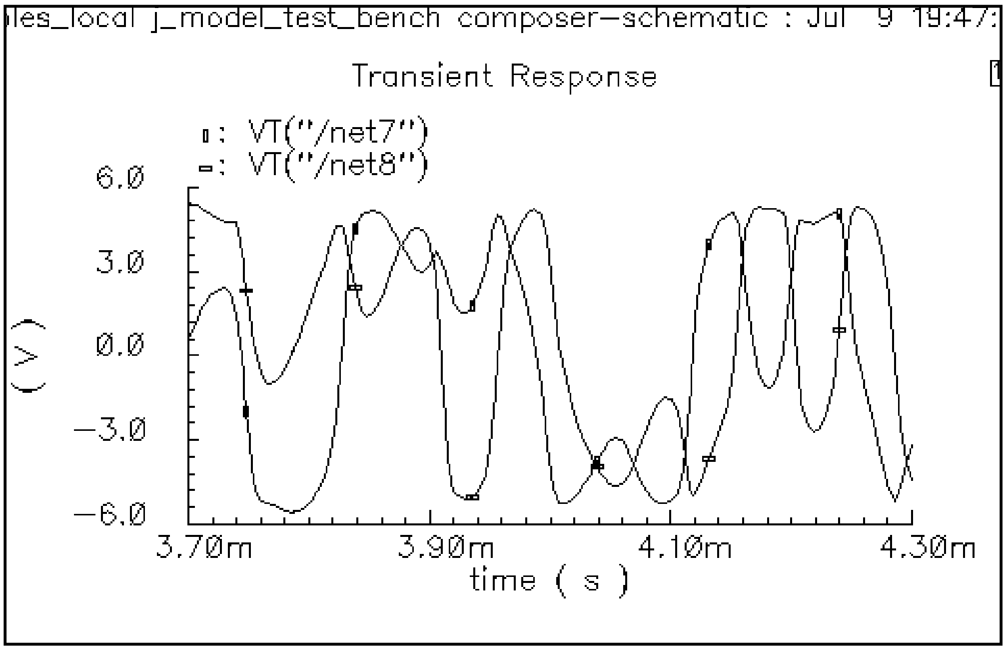
4. Highlight a small portion of the waveform by performing the following steps:

a. In the Waveform window, choose *Zoom – Zoom In*.

b. Left click and drag with the mouse to form a rectangle that includes the waveform over a small time value.

c. Left click to complete the selection.

The waveform display changes to show the portion you select with the mouse.



5. In the Simulation window, choose *Tools – Calculator*.

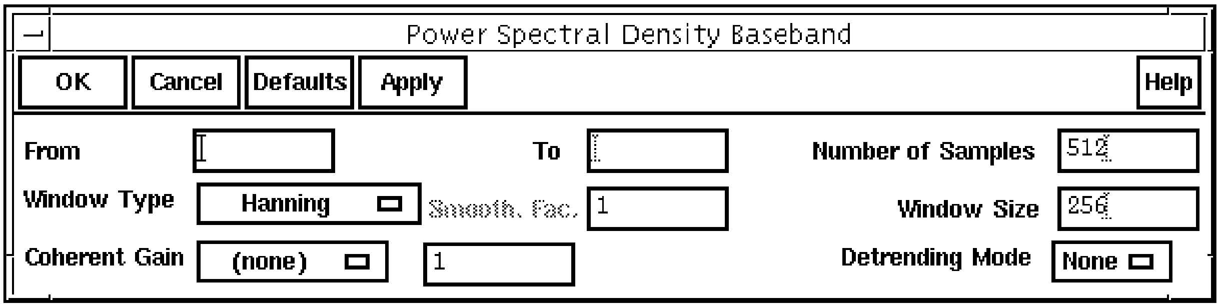
   The Waveform Calculator appears.

6. In the Waveform Calculator, click on the *wave* key.

7. In the Waveform window, click on each of the two waves that are displayed.

   In the Waveform Calculator, choose *psdbb…* from the *Special Functions* cyclic field. The psdbb function is documented in the Waveform Calculator section.

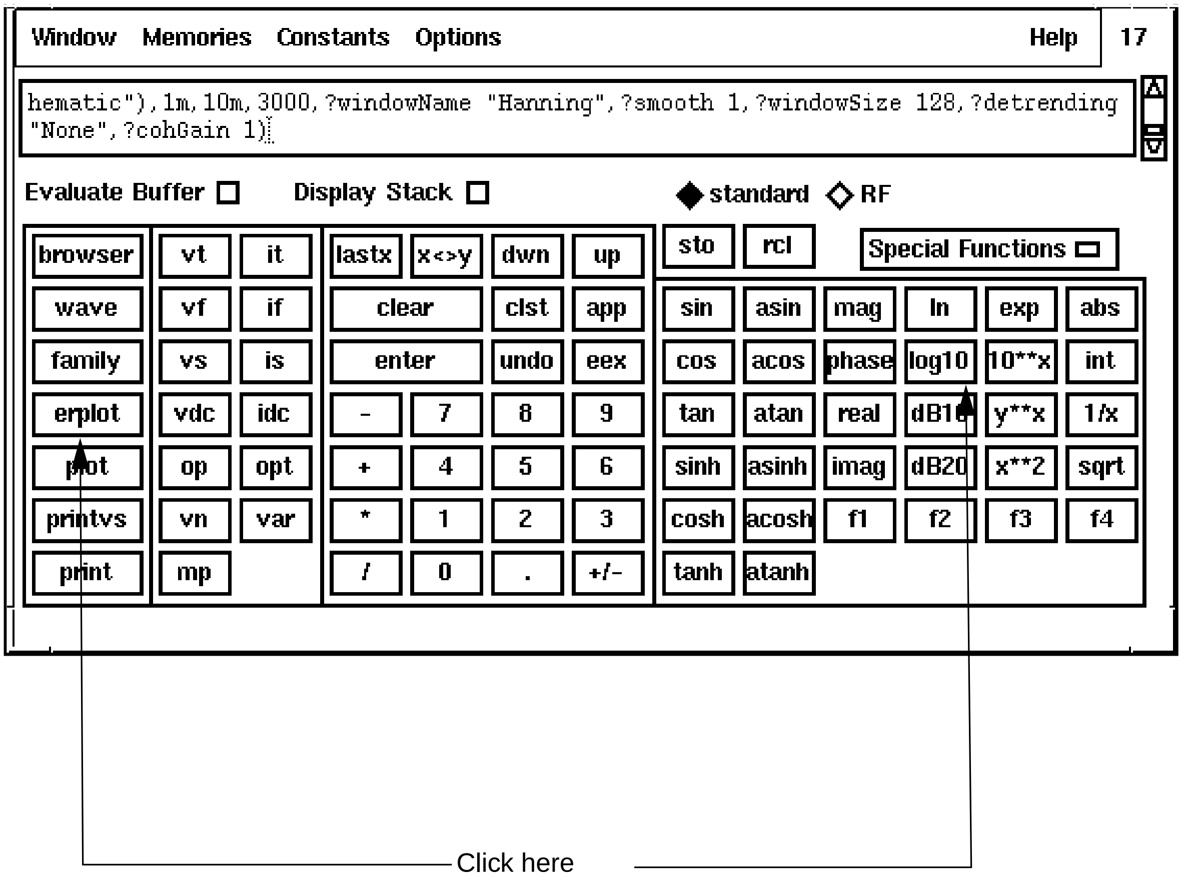
The Power Spectral Density Baseband form appears.



8. In the Power Spectral Density Baseband form, do the following:

- **a.** Type `1m` in the *From* field.
- **b.** Type `10m` in the *To* field.
- **c.** Type `3000` for the *Number of Samples*.
- **d.** Type `128` for the *Window Size*.
- **e.** Click *OK*.

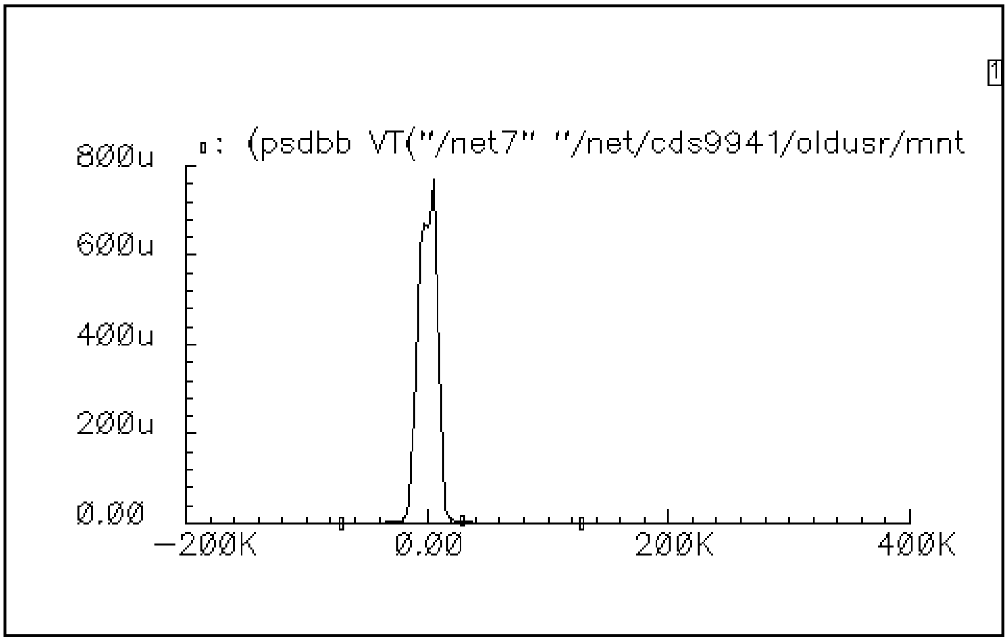
The Waveform Calculator now looks like this.



9. In the Waveform window, choose *Window – Reset*.

10. In the Waveform Calculator, click on *Plot*.

The plot for *psdbb* appears in the Waveform window.



11. In the Waveform window, click on *Curves – Edit*.

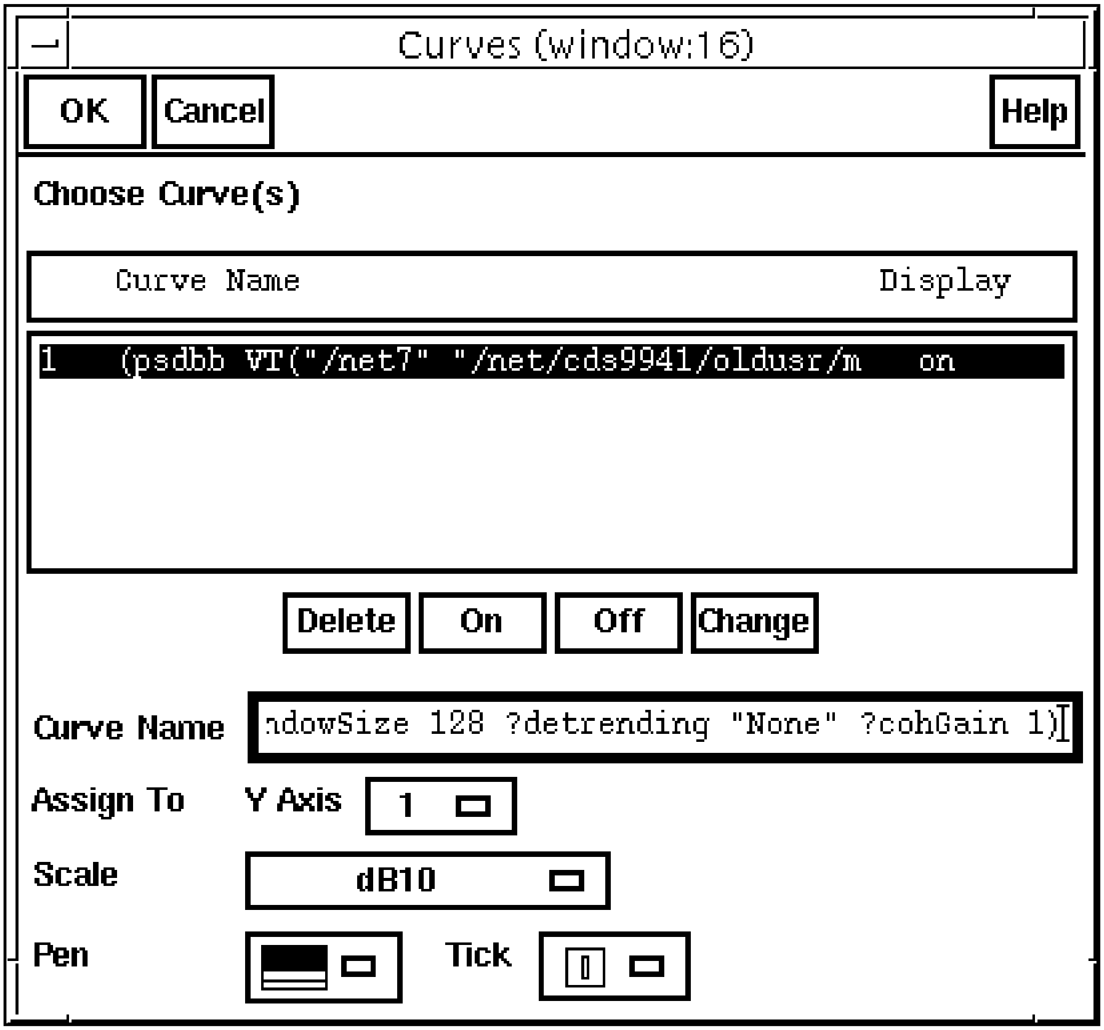
The Curves window appears.

12. In the Curves window, do the following:

a. Click on the value in the *Choose Curve(s)* list box.

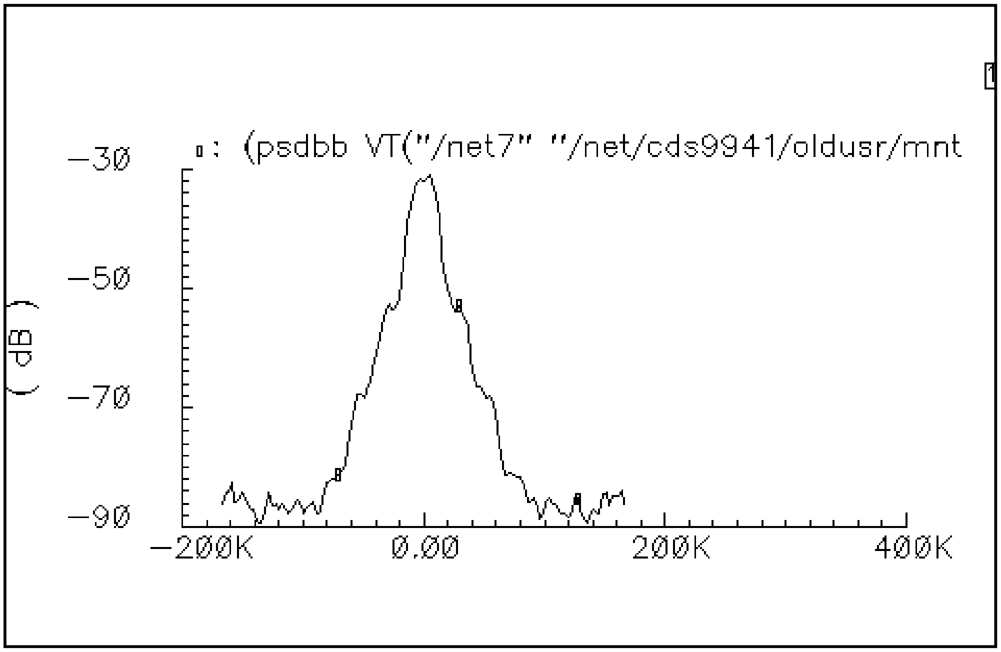
b. Choose db10 for *Scale*.

The Choose Curve(s) list box looks like this.



c. Click *OK*.

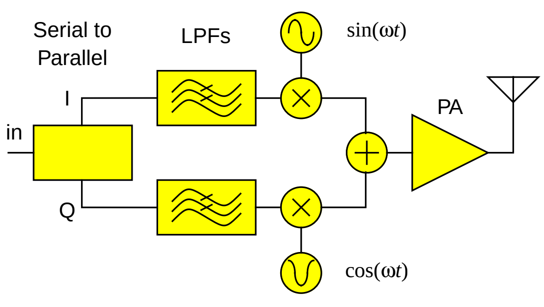
The plot changes to reflect your new specifications. ACPR is computed by taking the difference in db between the power spectral density measured at the center frequency and at the adjacent channel.



More About the J-model

Figure 9-2 shows a typical direct-conversion digital transmitter architecture.

Figure 9-2 Direct Conversion Transmitter



The low-pass filters can eliminate out-of-band signal components from the input baseband signal. However, these components might reappear because of intermodulation distortion generated within the transmitter, such as in the downstream power amplifier. This return of unwanted signal components is called “spectral regrowth.”

Adjacent Channel Power Ratio (ACPR) is a popular measure of spectral regrowth in digital transmitters and is often a design specification. To get the frequency resolution required for accurate estimation of ACPR, the simulation often must process a random sequence of between 500 and 10k input symbols.

Envelope Following simulates the transmitter efficiently, but it must be driven by deterministic or unfiltered random signals, and a simulation of 10k signals is lengthy.

Most filters, like those in Figure 1, are FIR filters. SpectreRF does not work with FIR filters (implemented in AHDL) because they contain hidden state. To drive an Envelope Following analysis with realistic random baseband signals, you must create the signals in a separate simulation, store them to a file, then read the data into the Envelope Following analysis with piece-wise linear sources. However, the QPSS analysis can perform a fast, indirect ACPR estimation that can simulate 10k symbols within minutes after you extract the model. Extraction time is independent of the number of symbols you want to simulate.

Initially, QPSS extracts a behavioral baseband-equivalent model of the transmitter. In transmitter circuits, the input baseband signal is usually well within the transmitter’s bandwidth, so a model without memory, such as the J-model, is often sufficient. Because the behavioral model eliminates the carrier and unnecessary circuit details, the subsequent ACPR calculation step is fast, even for large or complex circuits.

The most familiar spectral regrowth mechanism is the intermodulation distortion of amplitude modulation (AM) signal components that enter the power amplifier. Modulation schemes that carry information only on the carrier phase or frequency try to minimize spectral regrowth by eliminating this AM component. However, the digital baseband filters can still convert discontinuous phase changes into amplitude transients in the composite RF signal. Models based on AM/AM and AM/PM conversion [2] capture most such power amplifier related distortion mechanisms. However, imperfections in the *I/Q* modulators can convert input phase variations into output amplitude and phase variations that also contribute to the distortion. This is particularly true for GSM transmitters. For this reason, we also include PM/AM and PM/PM conversion effects in our model. Formally, the J-model is a narrowband approximation to a vector-valued, periodically time-varying, functional series expansion [3].

To understand the extraction procedure, write the inputs in polar coordinates, such as magnitude ρ and phase ϕ. Consider the mapping $f(\rho,\phi)$ from baseband input to the baseband representation of the RF output. That is, if

$$input = \rho\cos(\phi) + j\rho\sin(\phi)$$

$$output = Re[f(\rho, \phi)]\cos(\omega_c t) - Im[f(\rho, \phi)]\sin(\omega_c t) = Re[f(\rho, \phi)exp(j\omega_c t)] \quad .$$

The baseband representation of the output is $f(\rho,\phi)$. The RF carrier frequency is ω_c. Because we assume the system is memoryless with respect to baseband signals, and that we have only one frequency translation, we can assume $f(\rho,\phi)$ is periodic in ϕ and can be expressed as a Fourier series,

$$f(\rho, \phi) = \sum_k A_k(\rho)e^{jk\phi} \quad ,$$

where the magnitude-dependent Fourier coefficients are

$$A_k(\rho) = \frac{1}{2\pi}\int_0^{2\pi} f(\rho, \phi)e^{-jk\phi}d\phi \quad .$$

To extract the Fourier coefficients, the *I* and *Q* inputs of the transmitter are driven with sinusoids in quadrature, at a frequency within the transmitter's bandwidth. For these circular input trajectories, $\phi = \omega_0 t$, and the Fourier coefficients are given by

$$A_k(\rho) = \frac{\omega_0}{2\pi}\int_0^{2\pi/\omega_0} f(\rho, \omega_0 t)e^{-jk\omega_0 t}dt \quad .$$

Therefore, for a given input magnitude, the Fourier coefficients are obtained directly from the output spectrum calculated by QPSS. For example, if the input circle is large enough to alternately saturate the modulators, the –3 harmonic of the complex baseband tone dominates the distortion.The output spectrum has lines at $\omega_c+\omega_0$ and $\omega_c - 3\omega_0$. QPSS computes the real and imaginary parts of the Fourier coefficients associated with those lines. The simulation is repeated for a range of input magnitudes to capture the magnitude dependence of the Fourier coefficients. At each input magnitude, the fundamental and its relevant harmonics are recorded for interpolation. The model implementation reads the Fourier coefficients and then processes any amount of input baseband data according to $f(\rho,\phi)$.

Figure 9-3 on page 673 shows the model extraction process for the fundamental and (-3) terms in the expansion $f(\rho,\phi)$. The extraction script records all harmonics up to 9th order, including even-numbered harmonics. Figure 9-4 on page 674 shows a block diagram of the associated model for just the fundamental and -3 harmonic.

Figure 9-3 Model Extraction

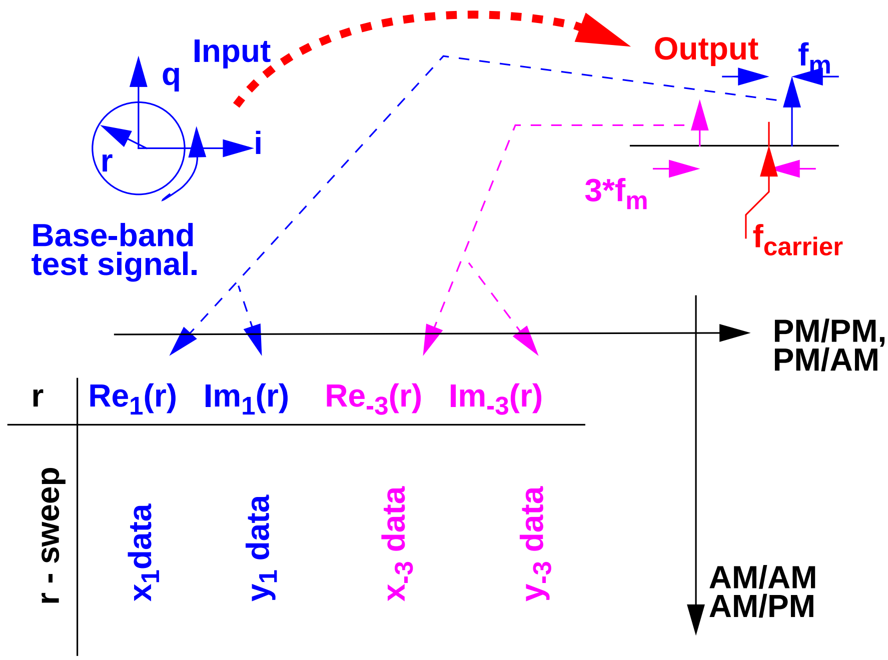
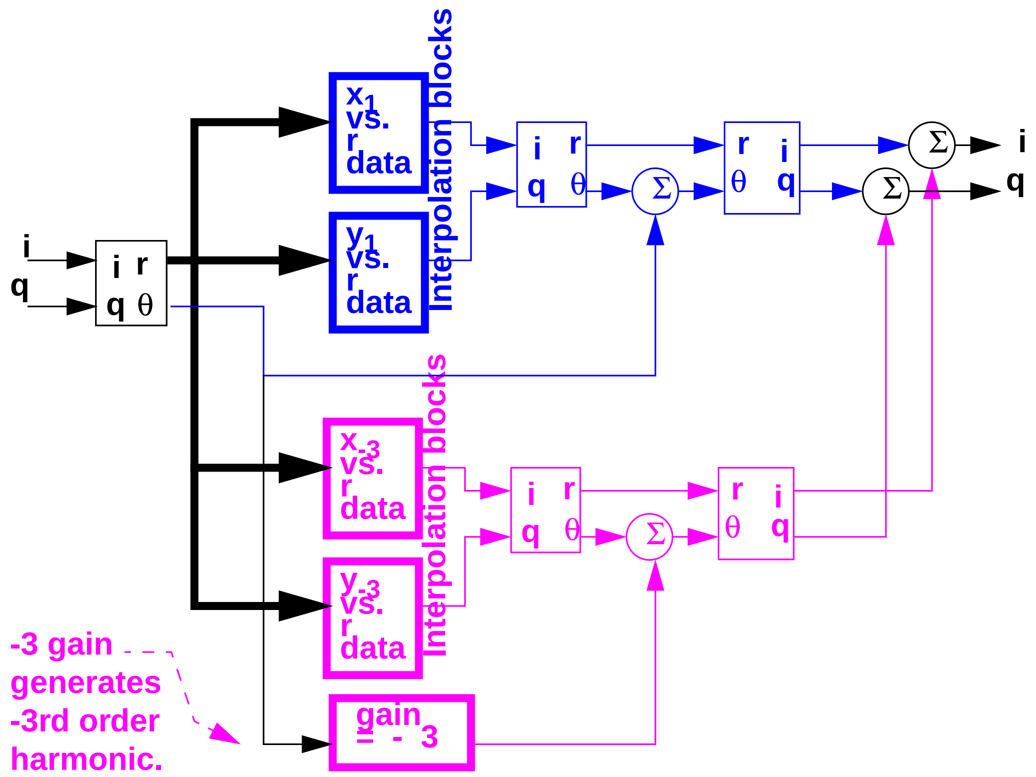
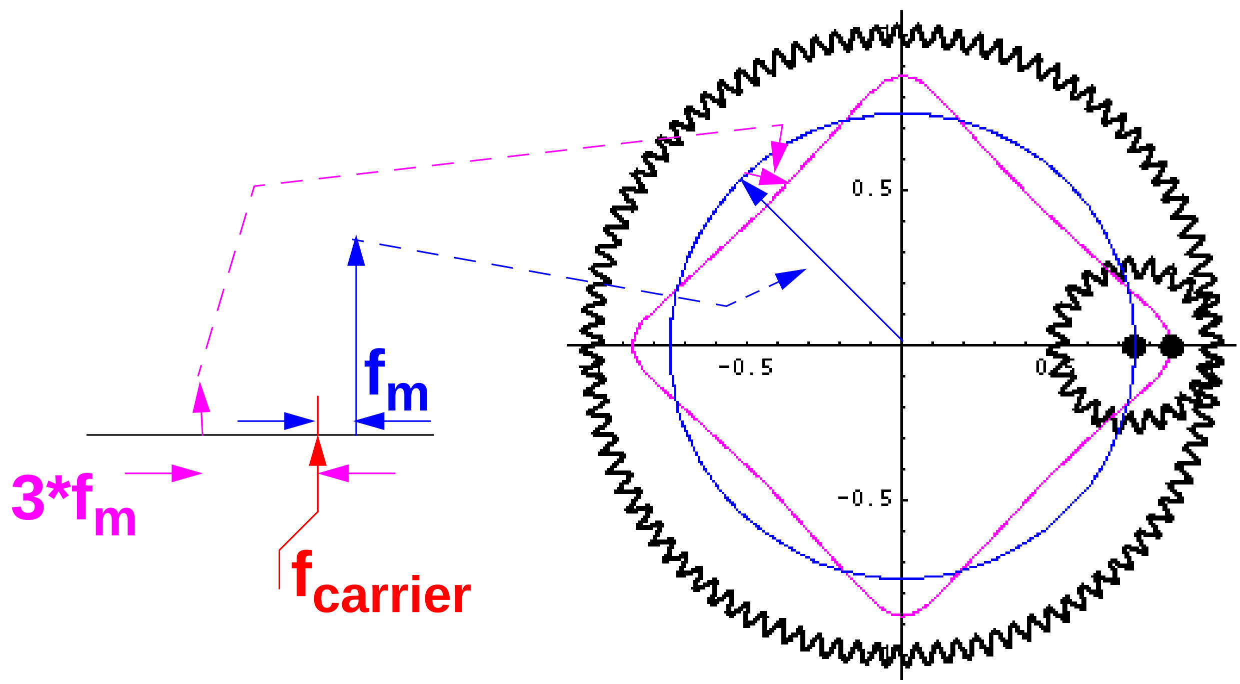


Figure 9-4 Block Diagram



You can visualize PM/AM conversion, PM/PM conversion, and the new macromodel with the help of the Spirograph™ plot shown in Figure 9-5 on page 675. Consider just one distortion harmonic, the minus three harmonic, in the baseband output that is generated when the input trajectory is circular. There are no baseband harmonics generated by AM/AM or AM/PM conversion because the input radius is constant. The fundamental term. The ideal term, in the power amplifier output is represented by a vector from the center of the large inverted gear to the center of the small non-inverted gear. Since the input trajectory is circular, the fundamental vector rotates about the origin at the same frequency the input trajectory rotates about its origin. That is exactly the trajectory traced out by a pen placed in a hole at the center of the small gear as the small gear rolls around the inside of the inverted gear. If we place the pen in a hole **offset** from the center of the small gear, the vector from the center of the small gear to the hole represents the $-3\omega_0$ distortion term. The small gear rotates -3 times for each complete trip around the inverted gear. This is precisely what the model does for each recorded harmonic. The model determines the radii of both gears, as well as the phase offsets, from the recorded QPSS results. (Although not intuitive, the -3 frequency ratio really does produce a four-cornered trace, demonstrating that the square pattern caused by independently saturating I/Q modulators results from a minus third harmonic.) The gear analogy clearly illustrates PM/AM and PM/PM conversion. The offset hole distorts the otherwise circular trajectory with phase and amplitude perturbations that vary with the input phase.

Figure 9-5 Spirograph™ P lot



To measure run times we contrived a benchmark transmitter that included the up-conversion mixers and power amplifier. The benchmark circuit contained 46 transistors and generated 328 equations in the circuit simulator. A 7th order model took 4 hours to extract and simulated 30ms of CDMA data (40k chips @ 1.25Mchips/sec) in less than 21 minutes using a Sun Ultra 1.

Driving 1st order and 7th order models with CDMA and GSM baseband signals shows when PM/AM effects become important. The first order model only accounts for AM/AM and AM/PM conversion. The 7th order model includes PM/PM and PM/AM conversion as well AM/AM and AM/PM conversion. Figure 9-6 on page 676 shows output CDMA trajectories. Although the trajectory from the 7th order model is more square, the power spectral density (PSD) estimates are nearly identical, as Figure 9-7 on page 677 shows.

Figure 9-8 on page 677 and Figure 9-9 on page 678 show the output trajectories and power spectral densities when the models are driven by GSM signals of the same maximum amplitude. However, unlike the CDMA spectral results, the GSM spectral results differ significantly. Because the GSM input signal has no AM component, only the higher order models capture the true ACPR beyond about 150 Khz.

Figure 9-6 Output CDMA Trajectories

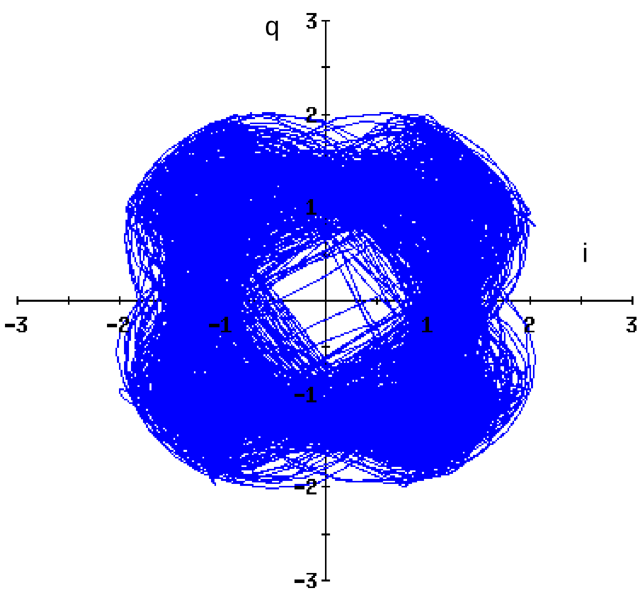


Figure 5a, 1st order output

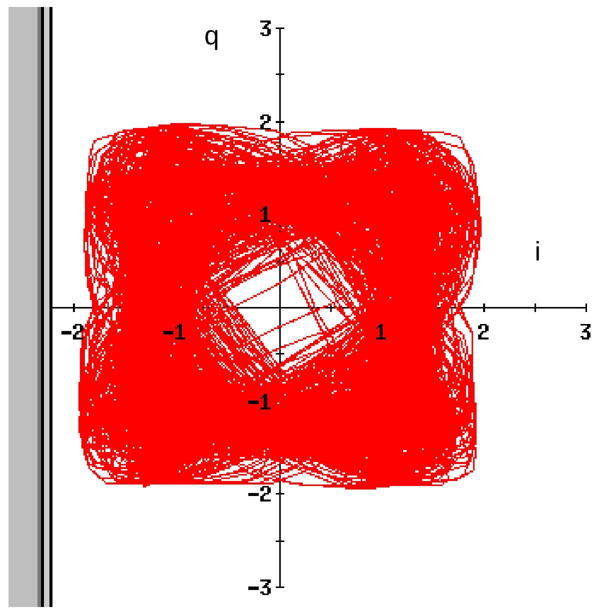


Figure 5b, second order output

Figure 9-7

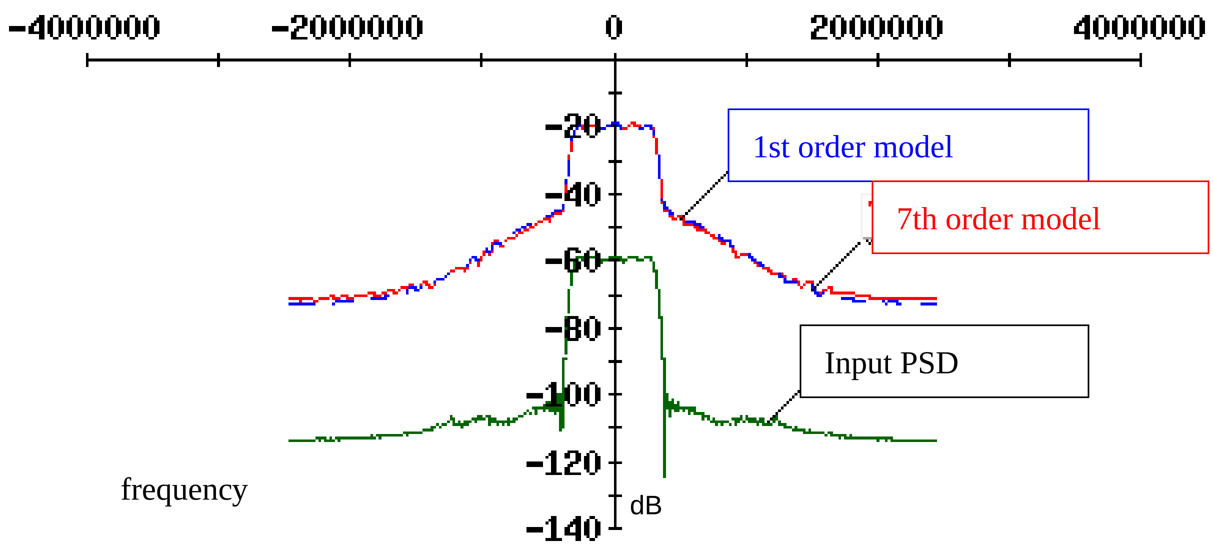


Figure 9-8 Trajectory for GSM

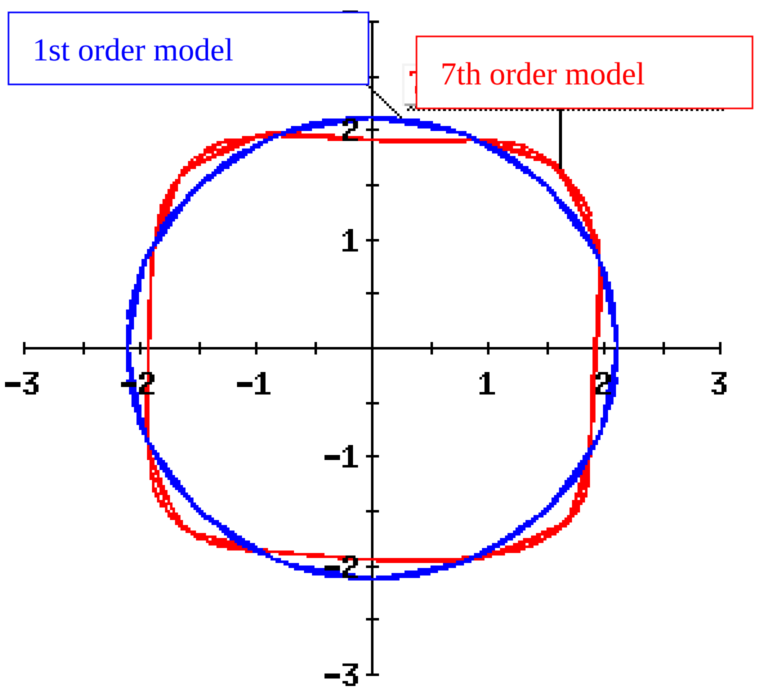
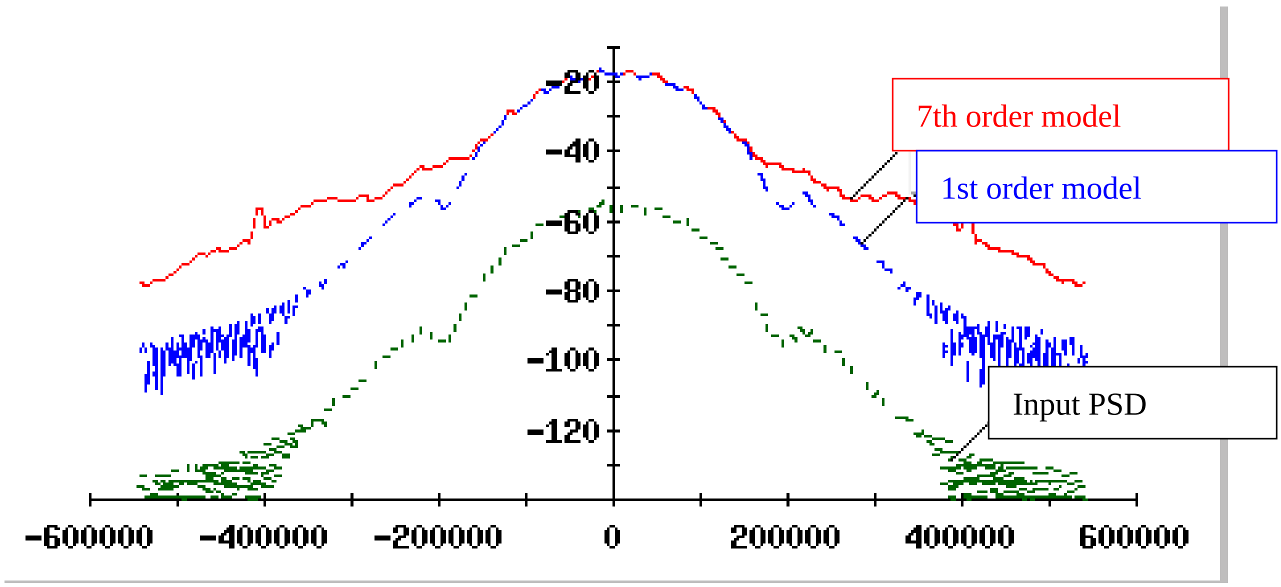


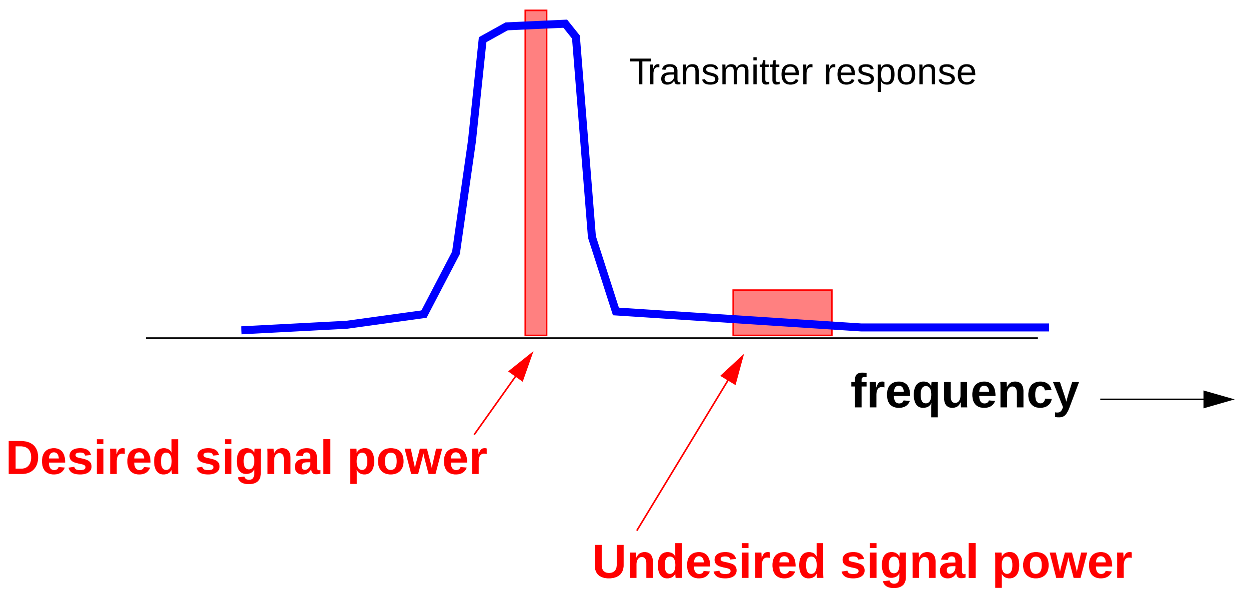
Figure 9-9 PSD for GSM



J Model Limitations

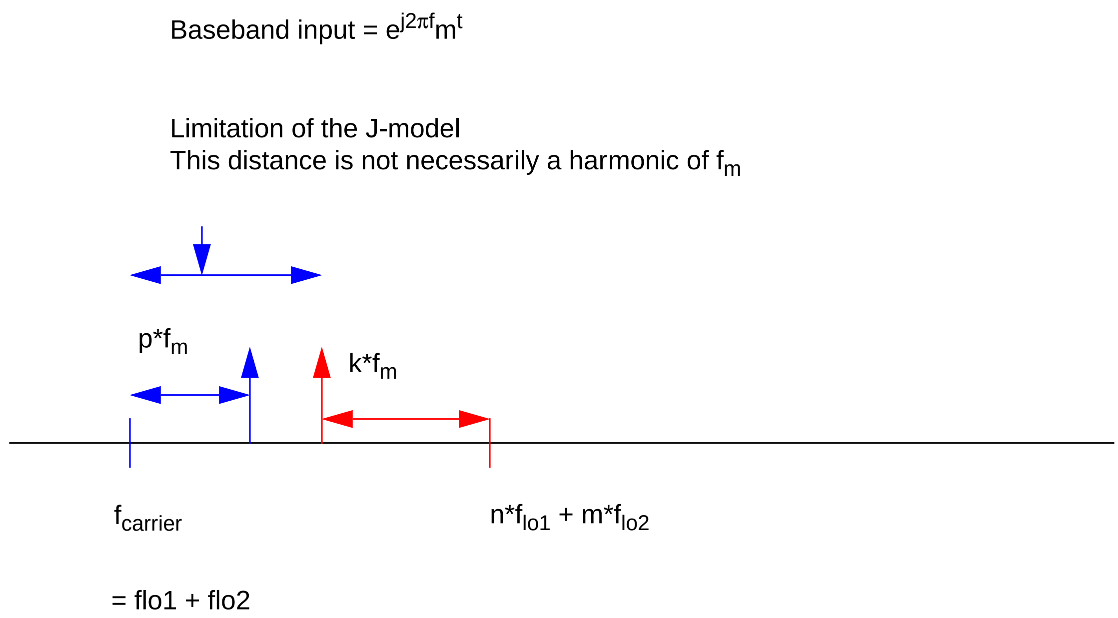
Like any behavioral model, the J-model has limitations. The J-model is a static model that has no memory. You can see this in the mechanical analogy. The pattern traced out by the pen in the small gear does not depend on how fast the pen and gear move. Figure 9-10 illustrates the limitation in the frequency domain. Let the thick curve be the transmitter's frequency response to baseband input. The rectangles show where signal power exists. The smaller rectangle of signal power in the picture below is generated within the transmitter. It is not part of the input signal. The input signal must be contained in the larger rectangle.As long as the transmitter's frequency response is flat over bands containing any significant signal power, the J-model is valid.

Figure 9-10 J Model Limitations in the Frequency Domain



Why is the J-model limited to direct conversion transmitters? Consider a double-conversion transmitter, one with an IF stage. With an IF stage, it might be possible for the output signal to contain power at some linear combination of the two local oscillator frequencies that falls near the output RF signal. Those frequencies are not displaced from the target RF carrier by some multiple of the input baseband frequencies. The figure below shows the problem. If you can ignore terms such as those in Figure 9-11 on page 680, you can use the J-model for a transmitter with more than one frequency translation.

Figure 9-11 J Model Limitations



J-model data files and data format

The J-model data files are listed below:

```
imag_minus1.ascsig
imag_minus2.ascsig
imag_minus3.ascsig
imag_minus4.ascsig
imag_minus5.ascsig
imag_minus6.ascsig
imag_minus7.ascsig
imag_minus8.ascsig
imag_minus9.ascsig
imag_plus1.ascsig
imag_plus2.ascsig
imag_plus3.ascsig
imag_plus4.ascsig
imag_plus5.ascsig
imag_plus6.ascsig
```

```
imag_plus7.ascsig
imag_plus8.ascsig
imag_plus9.ascsig
imag_zero.ascsig
real_minus1.ascsig
real_minus2.ascsig
real_minus3.ascsig
real_minus4.ascsig
real_minus5.ascsig
real_minus6.ascsig
real_minus7.ascsig
real_minus8.ascsig
real_minus9.ascsig
real_plus1.ascsig
real_plus2.ascsig
real_plus3.ascsig
real_plus4.ascsig
real_plus5.ascsig
real_plus6.ascsig
real_plus7.ascsig
real_plus8.ascsig
real_plus9.ascsig
real_zero.ascsig
```

The first word in the file name, up to the underscore, tells you whether the data file contains the real or imaginary part of the Fourier coefficient. The second part, following the underscore, specifies the signed harmonic number. The suffix identifies an ascii file for Alta SPW.

The format of all file is the same and is shown below. The format is ascii. Each file contains exactly three header lines and no blank lines at the end. The data is arranged in two columns. The first column is the input signal level in volts. The second column is the real or imaginary part of the baseband Fourier coefficient as described above.

sweep	**imag(harmonic)**
1.000e-03	-8.475e-07
2.450e-03	-1.366e-05
3.900e-03	-4.850e-05
5.350e-03	1.595e-05

6.800e-03	3.690e-05
8.250e-03	-7.153e-04
9.700e-03	-1.055e-04
1.115e-02	-4.324e-05
1.260e-02	-1.431e-04
1.405e-02	-1.143e-03
1.550e-02	-4.352e-03
1.695e-02	-5.443e-04
1.840e-02	1.107e-03
1.985e-02	1.913e-03
2.130e-02	3.774e-02
2.275e-02	8.256e-02
2.420e-02	1.277e-01
2.565e-02	1.680e-01
2.710e-02	2.114e-01
2.855e-02	2.474e-01
3.000e-02	2.841e-01

References

[1] E. Ngoya and R. Larcheveque. Envelope transient analysis: a new method for the transient and steady state analysis of microwave communication circuits and systems. *IEEE MTT Symposium Digest*, pp. 1365-1368, June 1996.

[2] M. Jeruchim, P. Balaban and K. Shanmugan, *Simulation of Communication Systems*, Plenum, 1992.

[3] M. Schetzen, “*The Volterra & Wiener Theories of Non-Linear Systems*”, Krieger Publishing Company, 1980.

10

Modeling Transmitters

This chapter tells you how to use several SpectreRF transmitter design features. It emphasizes

- Setting Up an Envelope Following analysis (Envlp)

  The Envelope Following analysis example explains

  - How to interpret Envlp analysis results
  - How to visually detect distortion quickly

- Measuring ACPR and PSD. PSD (Transmitted Power Spectral Density) is characterized by the ACPR number (Adjacent Channel Power Ratio).

  The ACPR and PSD example explains

  - How to use the ACPR wizard.
  - How to estimate PSD.

- Measuring Load Pull Contours and Reflection Coefficients

  The Load-Pull example explains

  - How to select an optimal power amplifier load
  - How to determine whether the input matching network needs to be re-designed for the optimal load

- Using S-parameter Input Files

  The S-parameter example explains

  - How to generate tabulated S-parameters
  - How to include tabulated S-parameters as input in a SpectreRF analysis

- Measuring AM and PM Conversion with PAC and PXF Analyses

The AM/PM conversion example explains how to determine the amplitude and phase modulation effects in RF circuits.

Envelope Following Analysis

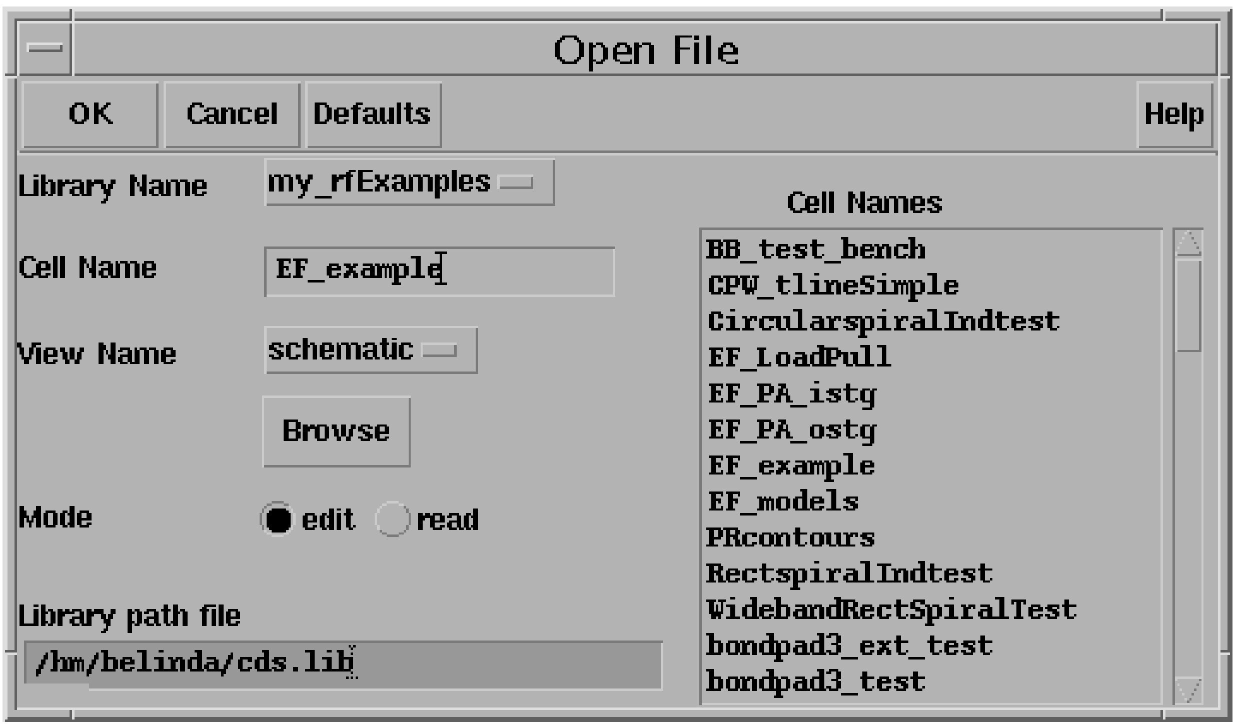
This example tells you how to set up and run the Envelope Following analysis then explains why the time results look somewhat unusual.

Before you start, perform the setup procedures described in Chapter 3.

Opening the EF_example Circuit in the Schematic Window

1. In the CIW, choose *File – Open*.

The Open File form appears.



2. In the Open File form, do the following:

a. Choose *my_rfExamples* for *Library Name,* the editable copy of *rfExamples*.

Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

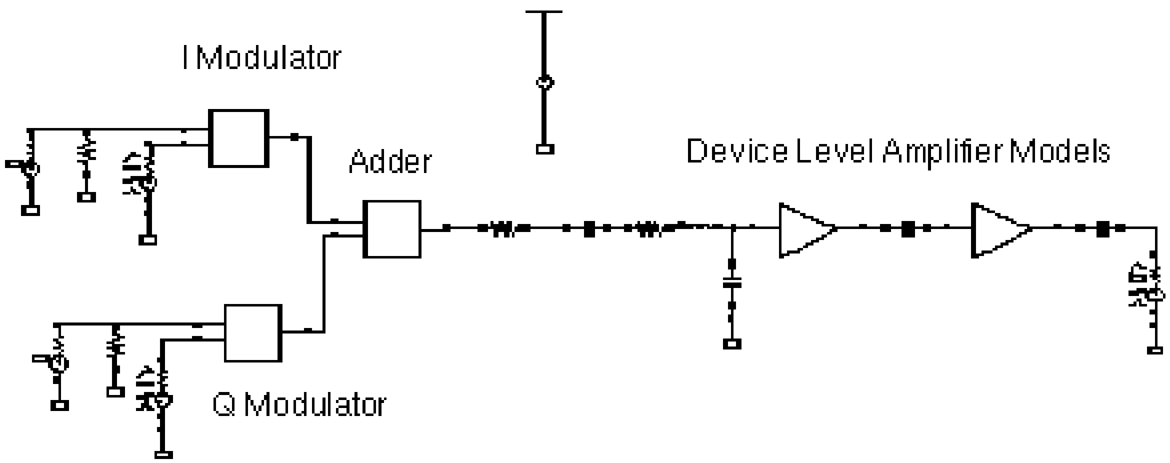
b. In the *Cell Name* list box, highlight *EF_example*.

c. Choose *schematic* for *View Name*.

d. Highlight *edit* for *Mode*.

e. Click *OK*.

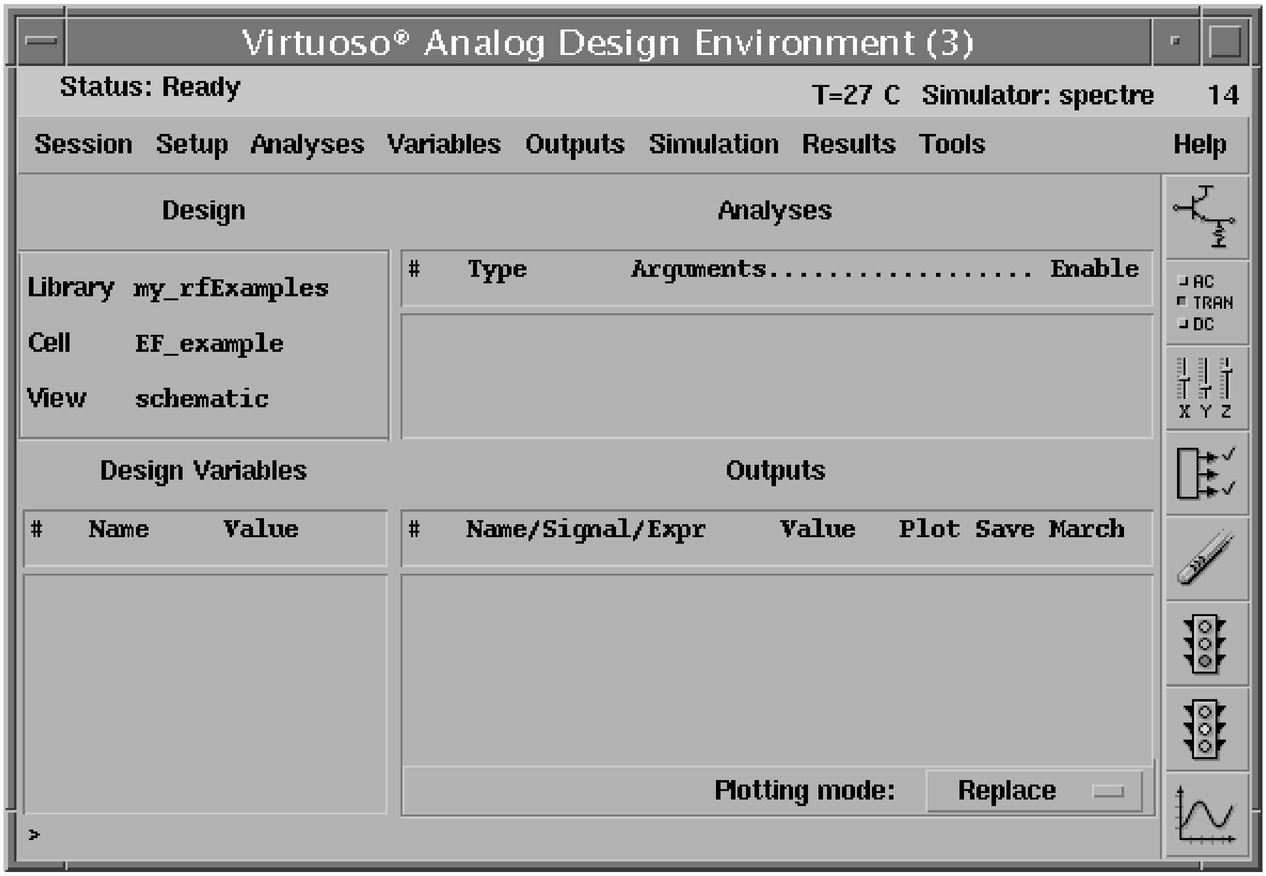
The Schematic window appears with the *EF_example* schematic. This is a simple direct-conversion transmitter with ideal I/Q modulators.



Opening the Simulation Window

1. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window opens.



Note: You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing *EF_example* in the Choosing Design form.

Setting Up the Model Libraries

1. In the Simulation window, choose *Setup - Model Libraries.*

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the following path, where `CDSHOME` is the installation directory for the Cadence software.

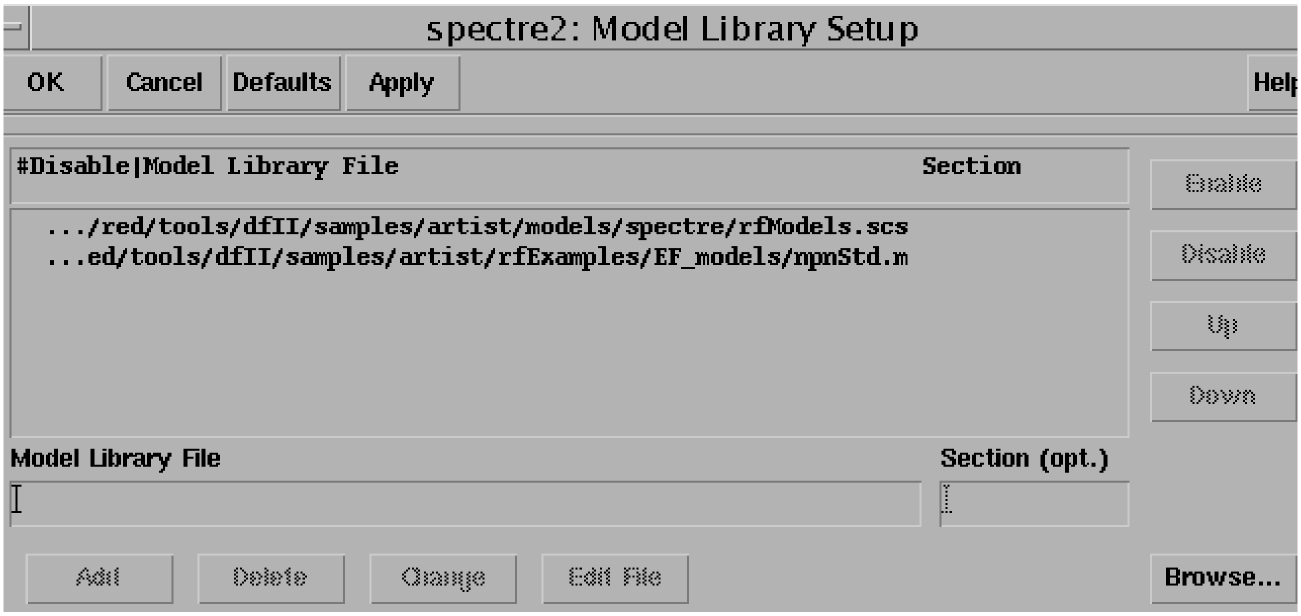
   `<CDSHOME>/tools/dfII/samples/artist/rfExamples/EF_models/npnStd.m`

3. Click on *Add*.

4. In the *Model Library File* field, type the full path to the model file including the file name,

<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs.

The Model Library Setup form looks like the following.



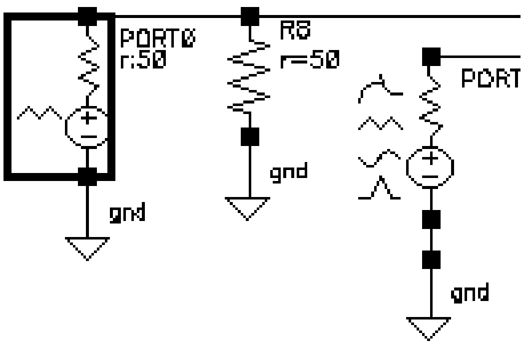
5. First click on *Add*, then click on *OK*.

Editing PORT0 and PORT1 in the Schematic Window

The EF_example circuit uses the programmable voltage source, *port*. The RF voltage source is based on the *port* sample component. You can easily change the behavior of this programmable component.

In the example, you edit PORT0 and PORT1 on the left side of the schematic.

1. In the Schematic window, select `PORT0`. (The port in the top, left corner of the schematic.)



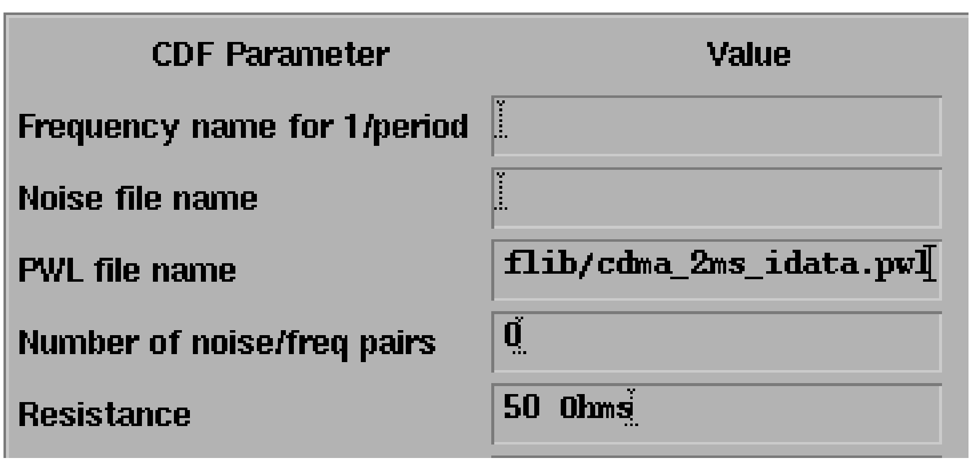
2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears and changes to display information for `PORT0.` You use this form to change the list of CDF (component description format) properties for the `PORT0` and modify the schematic for this simulation.

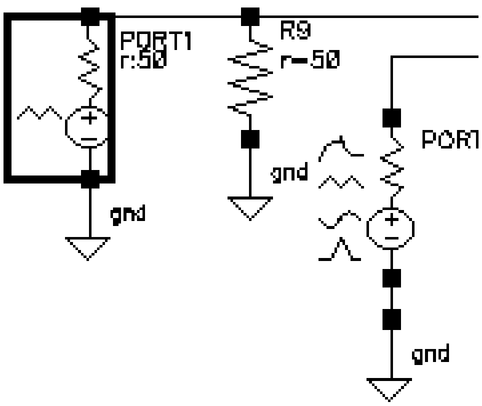
3. In the Edit Object Properties form, edit the *PWL filename* of `PORT0`. Type the following path, where *`CDSHOME`* is the installation directory for the Cadence software. Then click on *Apply*.

   `<CDSHOME>/tools/dfII/samples/artist/rfLib/cdma_2ms_idata.pwl`

   The section of the Edit Object Properties form that includes the *PWL file name* field looks like the following.



4. In the Schematic window, select `PORT1`. (The port in the bottom, left corner of the schematic.)



The Edit Object Properties form changes to display information for `PORT1`.

5. In the Edit Object Properties form, edit the *PWL filename* of `PORT1`. Type the following path, where *`CDSHOME`* is the installation directory for the Cadence software. Then click on *Apply*.

`<CDSHOME>/tools/dfII/samples/artist/rfLib/cdma_2ms_qdata.pwl`

6. Click on *OK*.

7. Choose *Design - Check and Save* in the Schematic window.

Setting Up an Envelope Following Analysis

1. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *envlp*.

The Choosing Analyses form changes to let you specify values for the Envelope Following analysis.

3. In the Choosing Analyses form, do the following and then click *OK*.

a. Click *Select Clock Name* to display the Select Clock Name form. In the Select Clock Name form, click `fff` then click *OK.*

In the *Clock Name* field, `fff` appears.

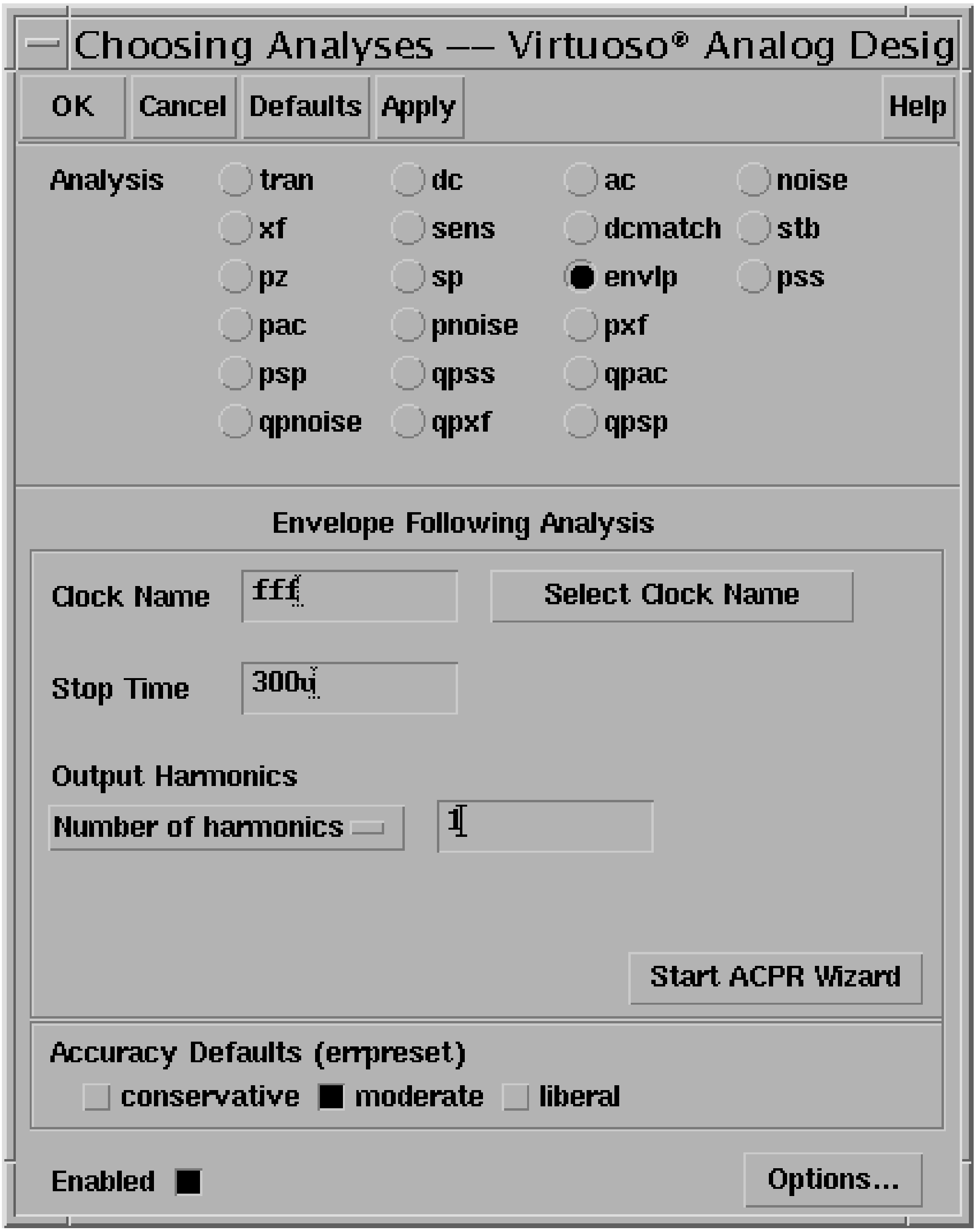
b. Type `300u` for *Stop Time*.

c. In the for *Output Harmonics* cyclic field, choose *Number of harmonics* and type 1 in the adjacent field.

d. Highlight *moderate* for the *Accuracy Defaults (errpreset)* setting.

e. Verify that *Enabled* is highlighted.

The completed form looks like this.



4. In the Simulation window, choose *Simulation – Netlist and Run*.

   Look in the CIW for messages saying that the simulation has started and completed successfully. Watch the simulation log file for information as the simulation runs.

Looking at the Envelope Following Results

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form and the Waveform window appear.

2. In the Direct Plot form, do the following:

   a. Choose *Replace* for *Plot Mode*.

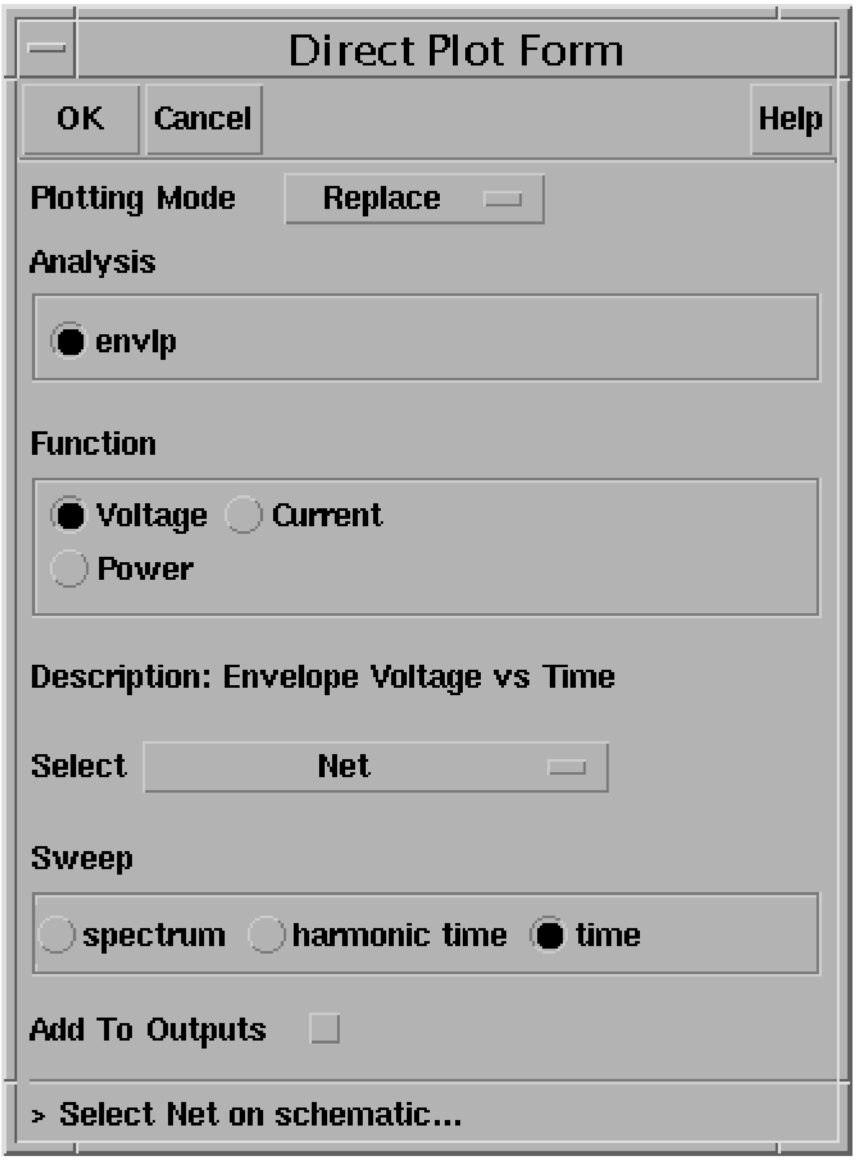
   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

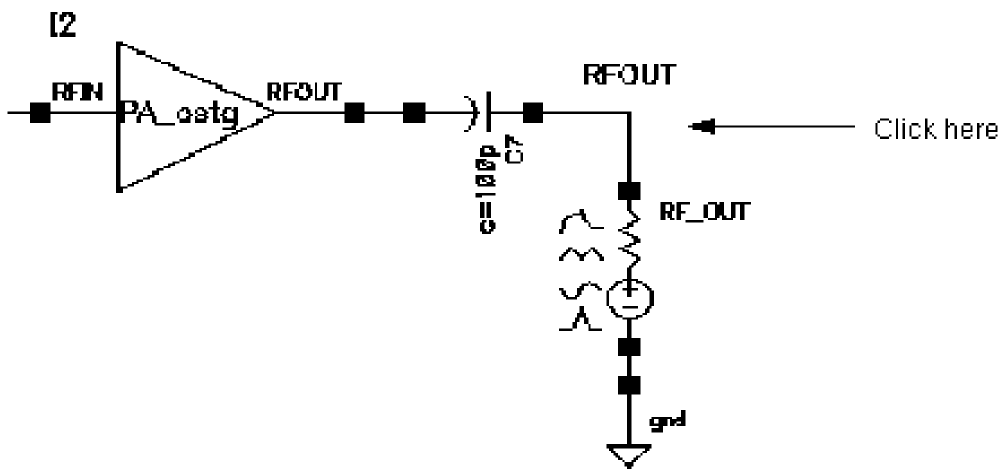
   d. Highlight *time* for *Sweep*.

   e. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.
   > *Select Net on Schematic...*.

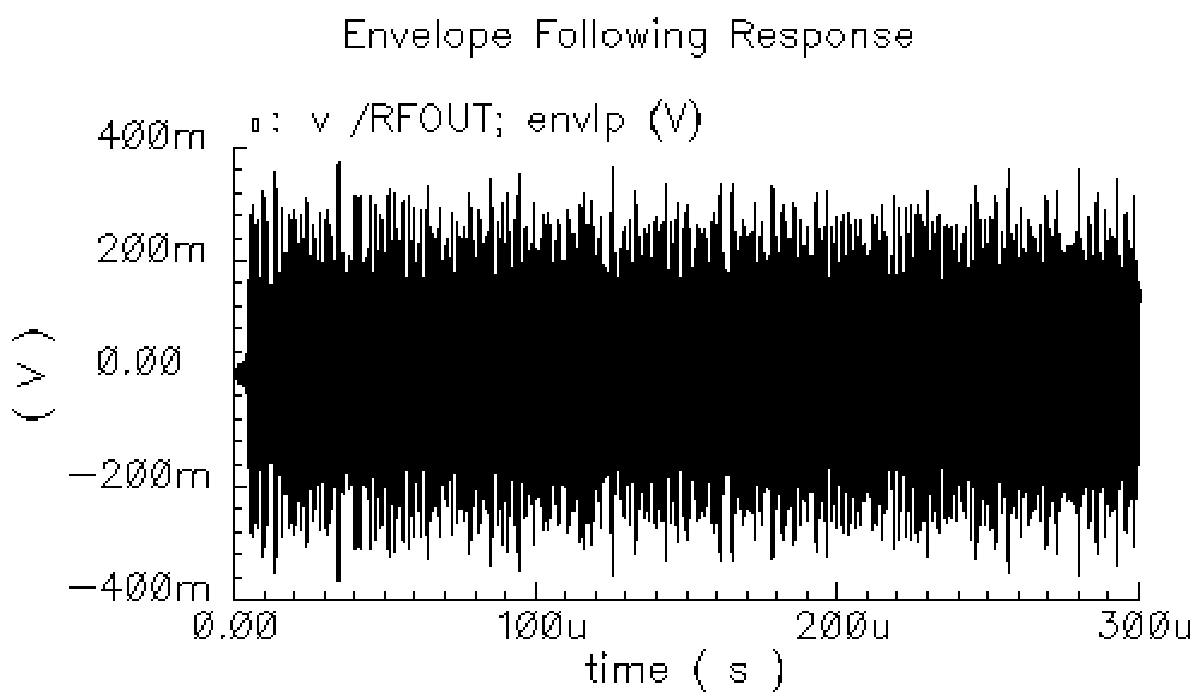
The completed form looks like this.



3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the RF output net (RFOUT) as shown.

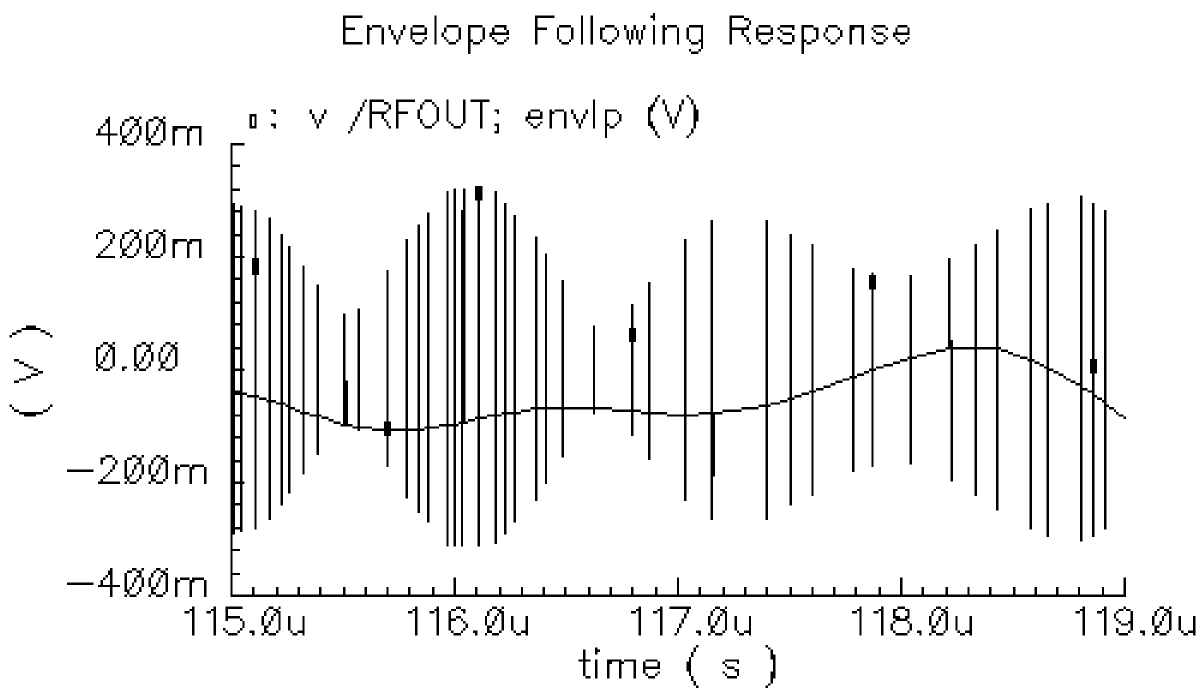


The voltage waveform for *RFOUT* appears in the Waveform window.



4. To get a closer look at a small section of the waveform, in the Waveform window, right click and drag to select a small section of the waveform. (The area you select covers a narrow vertical rectangle.)

You might have to do this several times to get a plot similar to the one shown.



The plot changes to display a number of vertical lines with a wavy line running through them.

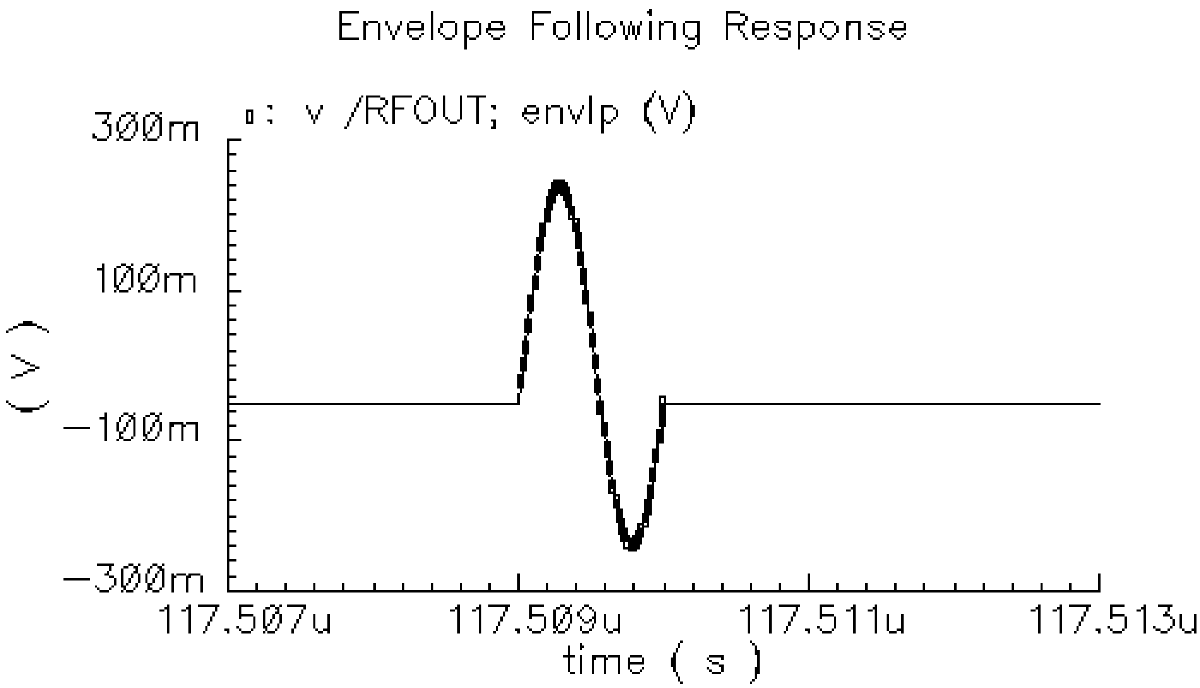
❑ Each vertical line is a time point where a detailed calculation was performed

❑ The wavy line connects these points

An Envelope Following analysis runs much faster than a Spectre transient analysis because Envelope Following analysis skips carrier cycles when it can do so while still satisfying numerical tolerances.

5. To get a closer look, right click and drag to select one of the vertical lines. You might have to do this several times.

After several selections, you will be able to see the detailed simulation plot for one complete cycle. What you see should be similar to the following waveform depending on the areas you selected.



Following the Baseband Signal Changes Through an Ideal Circuit

The modulation riding on the RF carrier is the baseband signal, the information to be transmitted. The baseband signal determines the amplitude and phase of the RF carrier. In transmitter design, it is important to determine how the transmitter might alter the baseband signal.

This section tells you how to extract and plot the baseband signal at several points in the circuit.

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form and the Waveform window appear.

2. In the Direct Plot form, do the following:

   a. Choose *Replace* for *Plot Mode*.

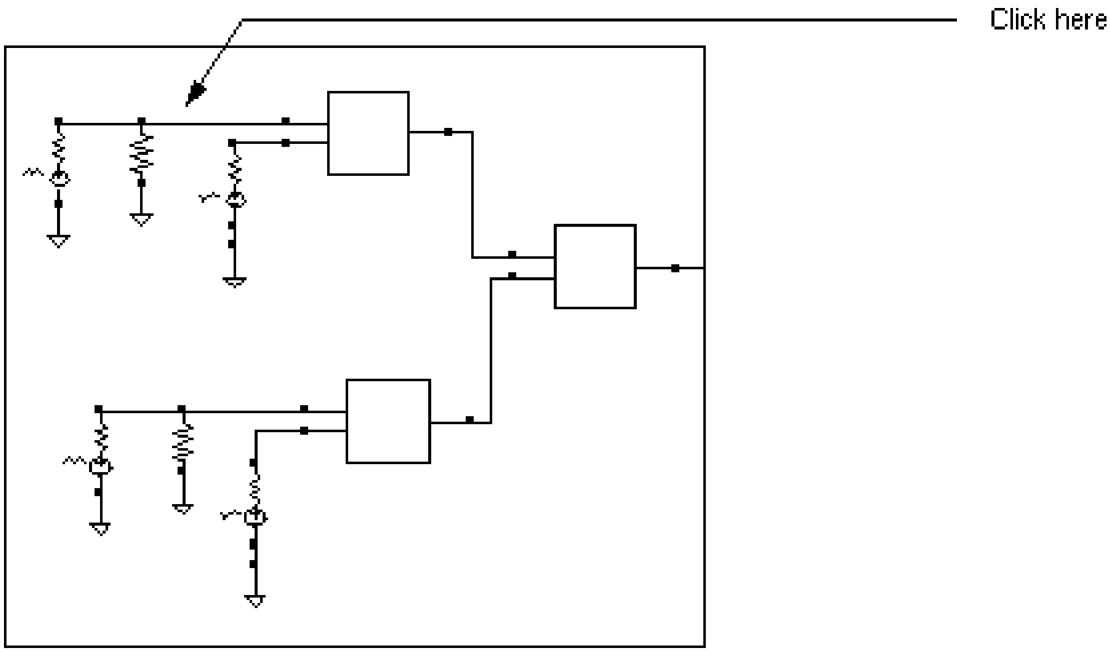
   b. Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form. displays > *Select Net on Schematic*.

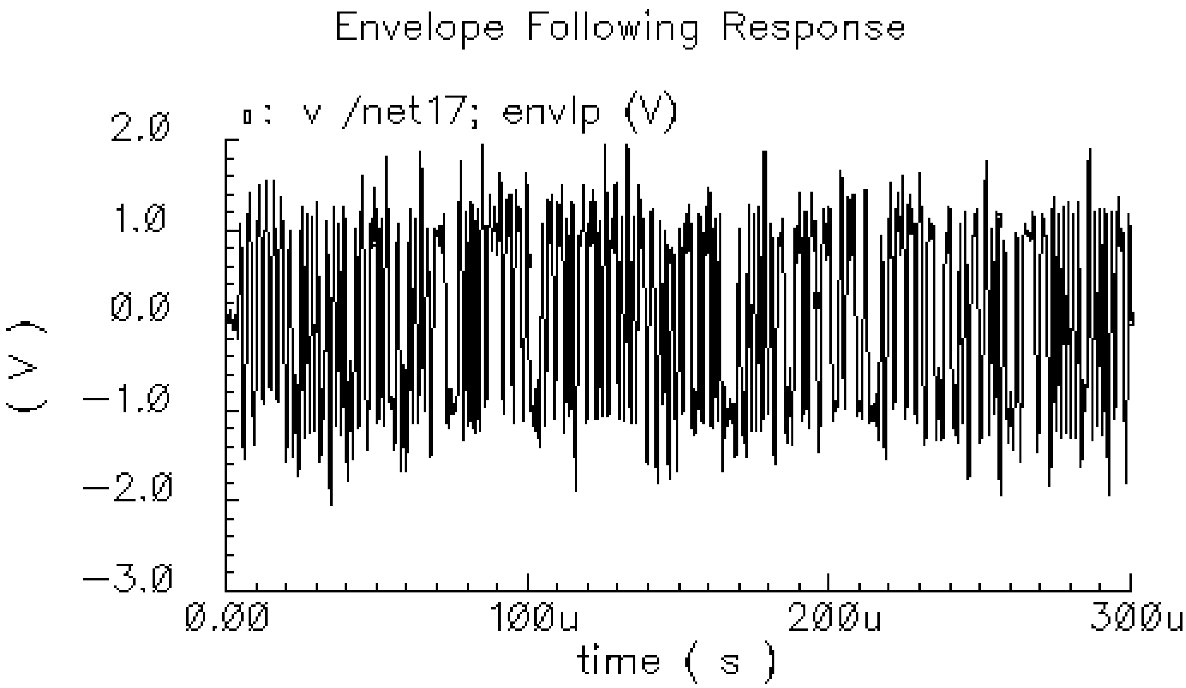
e. Highlight *time* for *Sweep*.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the I-modulator source net.



The plot for the modulation source waveform appears.

Note: You might have to click *Zoom – Fit* in the Waveform window before you are able to see the waveform.



4. In the Waveform window, choose *Axes – To Strip*.

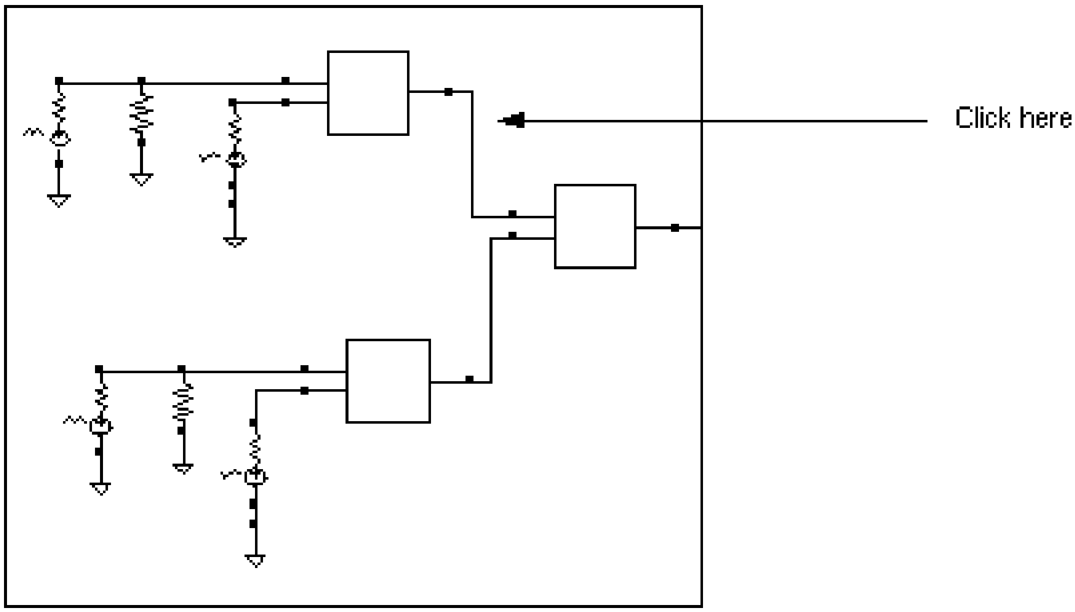
   This changes the Waveform window to display multiple waveforms in strips, one above the other.

5. In the Direct Plot form, highlight *Append* for *Plot Mode*.

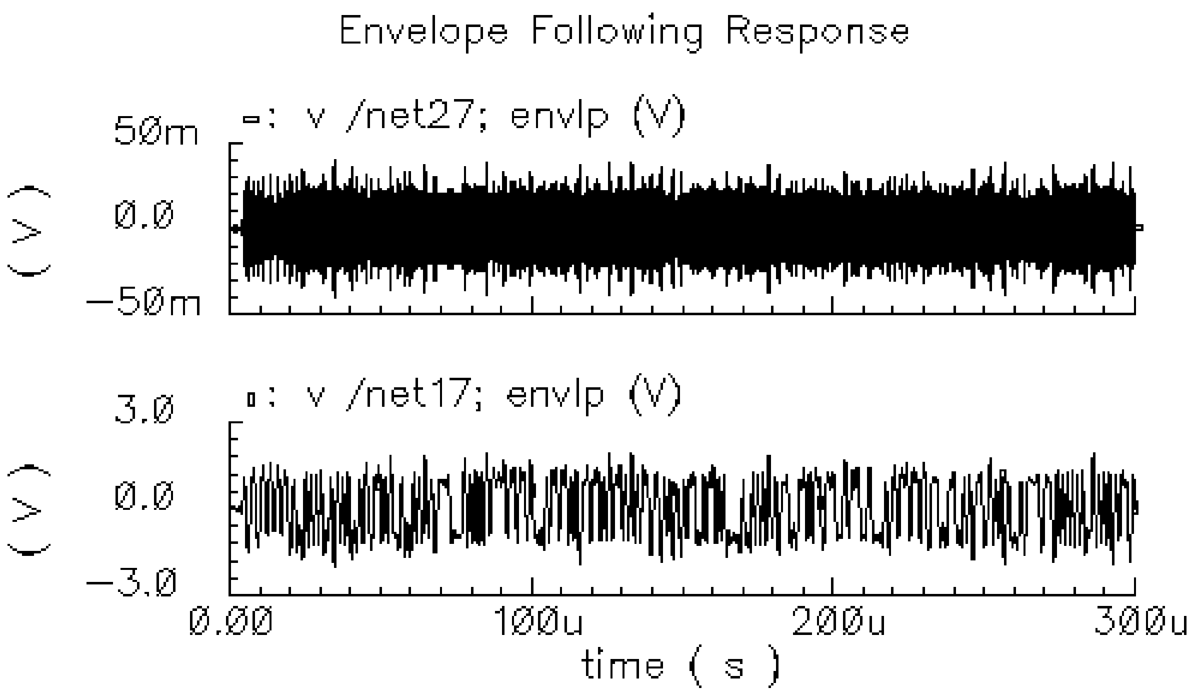
   This adds new waveforms to any waveforms currently displayed in the Waveform window.

6. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the I-modulator output net as shown in the following schematic.

This is the in-phase carrier modulated by the I-component of the baseband signal. In this example, the in-phase carrier is `cos(`ω_c`t)`, where ω_c is the carrier frequency in radians per second. The quadrature carrier is `sin(`ω_c`t)`.

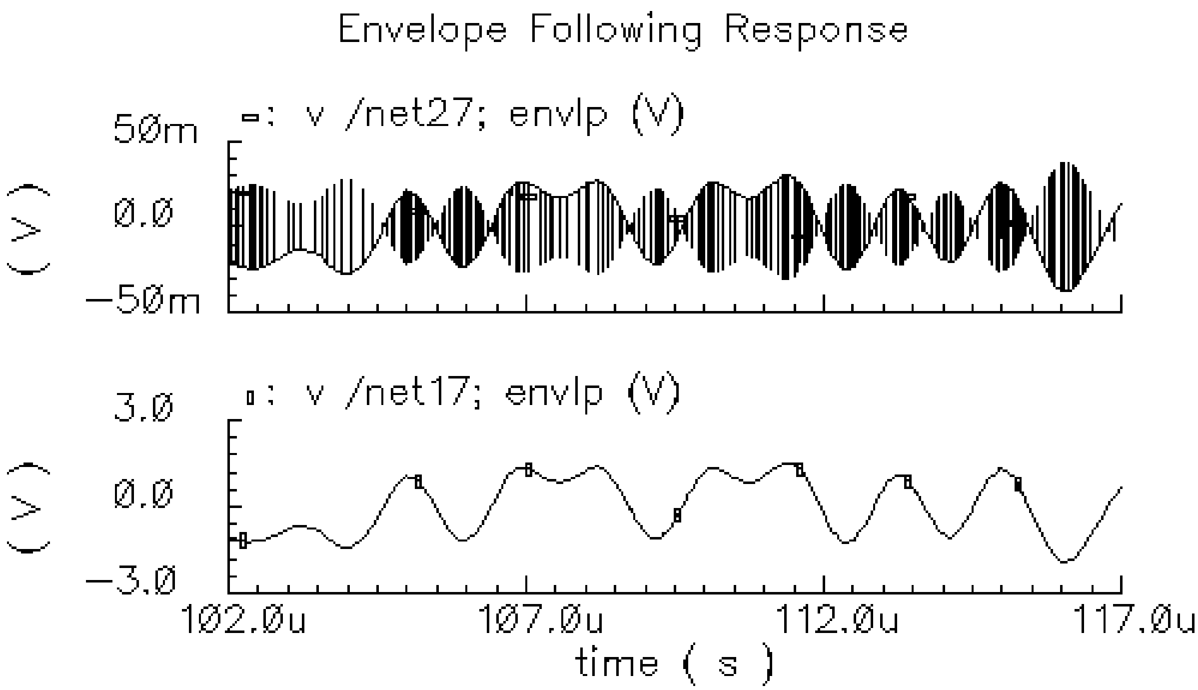


The I-modulator output waveform is added to the Waveform window. Note that the color of the selected net in the schematic matches the color of the waveform in the Waveform window.



7. In the Waveform window, right click and drag over a narrow section of both waveforms.

You can see the modulation wave within the output wave. It is coincidental that the bottom waveform and smooth curve in the top waveform look alike. If the individual cycles in the top waveform were sampled at a different phase, the smooth curve in the top waveform might look different.



Note: The appearance of this plot, as well as the appearance of subsequent plots, depends on the section of the Waveform window in which you perform the click and drag.

8. In the Direct Plot form, do the following:

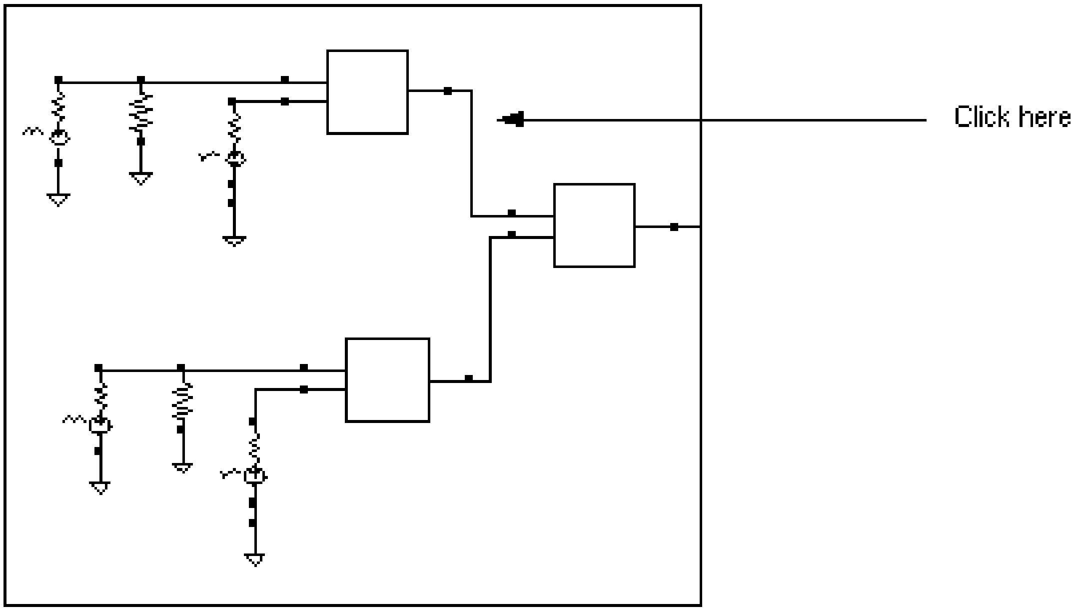
a. Highlight *harmonic time* for *Sweep*.

b. Highlight *Real* for *Modifier*.

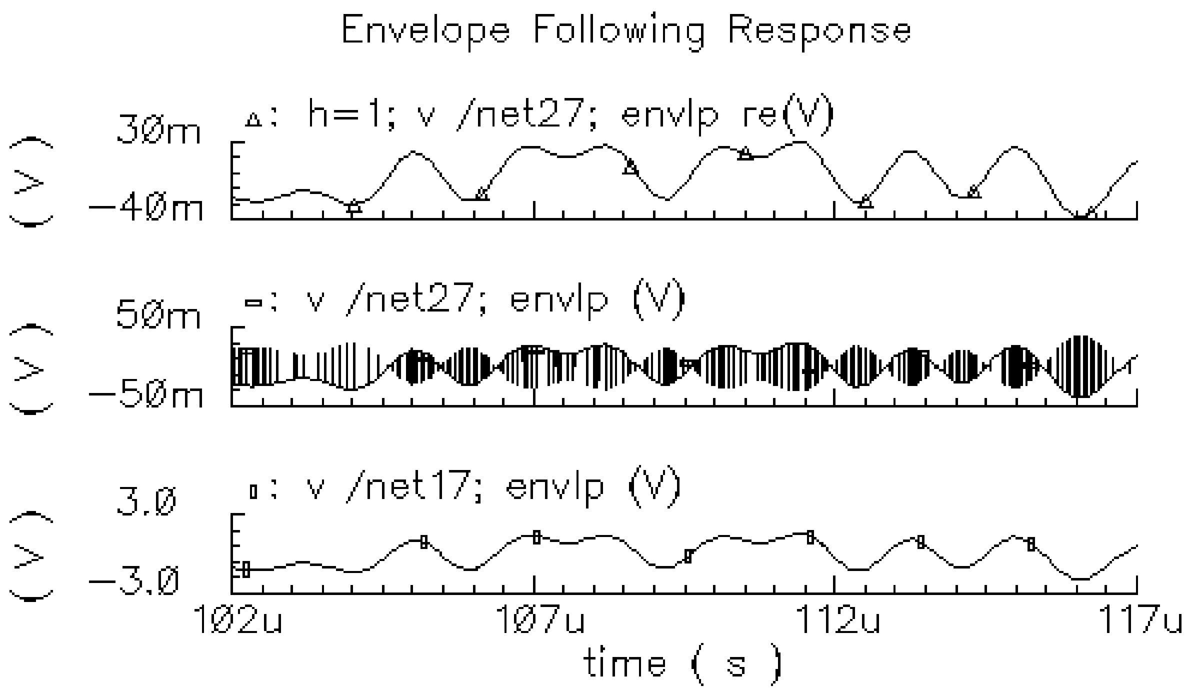
c. Highlight *1* for *Harmonic Number*.

d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.
> *Select Net on Schematic...* .

Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the I-modulator output net.



9. The top plot in the Waveform window is the baseband waveform recovered from the modulated RF carrier. Aside from a linear scale factor, It looks exactly like the bottom waveform because the modulator is ideal.



Following the Baseband Signal Changes Through a Non-Ideal Circuit

To see the change in the signal as it passes through the entire length of the circuit, which is not ideal, follow the instructions in this section.

1. In the Waveform window, choose *Window - Reset* to clear the Waveform window.

2. In the Direct Plot form, do the following:

a. Choose *Replace* for *Plot Mode*.

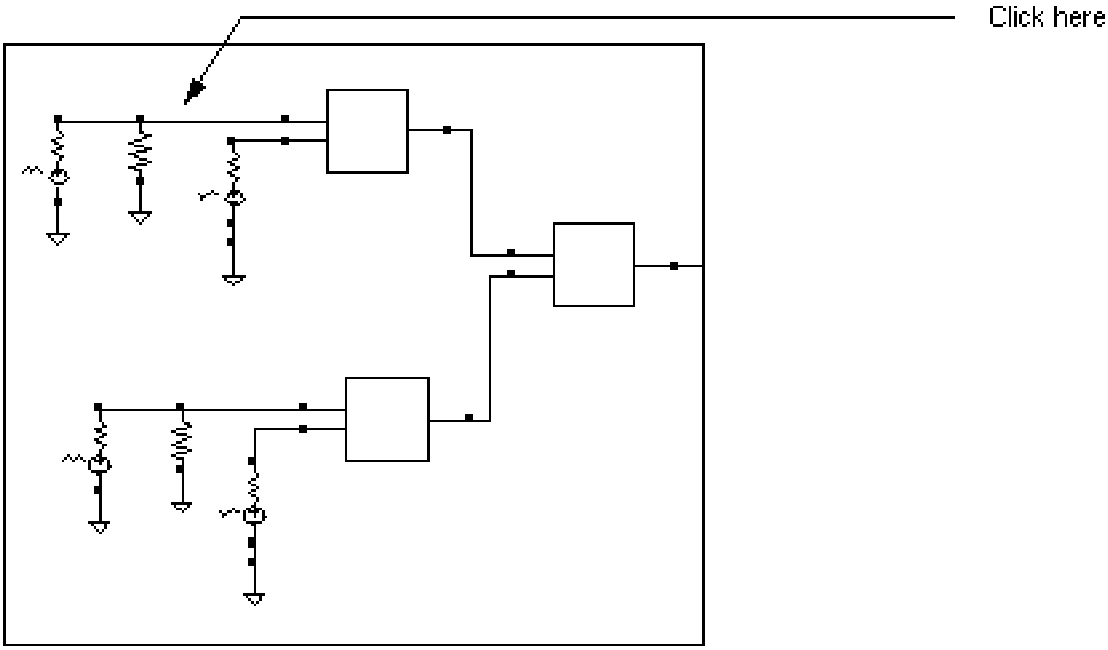
b. Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

d. Highlight *time* for *Sweep*.

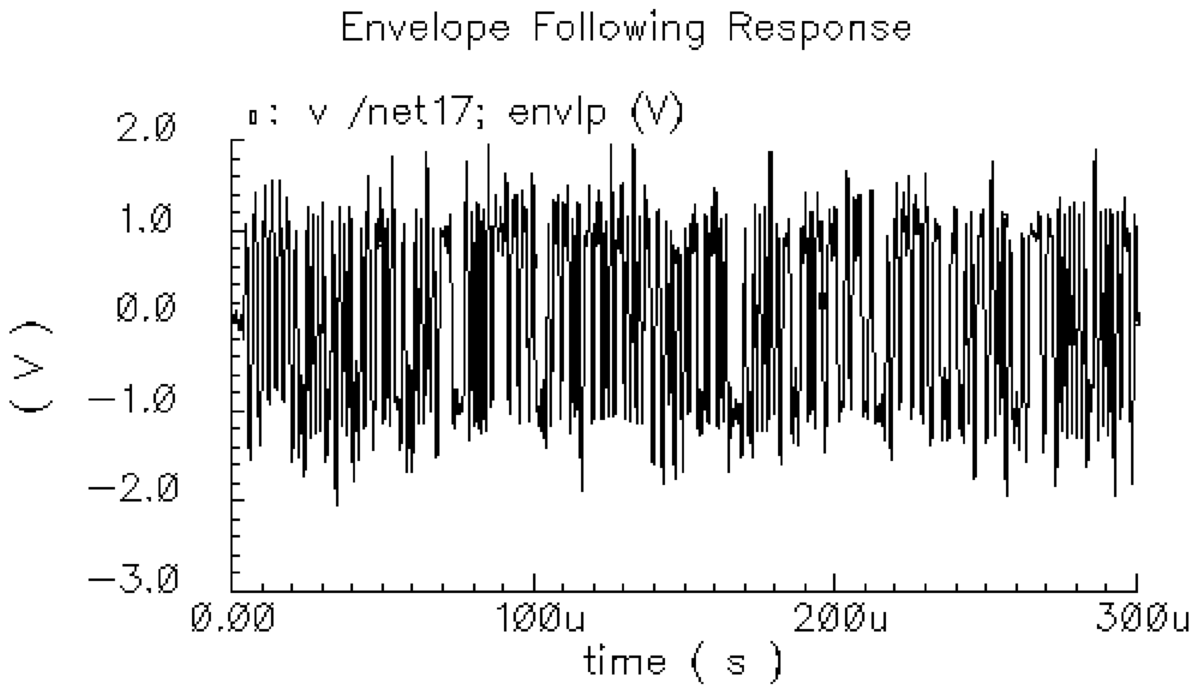
e. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form
> *Select Net on Schematic...*.

3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the I-modulator source net.



The plot for the modulation source waveform appears.

Note: You might have to choose *Zoom – Fit* in the Waveform window before you are able to see the waveform.



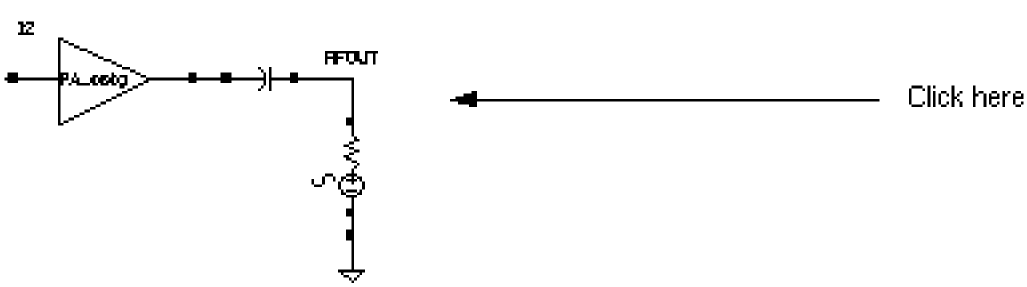
4. In the Waveform window, choose *Axes – To Strip*.

   This changes the Waveform window to display multiple waveforms in strips, one above the other.

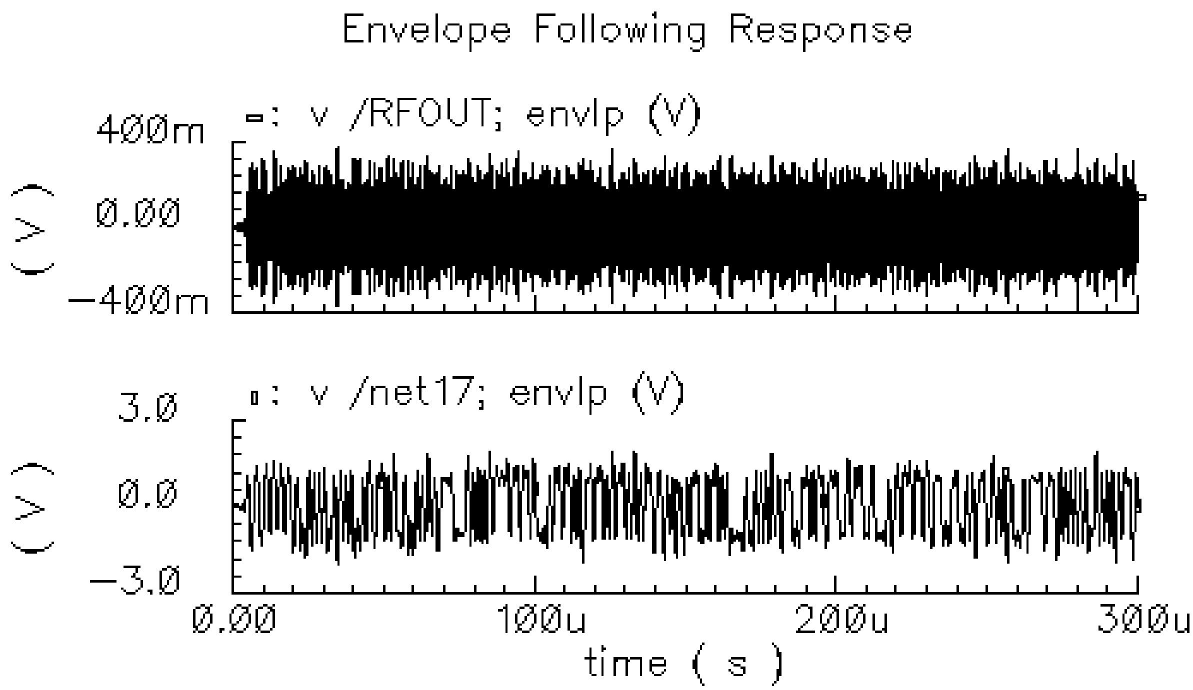
5. In the Direct Plot form, highlight *Append* for *Plot Mode*.

   This adds new waveforms to any waveforms currently displayed in the Waveform window.

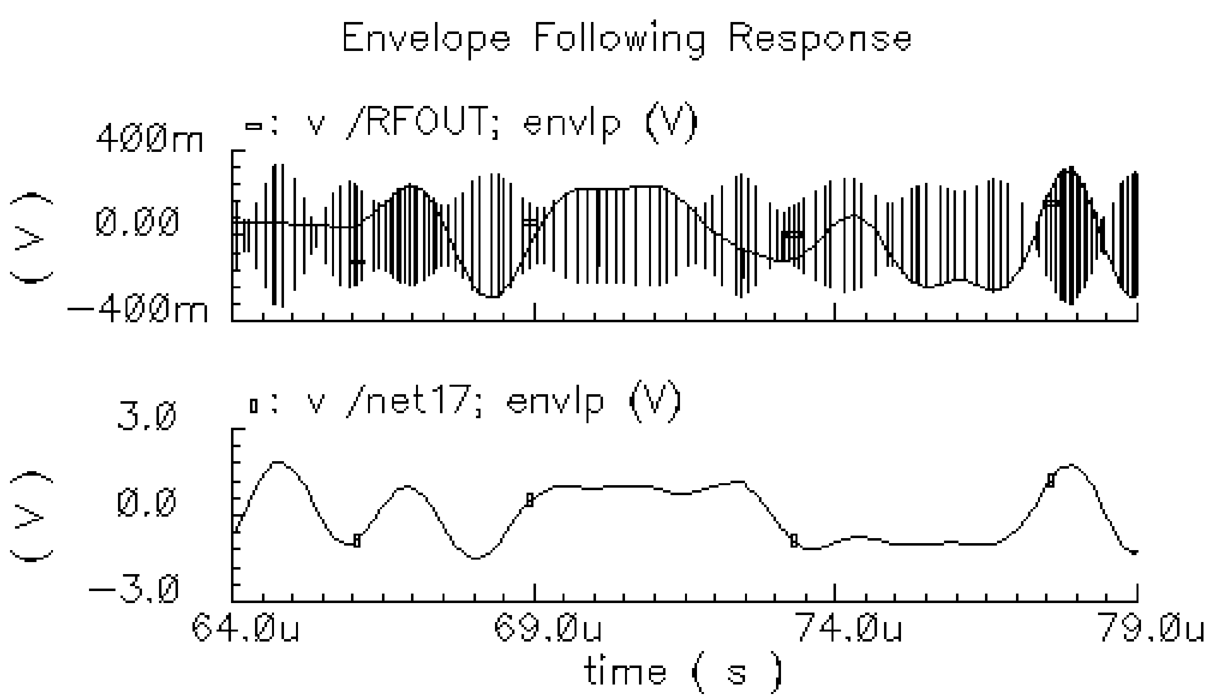
6. Following the prompt at the bottom of the Direct Plot form, in the Schematic window, click on the transmitter output net (RFOUT) as shown in the following schematic.



The transmitter output waveform is added to the Waveform window.



7. In the Waveform window, right click and drag over a narrow section of both waveforms.



You can see the modulation wave within the output wave. It is coincidental that the bottom waveform and smooth curve in the top waveform look alike. If the individual cycles in the top waveform were sampled at a different phase, the smooth curve in the top waveform might look different.

Note: The appearance of this plot as well as subsequent plots depends on the section of the Waveform window in which you perform the click and drag.

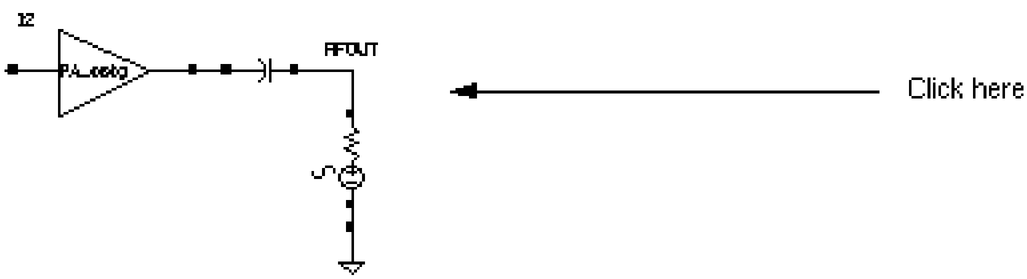
8. In the Direct Plot form, do the following:

  a. Highlight *harmonic time* for *Sweep*.

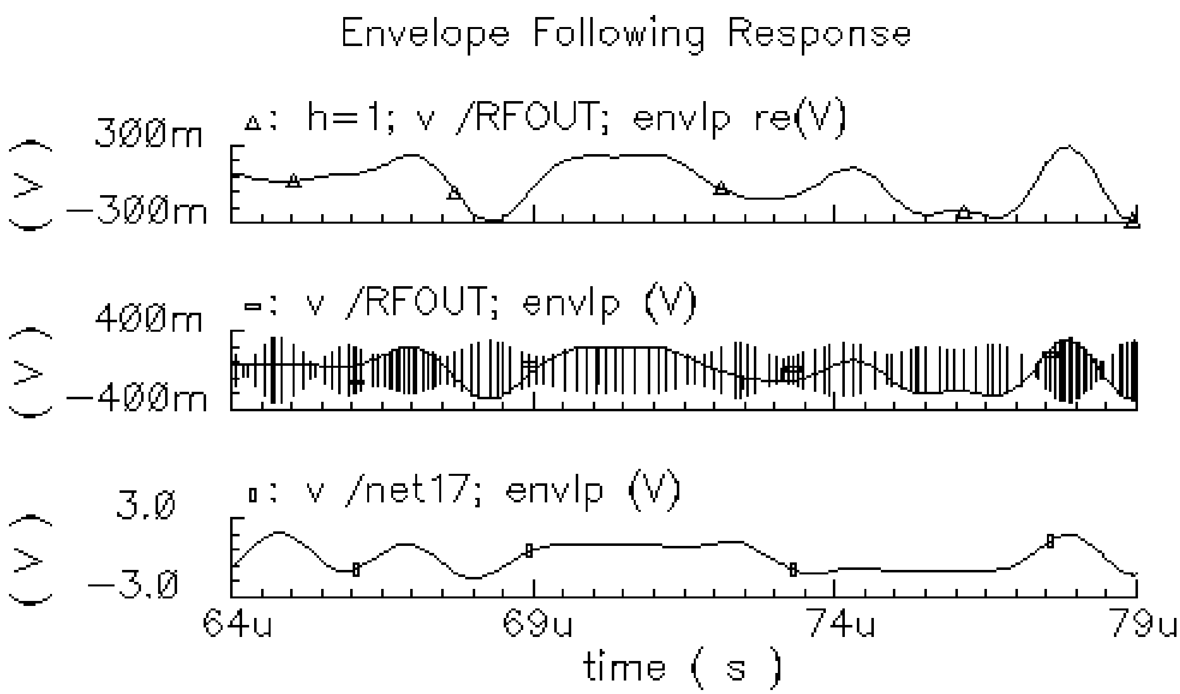
  b. Highlight *Real* for *Modifier*.

  c. Highlight *1* for *Harmonic Number*.

9. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the transmitter output net (RFOUT) as shown in the following schematic.



The waveform at the transmitter output appears at the top of the Waveform window. It shows a greater change in the signal.



Plotting the Complete Baseband Signal

1. In the Direct Plot form, do the following:

a. Choose *Replace* for *Plot Mode*.

b. Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

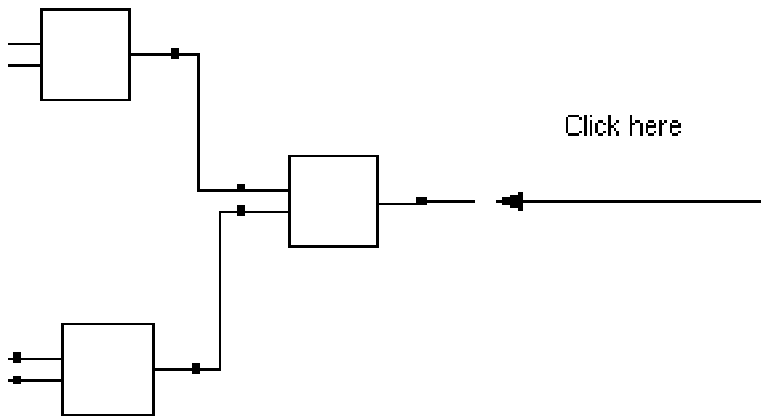
d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.
> *Select Net on Schematic*.

e. Highlight *harmonic time* for *Sweep*.

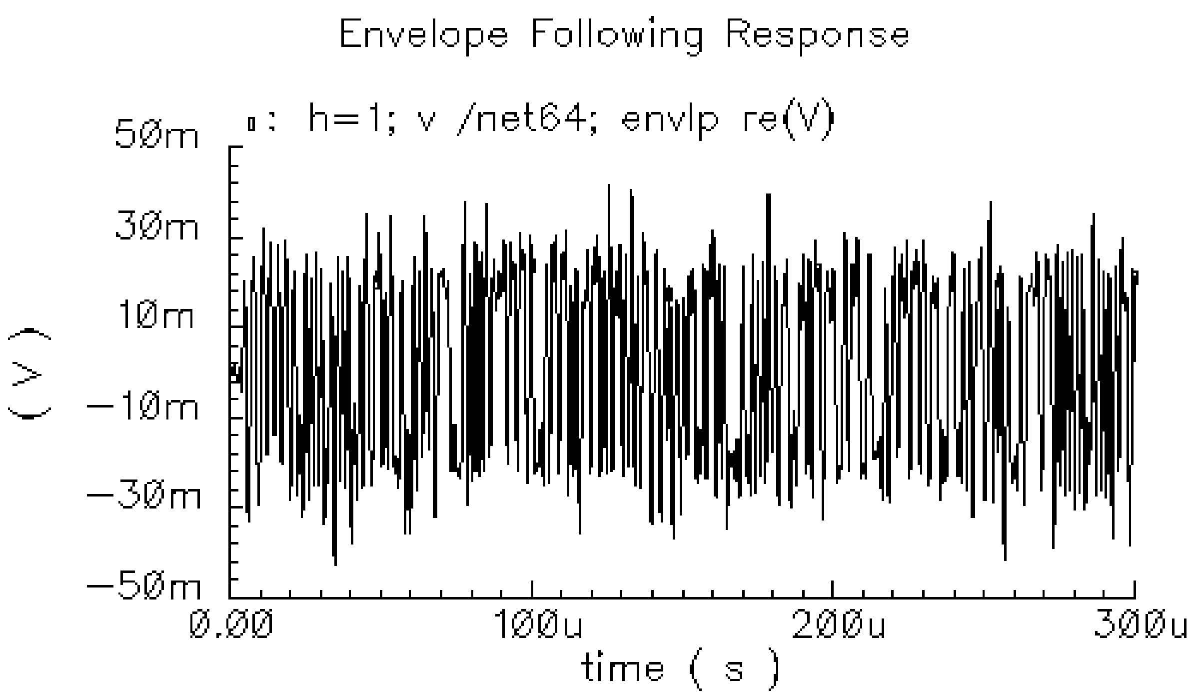
f. Highlight *Real* for *Modifier*.

g. Highlight *1* for *Harmonic Number*.

2. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the adder output net.



A plot for the real portion of the adder output waveform appears in the Waveform window.

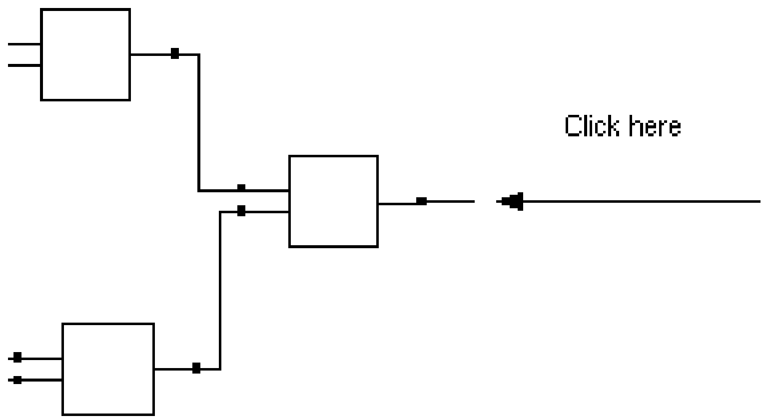


3. In the Direct Plot form, make the following changes:

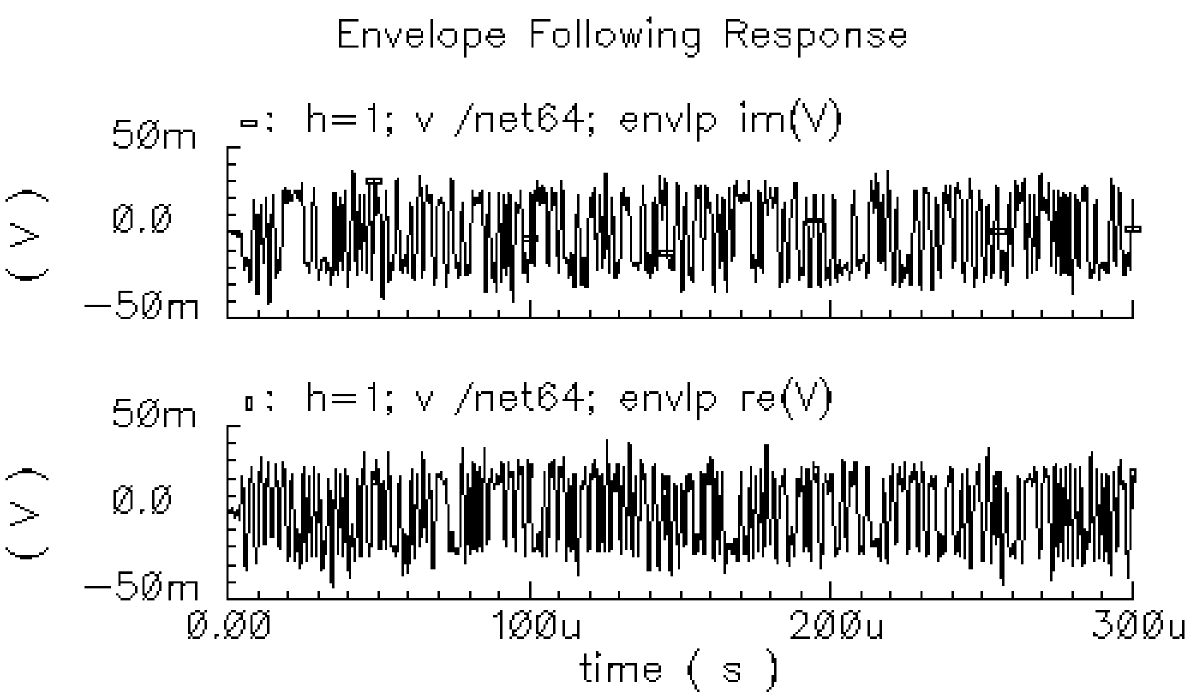
   a. Choose *Append* for *Plot Mode*.

   b. Highlight *Imaginary* for *Modifier*.

4. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the adder output net.

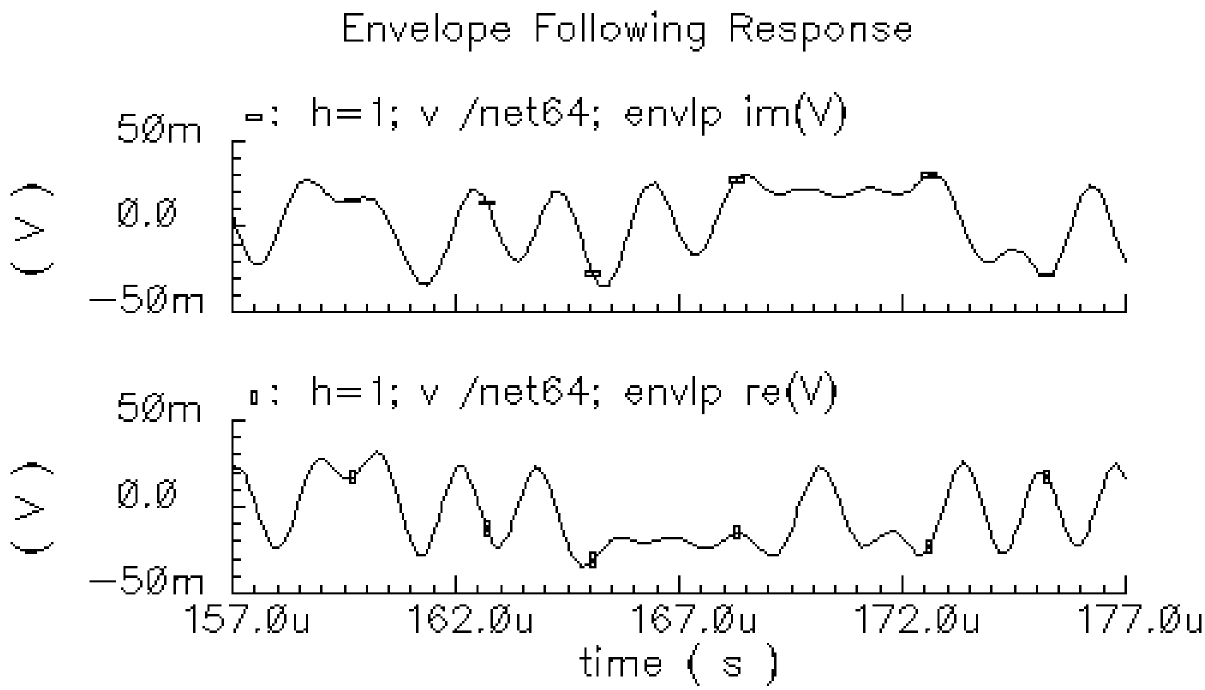


A plot for the imaginary portion of the adder output waveform is added to the Waveform window.



5. In the Waveform window, right click and drag to select a portion of the Waveform window over a narrow portion of the x-axis.

You can now clearly see both the real part and the imaginary part of the baseband signal.



Plotting the Baseband Trajectory

This section describes how to display the baseband trajectory, the plot of one waveform against the other. The baseband waveforms recovered from the modulated RF carrier, as displayed in the previous section, do not directly reveal much information about how the transmitter affects them.

The baseband trajectory reveals much more information about the kind of distortion the transmitter introduces. The procedure described in this section displays first the input baseband trajectory followed by the output baseband trajectory. A comparison of the two trajectories reveals whether the power amplifiers in this example introduce any phase shift.

Displaying the Input Baseband Trajectory

1. In the Waveform window, choose *Window - Reset*.
2. In the Direct Plot form, do the following:
   - **a.** Choose *Replace* for *Plot Mode*.
   - **b.** Highlight *envlp* for *Analysis*.

c. Highlight *Voltage* for *Function*.

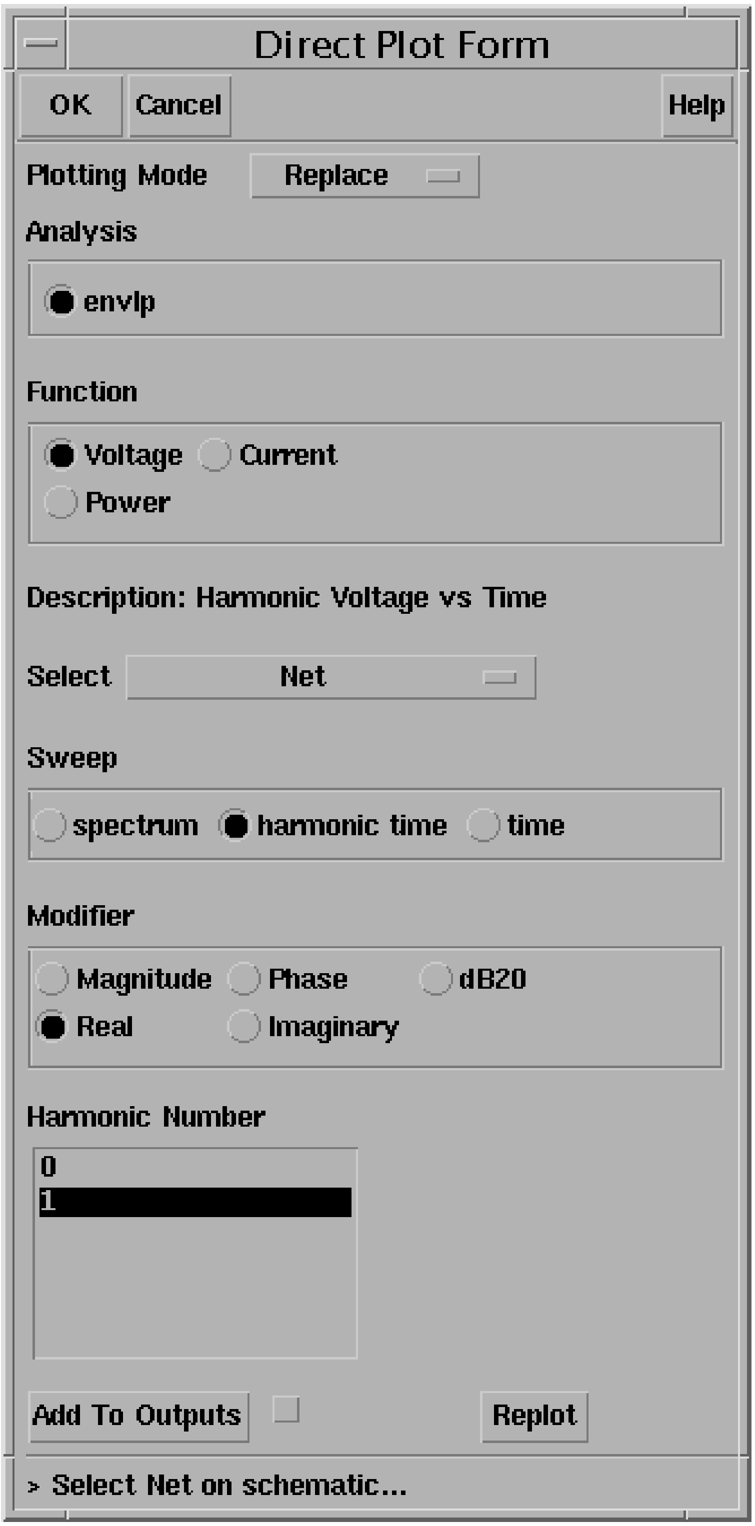
d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form.
> *Select Net on Schematic* .

e. Highlight *harmonic time* for *Sweep*.

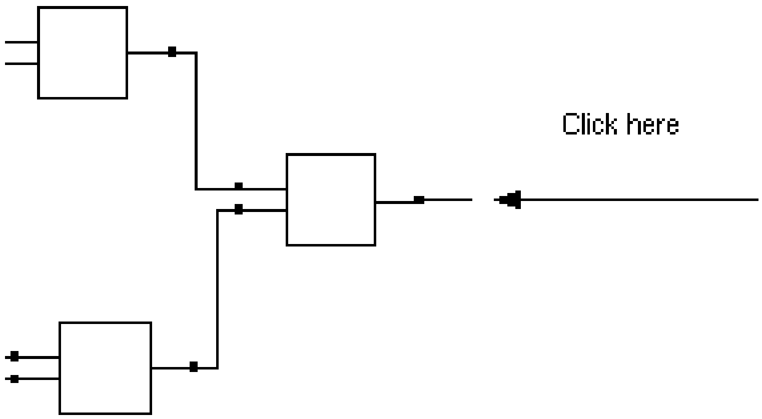
f. Highlight *Real* for *Modifier*.

g. Highlight *1* for *Harmonic Number*.

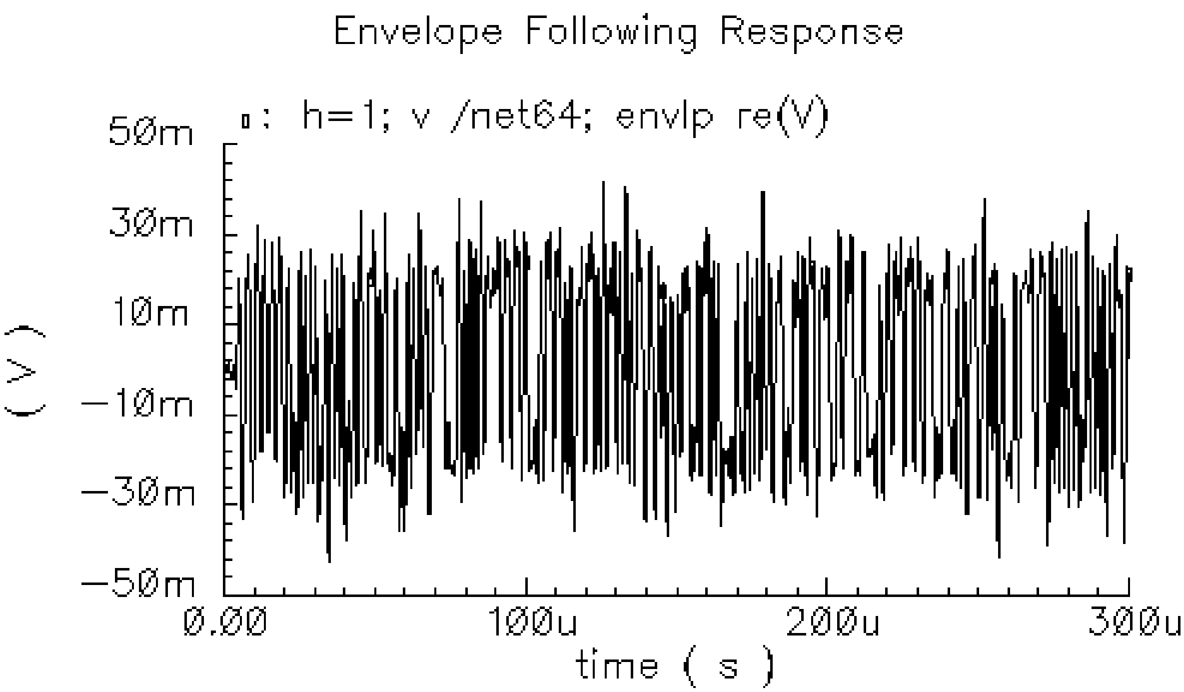
The completed form looks like this.



3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the adder output net.



A plot for the real portion of the adder output waveform appears in the Waveform window.

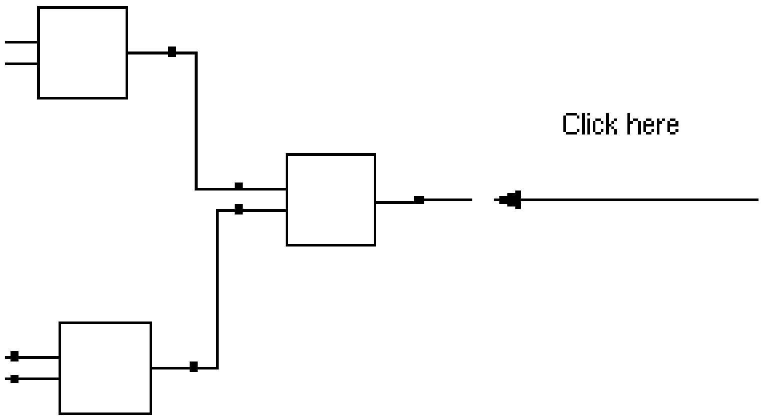


4. In the Direct Plot form, change the following:

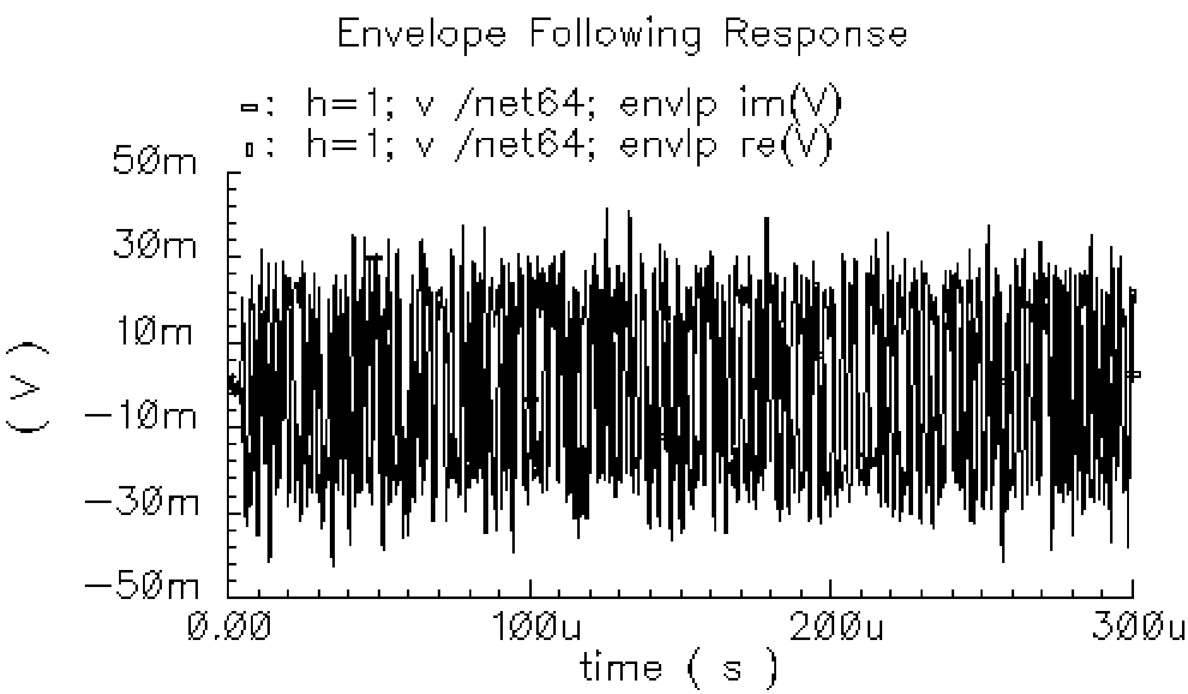
a. Choose *Append* for *Plot Mode*.

b. Highlight *Imaginary* for *Modifier*.

5. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the adder output net.



A plot for the imaginary portion of the adder output waveform is added to the real portion in the Waveform window.

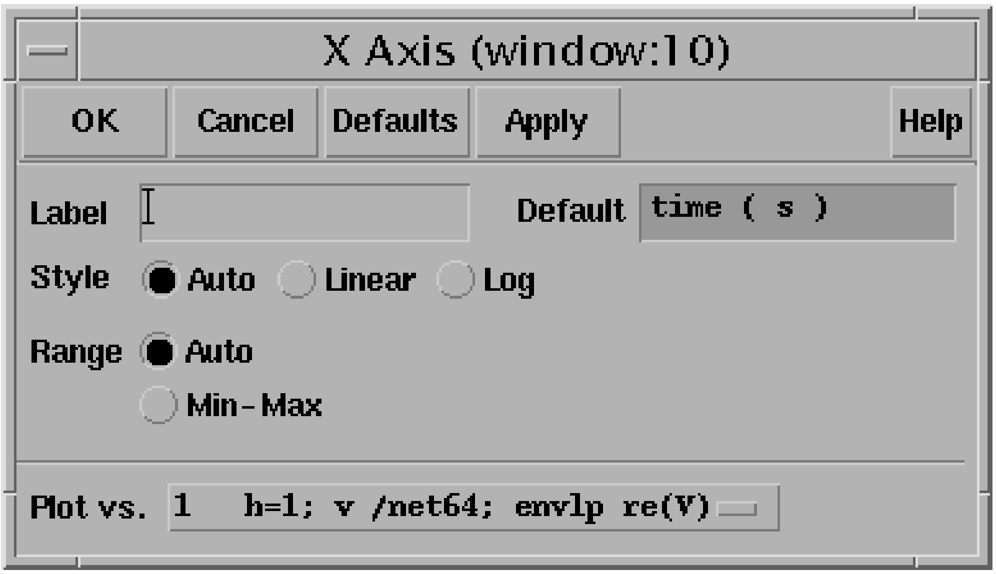


6. In the Waveform window, choose *Axes – X Axis*.

The X Axis form appears.

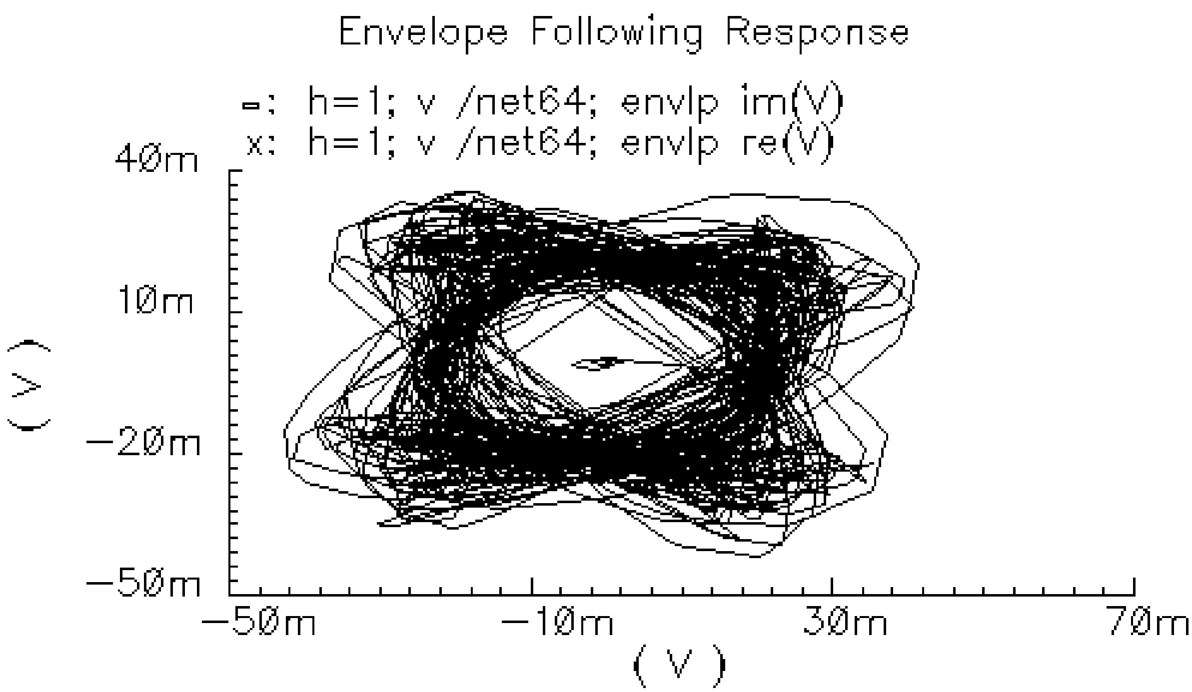
7. In the *Plot vs.* cyclic field of the X Axis form, select the real portion of the waveform. Choose the selection that contains *re(V)*. In this case, choose.

1 h-1; v /net64; envlp re(V)



Click *OK*.

The input baseband trajectory, undistorted by the power amplifiers, appears in the Waveform window.



Displaying the Output Baseband Trajectory

1. In the Simulation window, choose *Tools – Waveform*.

   A second Waveform window appears.

2. In the Direct Plot form, do the following:

   a. Highlight *Replace* for *Plot Mode*.

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

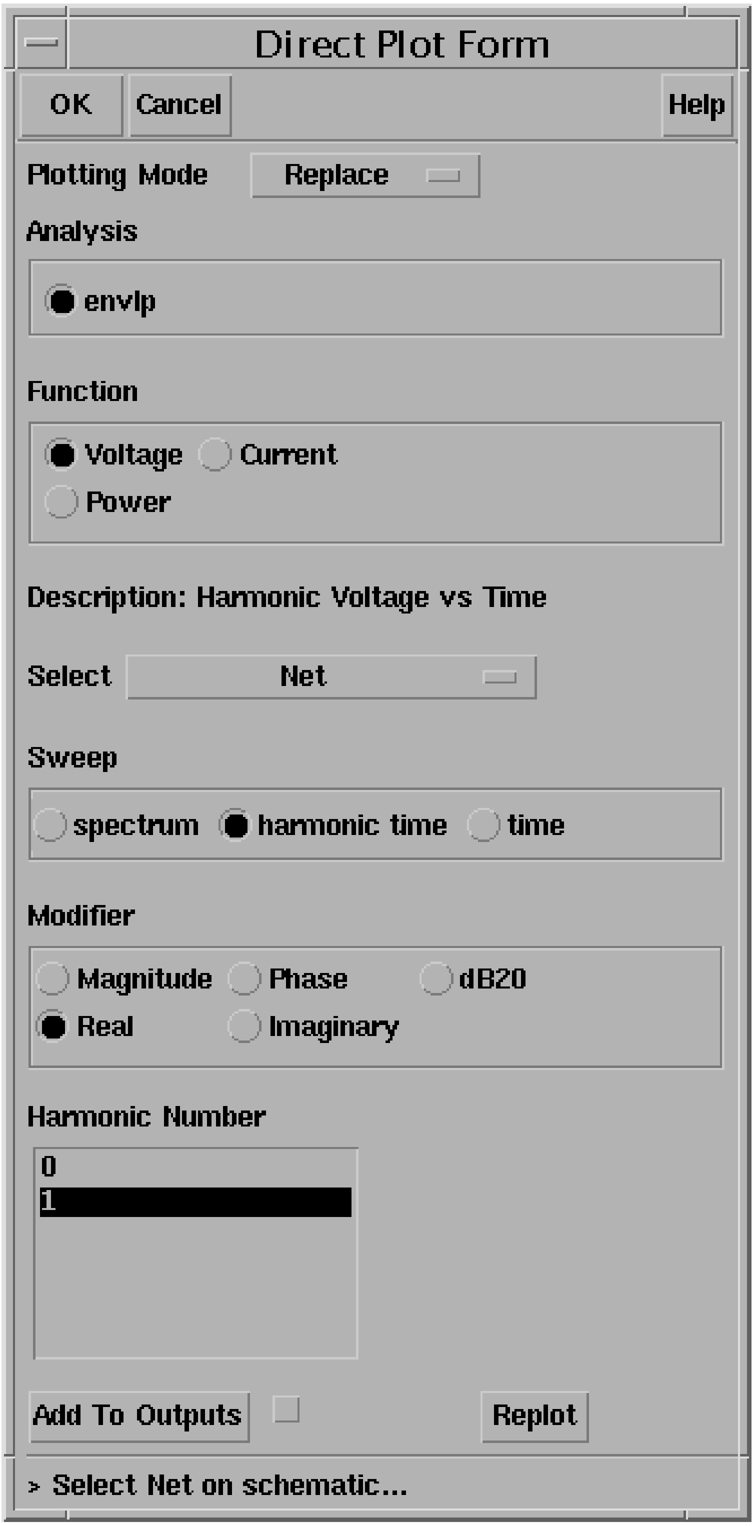
   d. The *Select* cyclic field displays *Net* and a prompt at the bottom of the form. > *Select Net on Schematic*.

   e. Highlight *harmonic time* for *Sweep*.

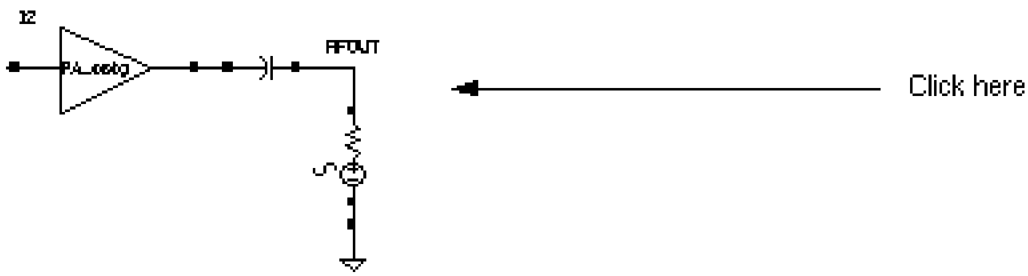
   f. Highlight *Real* for *Modifier*.

   g. Highlight *1* for *Harmonic Number*.

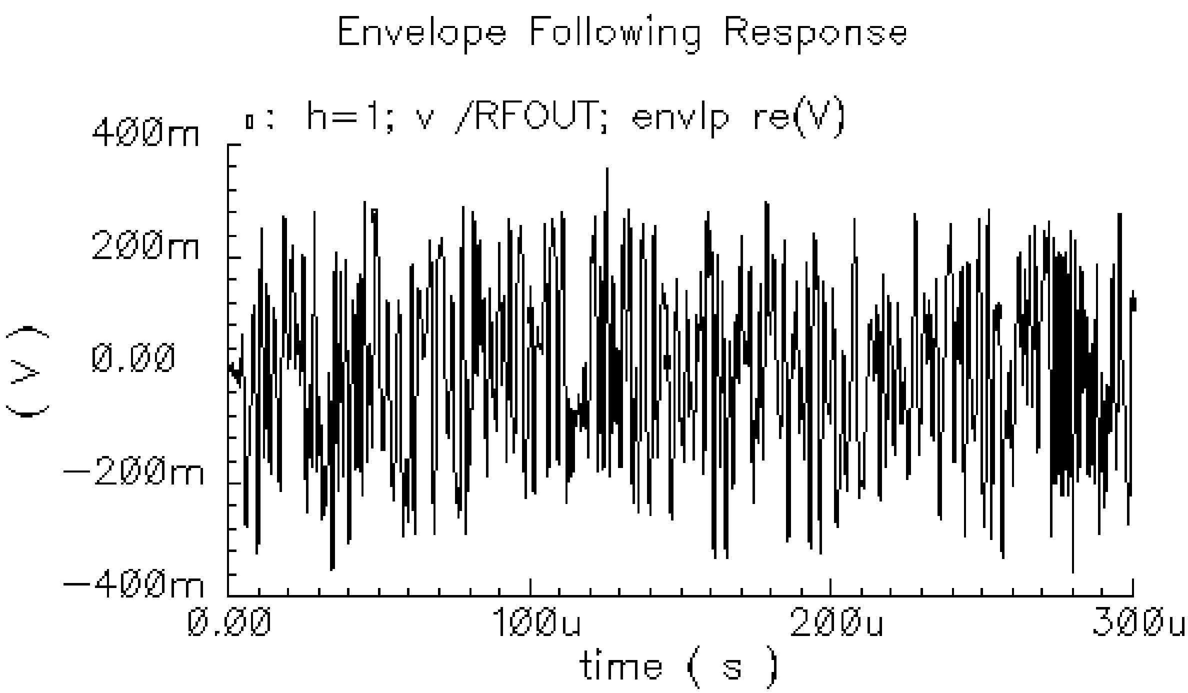
The completed form looks like this.



3. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the transmitter output net (/RFOUT).



A plot of the real portion of the transmitter output waveform appears in the Waveform window.



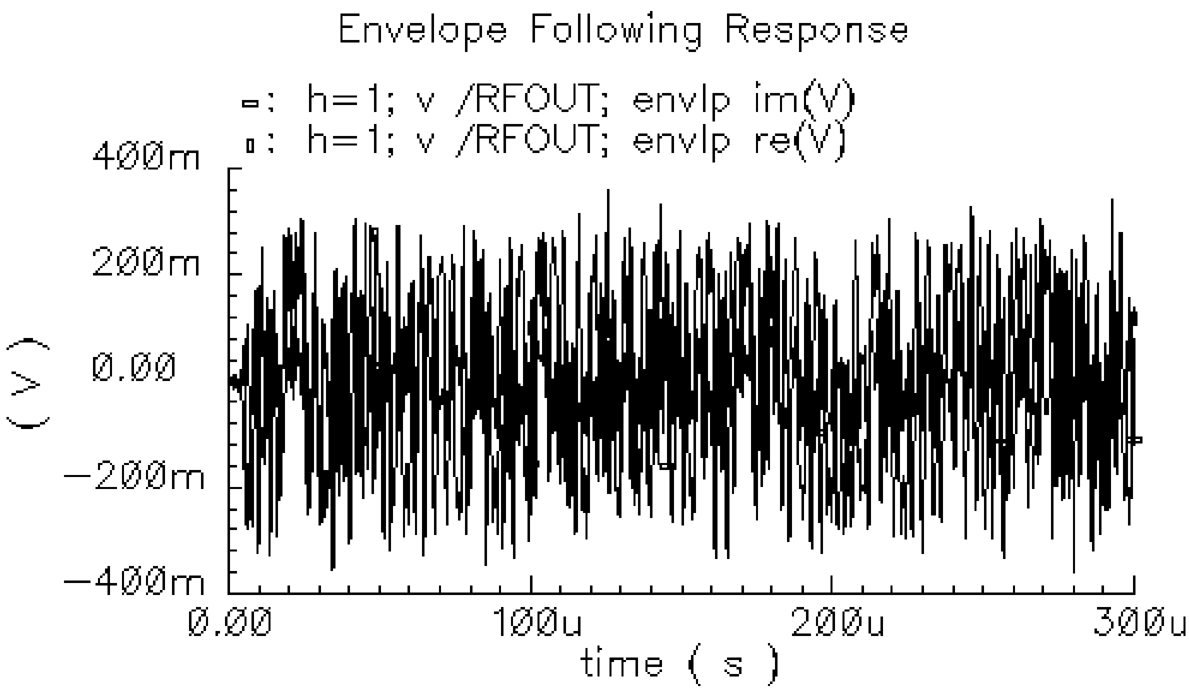
4. In the Direct Plot form, change the following:

a. Highlight *Append* for *Plot Mode*.

b. Highlight *Imaginary* for *Modifier*.

5. Follow the prompt at the bottom of the Direct Plot form. In the Schematic window, click on the transmitter output net (RFOUT).

A plot for the imaginary portion of the transmitter output waveform is added to the real portion in the Waveform window.

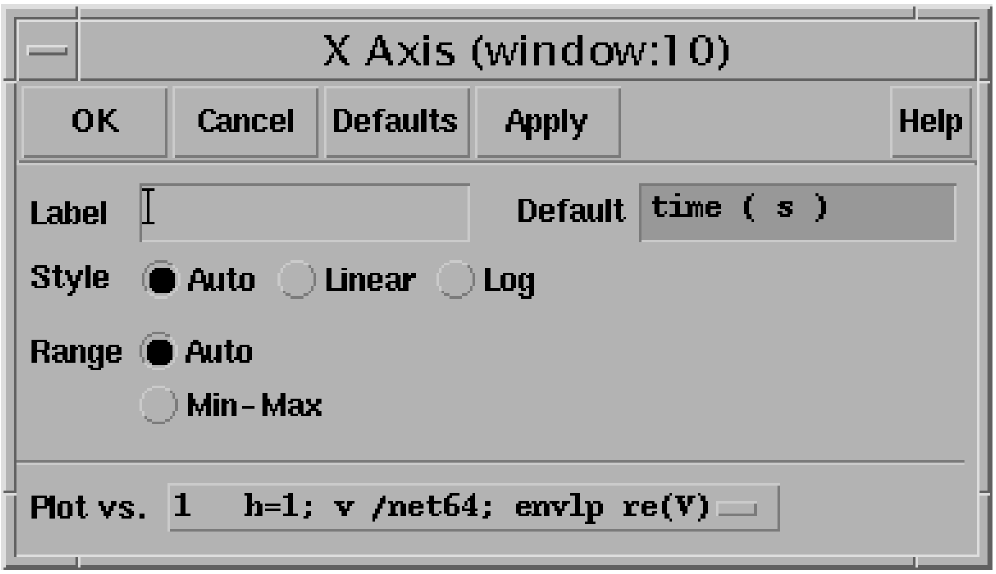


In the Waveform window, choose *Axes – X Axis*.

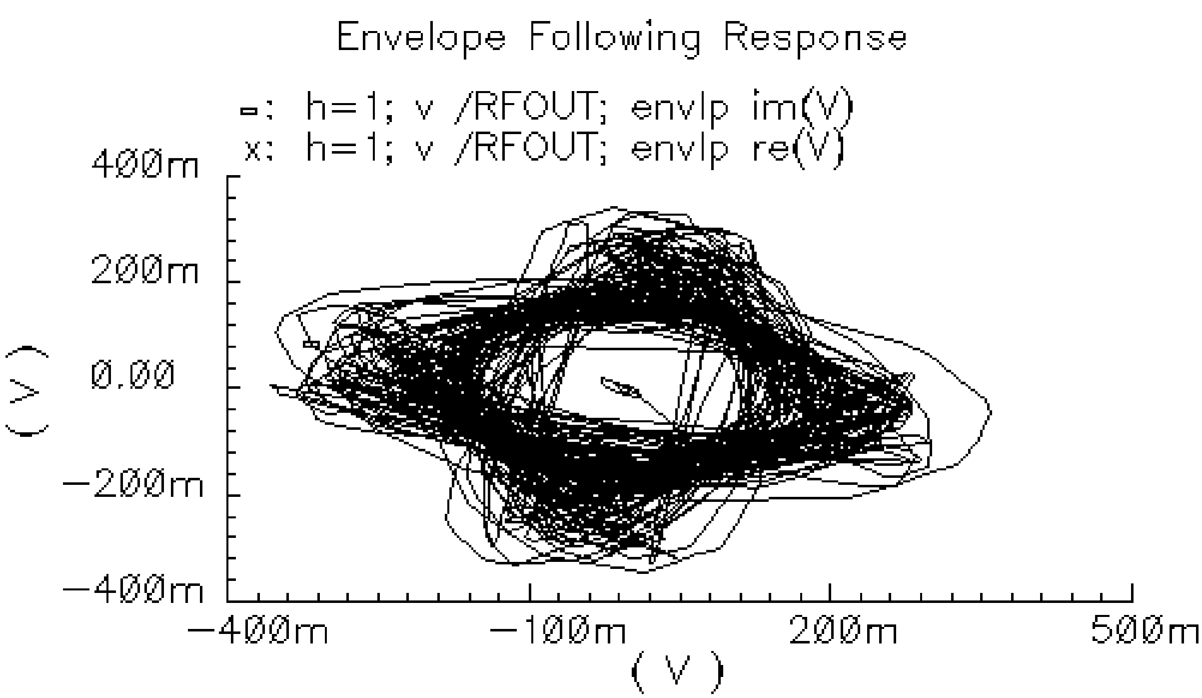
The X Axis form appears.

1. In the *Plot vs.* cyclic field of the X Axis form, select the real portion of the waveform. Choose the selection that contains *re(V)*. In this case, choose

   1 h-1; v /RFOUT; envlp re(V)

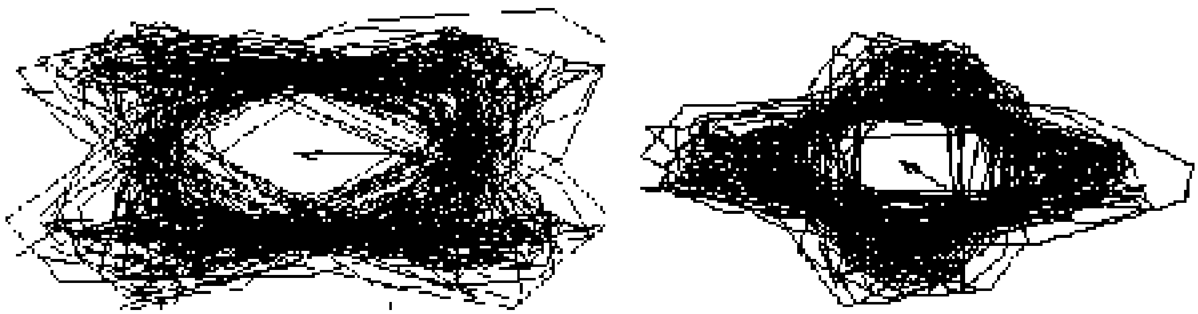


The output baseband trajectory appears in the second Waveform window.



As shown in Figure 10-1, the output baseband trajectory (on the right) is the input baseband trajectory (on the left) scaled linearly and rotated.

Figure 10-1 Input and Output Baseband Trajectories



In other words, the output baseband signal is the input baseband signal multiplied by a complex constant. The input and output waveforms look different because of the rotation, not because of some non-linear distortion. A common non-linear distortion, such as saturation, would make the outer edges of the trajectory lie on a circle.

Measuring ACPR and PSD

Adjacent Channel Power Ratio (ACPR) is a common measure of the power a transmitter emits outside its allotted frequency band. ACPR is the ratio of the power in an adjacent band to the power in the allotted band.

$$ACPR = \frac{\text{Power in an adjacent band}}{\text{Power in the allotted band}}$$

Regardless of how you choose the frequencies and bands for the ACPR measurement, ACPR is always extracted from the power spectral density (PSD) of the transmitted signal. PSD is a frequency-by-frequency average of a set of discrete Fourier transforms (DFTs) of the baseband signal. Here, the baseband signal is the harmonic-time result of an *envlp* analysis.

The ACPR Wizard

The ACPR Wizard simplifies the complicated calculations needed to measure ACPR. It determines the appropriate *envlp* simulation parameters and `psdbb` function arguments from the information you supply to the ACPR wizard. The ACPR wizard form contains a minimum number of clearly labeled fields so you can easily supply the required information for each field.

After you provide the information and press *Apply* or *OK* in the ACPR wizard*.* The calculated values are used to fill in the *envlp* Choosing Analyses form and the *envlp* Options form. You can then run the simulation and analyze the PSD and ACPR data.

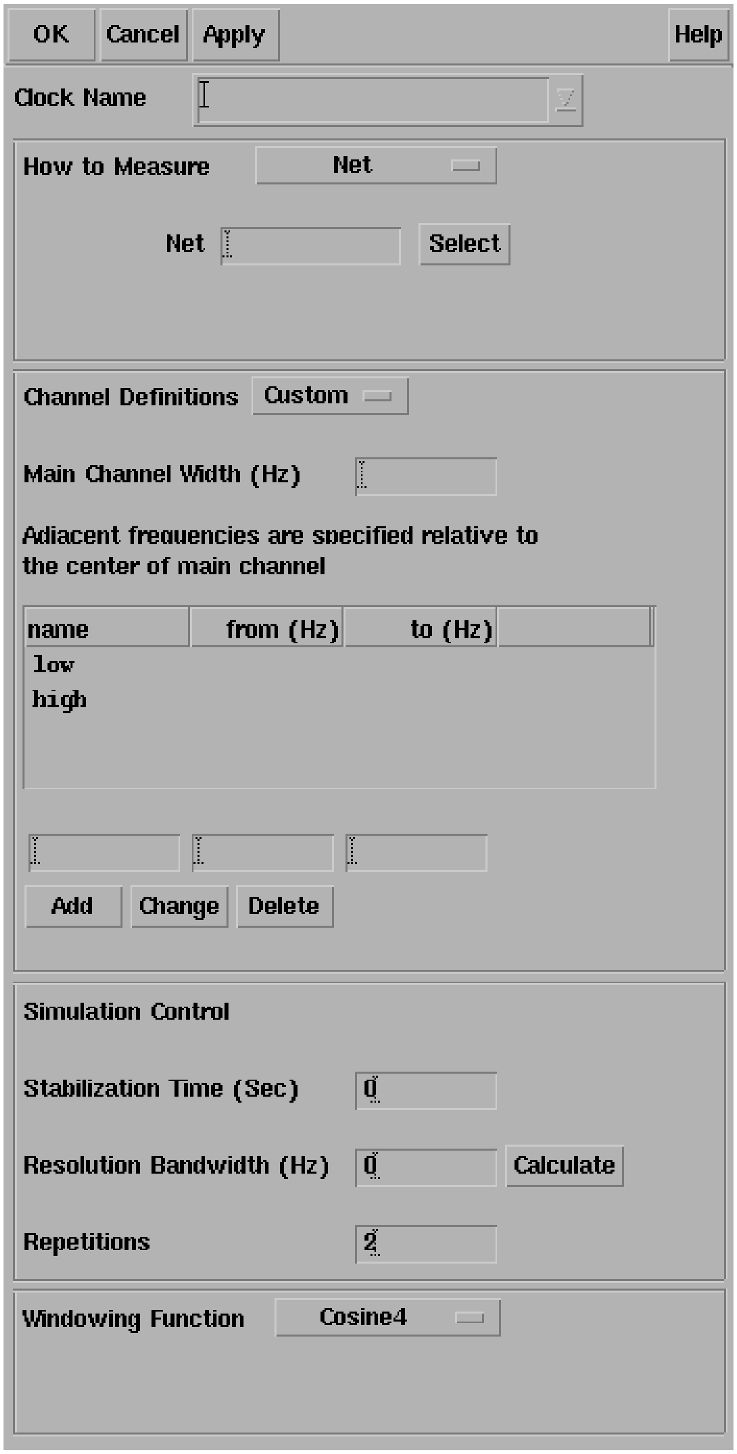
Open the ACPR wizard in one of two ways.

- In the Simulation window, choose *Tools – RF – Wizards – ACPR*

  or

- In the *envlp* Choosing Analyses form, press *Start ACPR Wizard*.

In either case the ACPR Wizard displays.



Measuring ACPR

The ACPR wizard minimizes the number of items you need to supply. Its fields are easy to understand and fill in. Once you provide the ACPR wizard field values, they are used to calculate and fill in the field values on both the *envlp* Choosing Analyses and Options forms. The ACPR wizard computes the appropriate simulation parameters and `psdbb` arguments from the information you enter in the ACPR wizard.

To use the ACPR wizard,

1. Fill in the ACPR wizard form.
2. Click *OK* or *Apply* in the ACPR wizard to calculate field values for both the *envlp* Choosing Analyses and *envlp* Options forms.
3. Run the Envelope Following analysis. When the simulation completes, the *envlp* analysis results are plotted.

This example illustrates how to use the ACPR wizard and set up and run the Envelope Following analysis. This example uses the *EF_example* circuit from your writable copy of the *rfExamples* library.

Note: Before you start, perform the setup procedures described in Chapter 3 if you have not yet set up the writable copy of the *rfExamples* library.

Set up the *EF_example* schematic and the simulation environment.

1. Open the *EF_example* schematic as described in "Opening the EF_example Circuit in the Schematic Window" on page 684.
2. Open the Simulation window as described in "Opening the Simulation Window" on page 685.
3. Set Up the Model Libraries as described in "Setting Up the Model Libraries" on page 686.
4. Edit the *PWL file names* for *PORT0* and *PORT1* as described in "Editing PORT0 and PORT1 in the Schematic Window" on page 687.

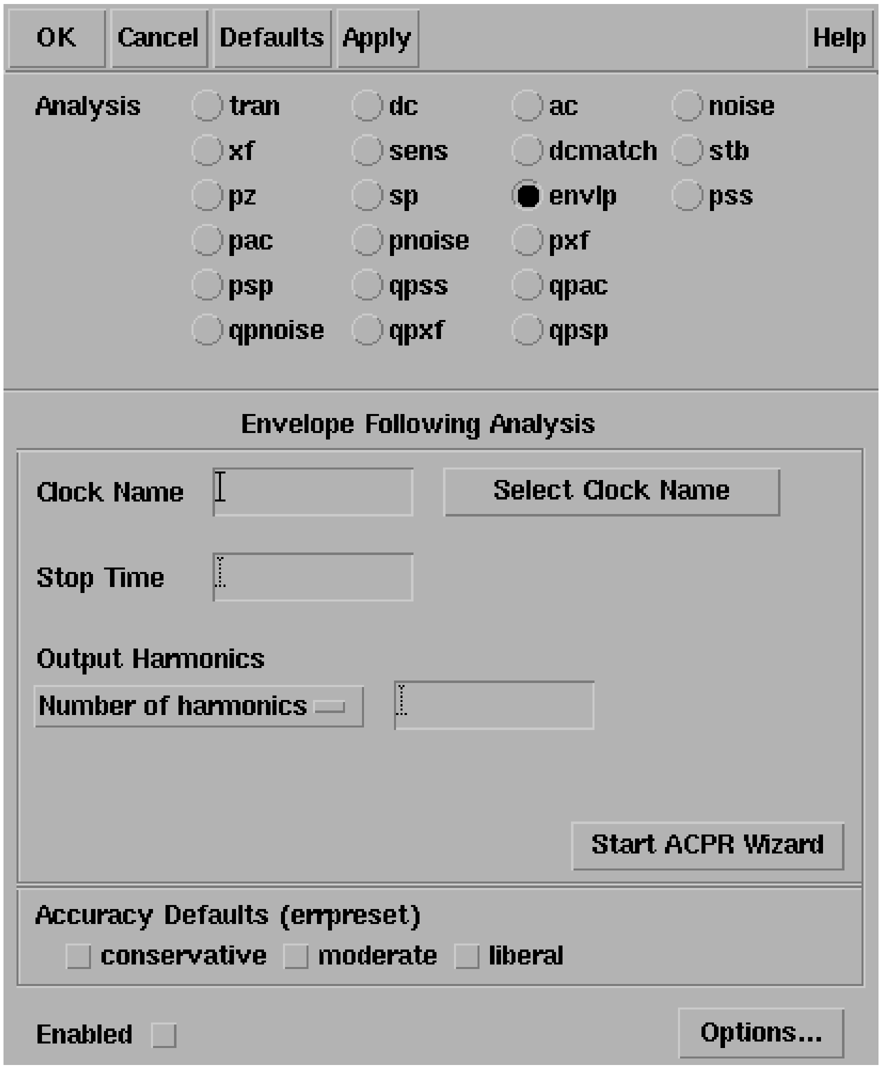
Setting Up the ACPR Wizard and the envlp Analysis

1. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *envlp*.

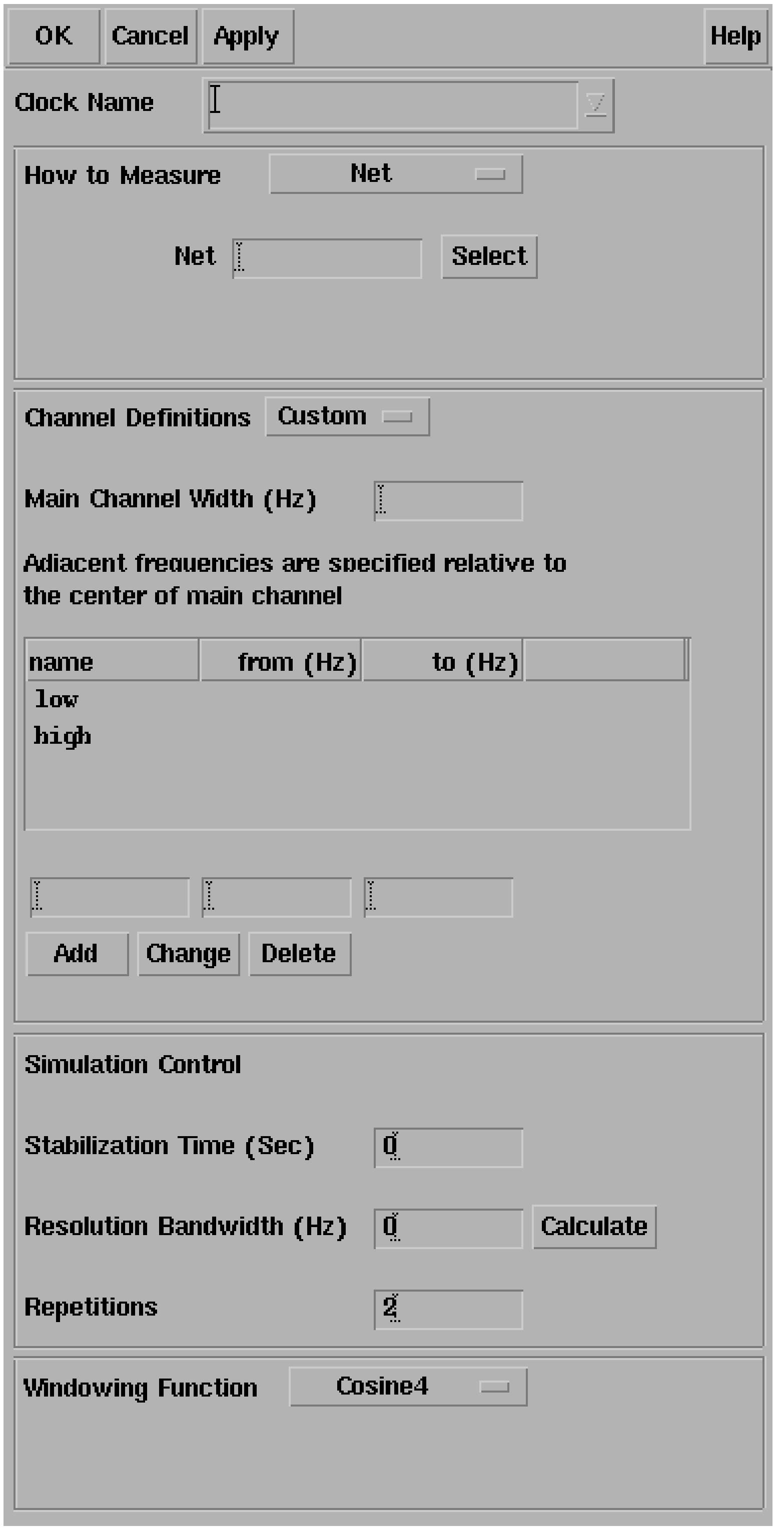
The Choosing Analyses form changes to let you specify values for the Envelope Following analysis.



3. In the *envlp* Choosing Analyses form, click *Start ACPR Wizard*.

Leave the *envlp* Choosing Analyses form open in the background.

The ACPR wizard appears.



4. In the ACPR wizard, do the following.

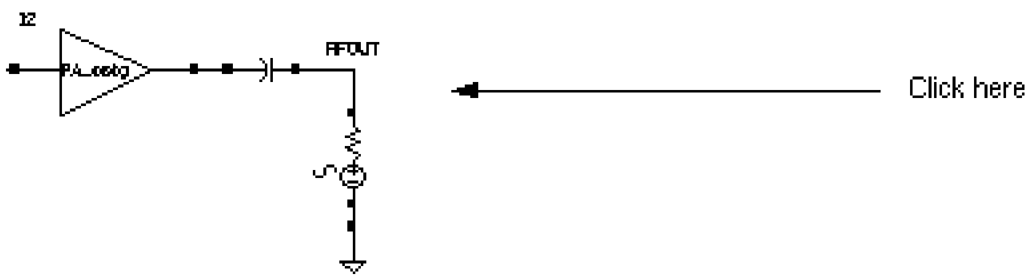
a. In the *Clock Name* cyclic field, select *fff*.

The clock name identifies the source of the modulated signal.

b. In the *How to Measure* cyclic field, select *Net*.

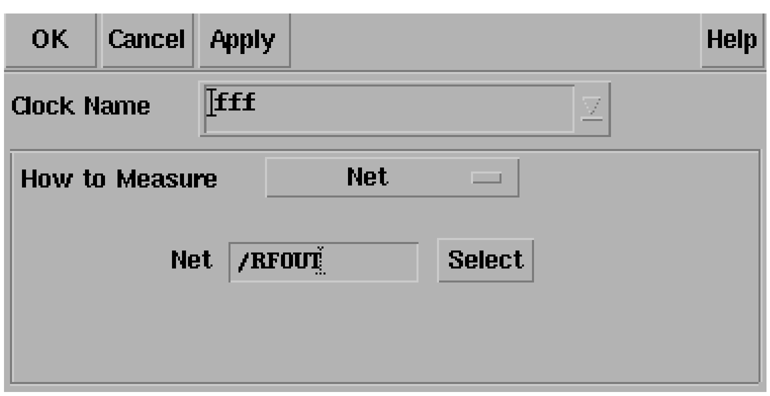
You can measure ACPR for a single *Net* or between *Differential Nets (Net+* and *Net-)*.

c. To select the output net in the Schematic window, click *Select* to the right of the *Net* field. In the Schematic window, select the transmitter output net, *RFOUT*.



The output net name, *RFOUT*, displays in the *Net* field.

The top of the ACPR wizard appears below.

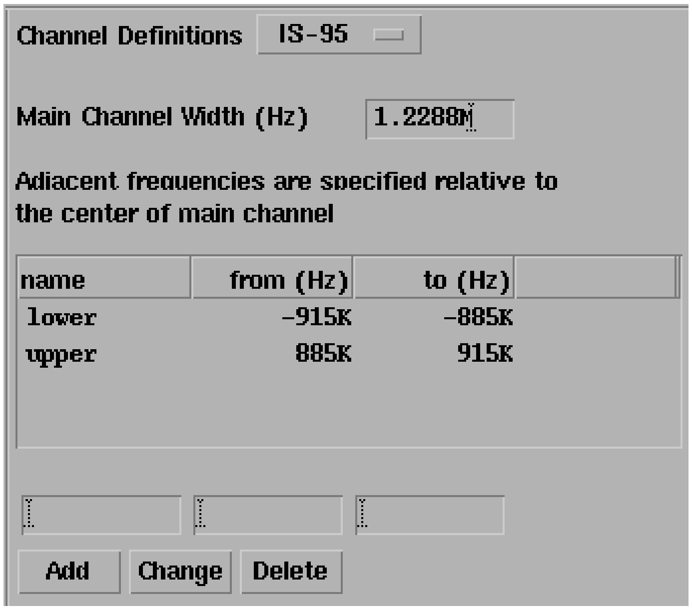


d. In the *Channel Definitions* cyclic field, select the *IS-95* standard.

- ❍ The *Main Channel Width (Hz)* field is calculated as `1.2288M`.
- ❍ The *adjacent frequencies* are determined and display in the list box. Note that adjacent frequencies are specified relative to the center of the main channel.

Channel Definitions presets include *Custom*, *IS-95*, and *W-CDMA*. *Main Channel Width (Hz)* specifies a frequency band in Hz.

The middle of the ACPR wizard appears below.



Use the edit fields and the *Add*, *Change* and *Delete* buttons to modify channel definitions in the list box. You can hand edit or enter adjacent frequency names and upper and lower boundaries. Specify adjacent frequencies relative to the center of the main channel. All channel widths must be greater than zero.

e. In the *Stabilization Time (Sec)* field, enter `72n`.

`72n` displays in the *Stabilization Time (Sec)* field and 2 displays in the *Repetitions* field.

The *Stabilization Time* is the length of time in seconds to wait before using the data for analysis. *Repetitions* is the number of times to repeat the discrete Fourier transform for averaging.

Note: Increasing the number of repetitions makes the power density curve smoother, but at the expense of longer simulation time and increased data file size.

f. To determine the *Resolution Bandwidth (Hz)*, click *Calculate* to the right of the *Resolution Bandwidth (Hz)* field.

`7500` displays in the *Resolution Bandwidth (Hz)* field and `72n` changes to `7.2e-08` in the *Stabilization Time (Sec)* field.

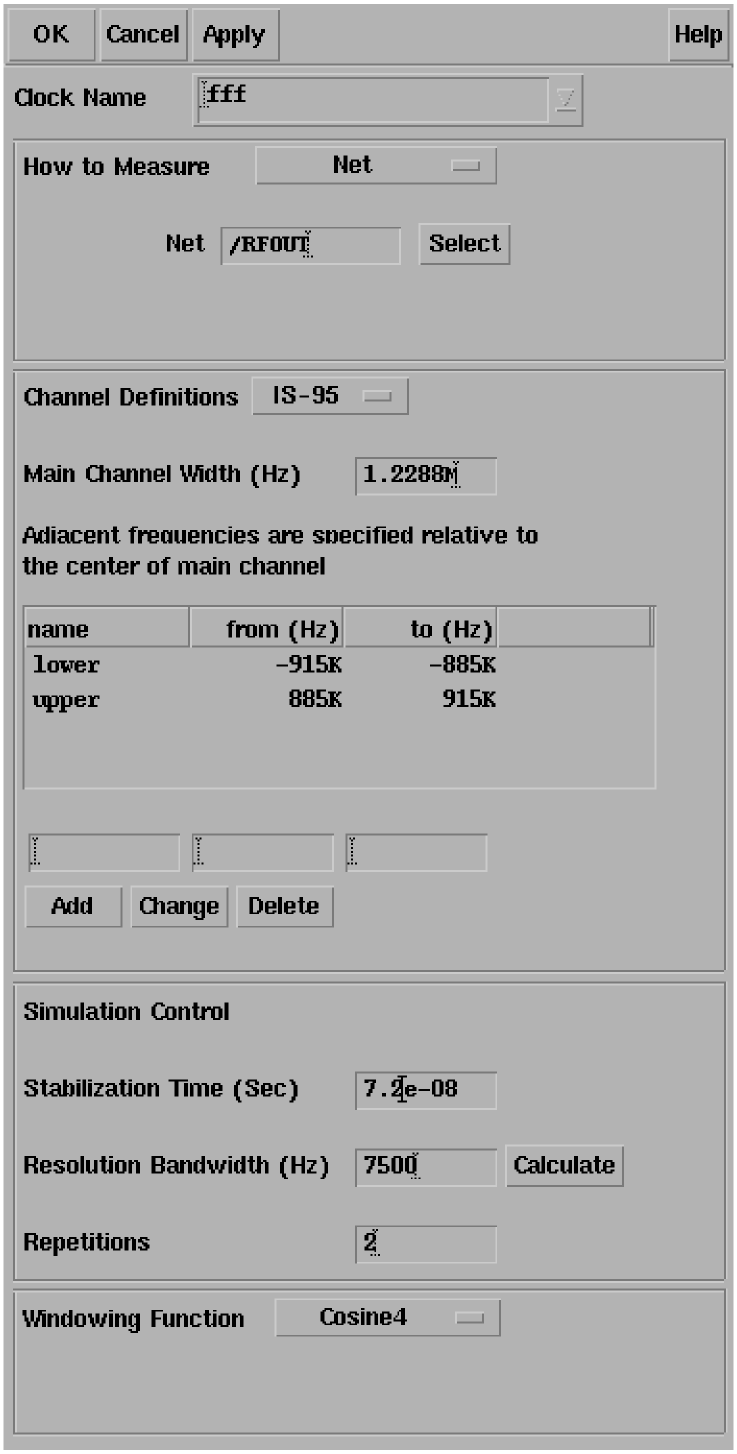
Resolution Bandwidth specifies the spacing of data points on the resulting power density curve, in Hz.

Note: Reducing *Resolution Bandwidth* increases simulation time and data file size.

g. In the *Windowing Function* cyclic field, select *Cosine4*.

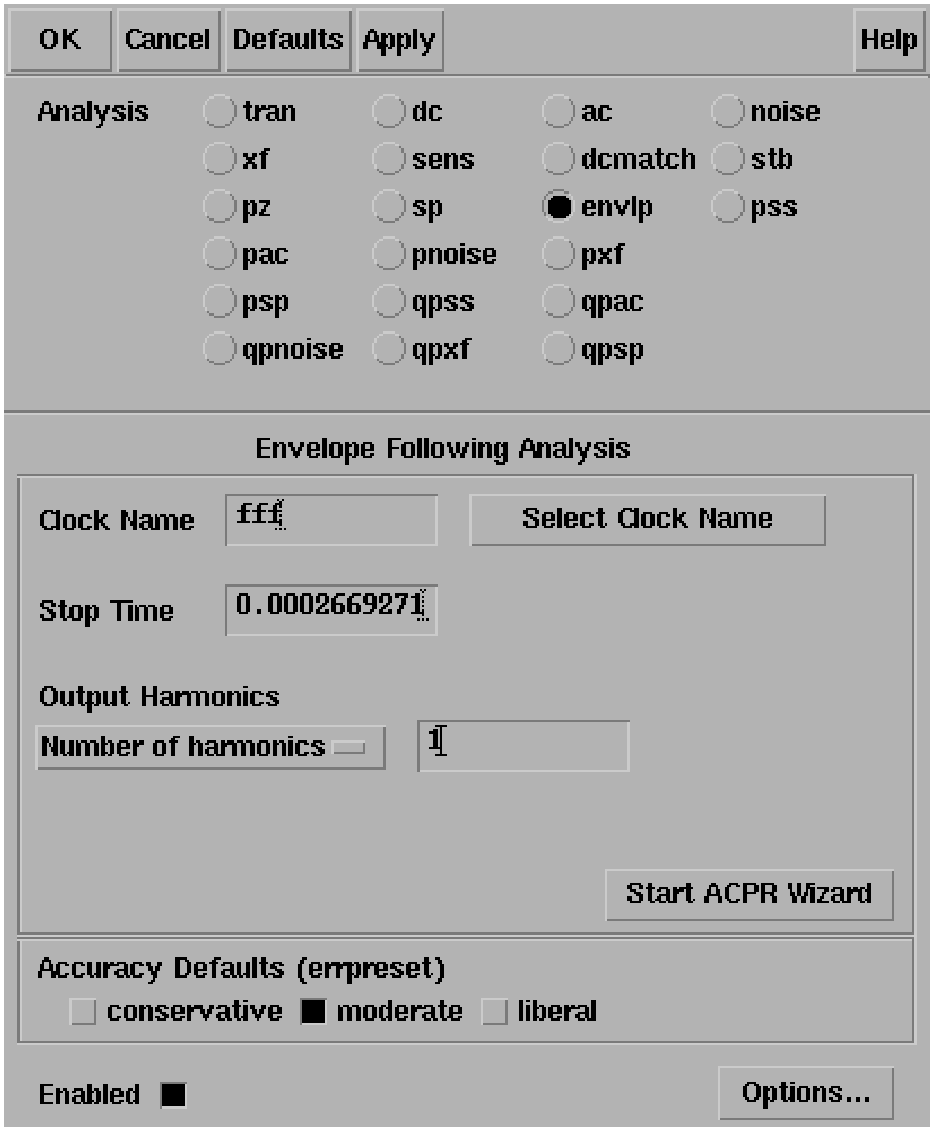
Windowing Function presets include *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser, Parzen, Rectangular* and *Triangular.*

The ACPR Wizard looks like the following.



5. In the APCR Wizard, click *Apply*

When you click *OK* or *Apply* in the ACPR wizard, values are calculated and appear in the *envlp* Choosing Analyses form.



Values in the *envlp* Choosing Analyses form are as follows.

- The *Clock Name* is the same in both forms. In this case *fff*.
- *Stop Time* for *envlp* is calculated.
- For the *envlp* analysis O*utput Harmonics, Number of Harmonics* is selected in the cyclic field and the *Number of Harmonics* is set to 1.
- The *envlp* analysis is *Enabled.*

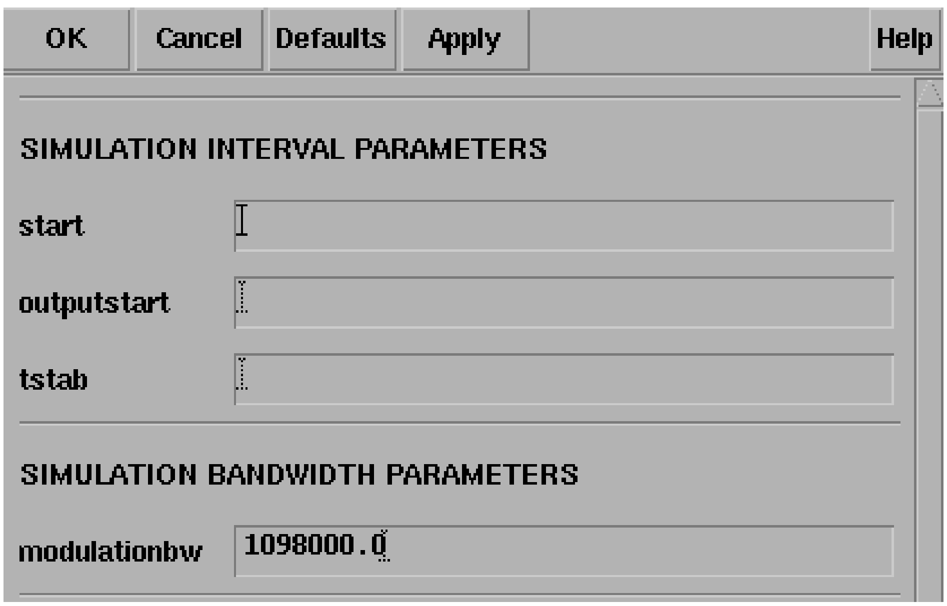
- ❑ The *envlp* option *start* is set to blank.
- ❑ The *envlp* option *modulationbw* is calculated
- ❑ The *envlp* option *strobeperiod* is calculated

Note: You can modify values on the *envlp* Choosing Analyses and Options forms but your changes are not propagated back to the ACPR wizard.

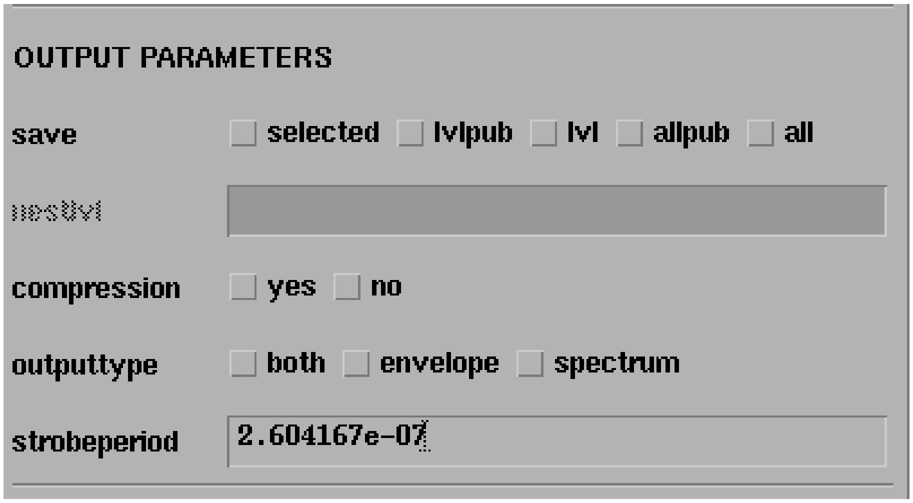
6. In the *envlp* Choosing Analyses form, click *Options* to open the *envlp* Options form.

   Notice that values for the *modulationbw* and *strobeperiod* parameters are calculated and the *start* parameter is blank.

   The *start* and *modulationbw* parameters.



The *strobeperiod* parameter.

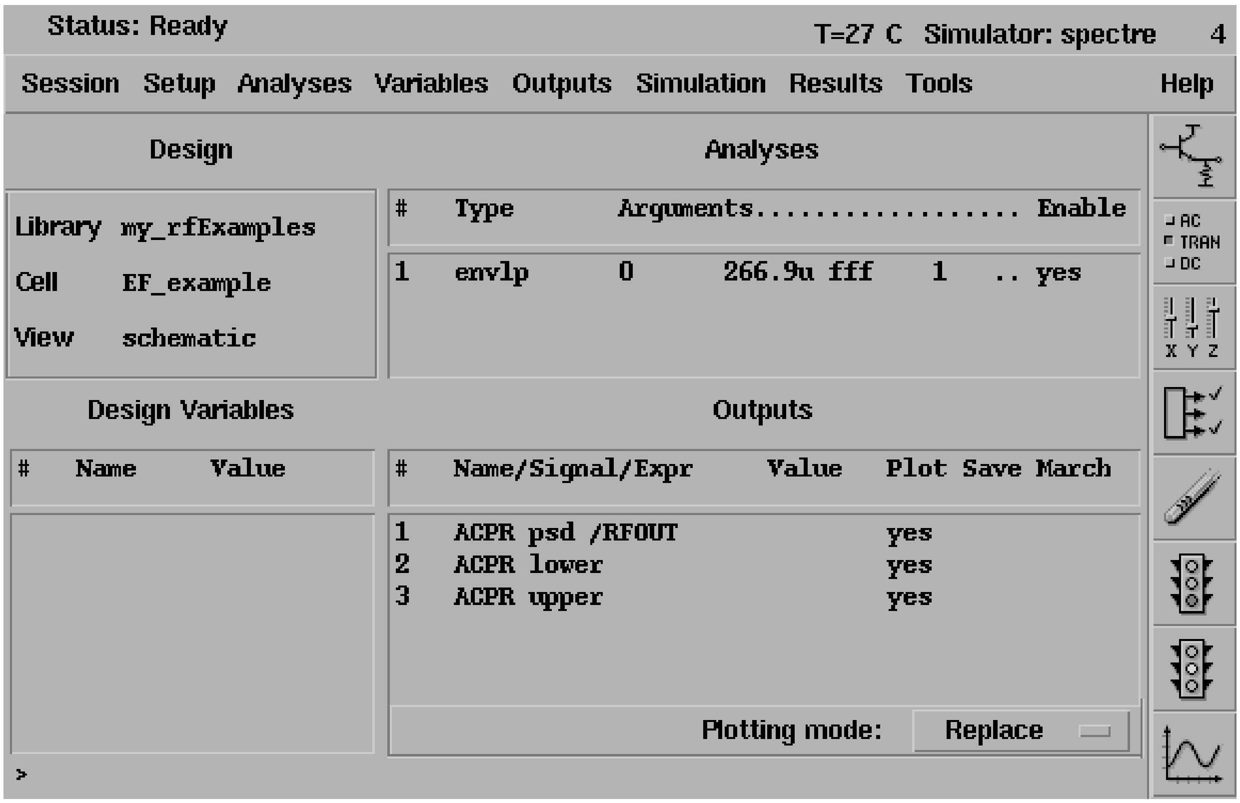


7. Click *OK* in the Envelope Following Options form.
8. In the *envlp* Choosing Analysis form, Select *Moderate* for *Accuracy Defaults (errpreset)*.
9. Verify that *Enabled* is highlighted and click *Apply* in the *envlp* Choosing Analysis form.
10. Click *OK* in the *envlp* Choosing Analysis form.

    The Choosing Analysis form closes.
11. Click *OK* in the ACPR Wizard.

    The ACPR Wizard closes.

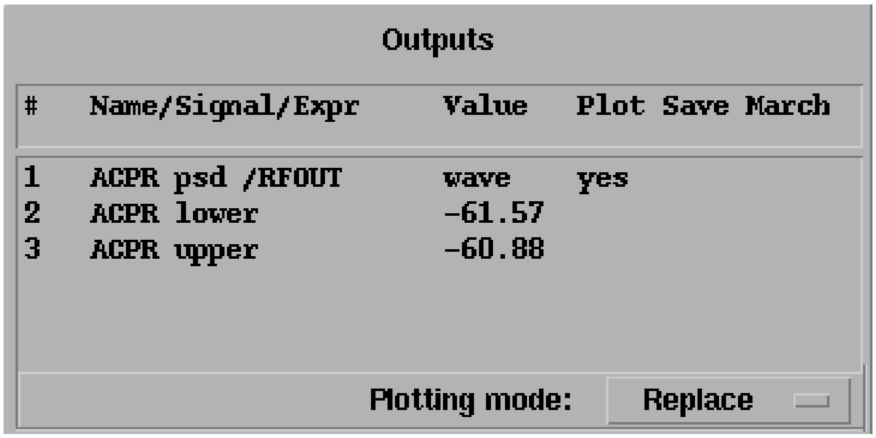
12. The Simulation window reflects the *Analysis* and *Outputs* information from the ACPR wizard, the *envlp* Choosing Analyses form and the calculations which resulted.



13. In the Simulation window, choose *Simulation – Netlist and Run*.

Look in the CIW for messages saying that the simulation has started and completed successfully. Watch the simulation log file for information as the simulation runs.

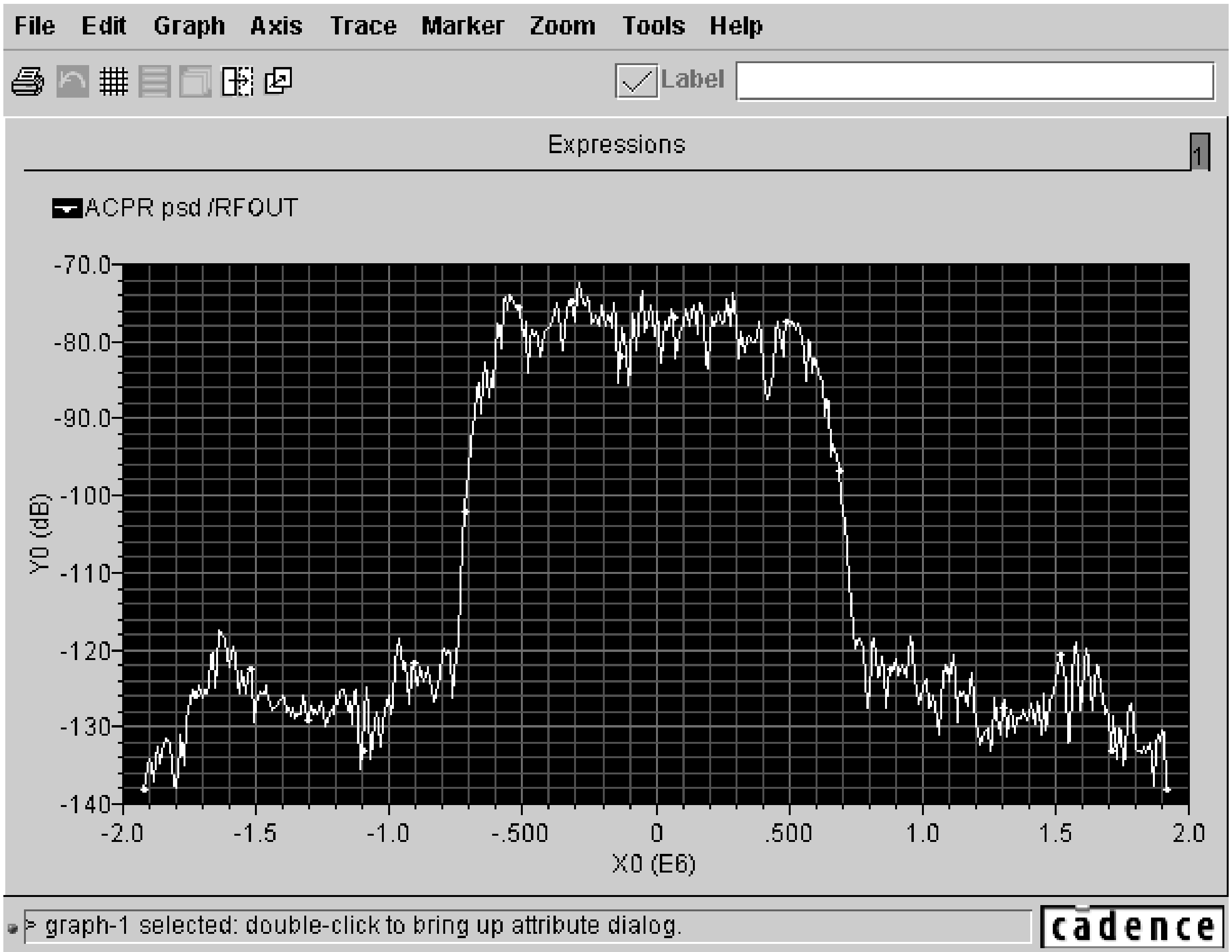
When the simulation successfully completes, an ACPR value for each channel appears in the *Value* column in the Simulation window *Output* section.



- The *ACPR* `psd` */RFOUT Value* is listed as a *wave* which is plotted. This is the PSD plot.
- The ACPR value for the *ACPR lower* channel is -61.57.
- The ACPR value for the *ACPR upper* channel is -60.88.

  When the simulation finishes, the PSD plot displays as in Figure 10-2.

Figure 10-2 PSD Plot Generated by the ACPR Wizard



Estimating PSD From the Direct Plot Form

The power spectral density (PSD) is always estimated because the information riding on the carrier is a stochastic process and the Fourier transform of a stochastic process is ill defined. No matter how you chose to define the spectral nature of a stochastic process, it always involves an averaging process. Any empirically derived average is an estimate because you can never take an infinite number of samples.

PSD is a frequency-by-frequency average of a set of discrete Fourier transforms (DFTs) of the baseband signal. Here, the baseband signal is the harmonic-time result of an *envlp* analysis.

This example shows how to estimate PSD given the results of the *envlp* analysis you just performed.

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

2. In the Top of the Direct Plot form, do the following:

   a. Choose *New Win* for *Plotting Mode*. This will plot PSD in a new window.

   b. Highlight *envlp* for *analysis.*

   c. Highlight *Voltage* for *Function*.

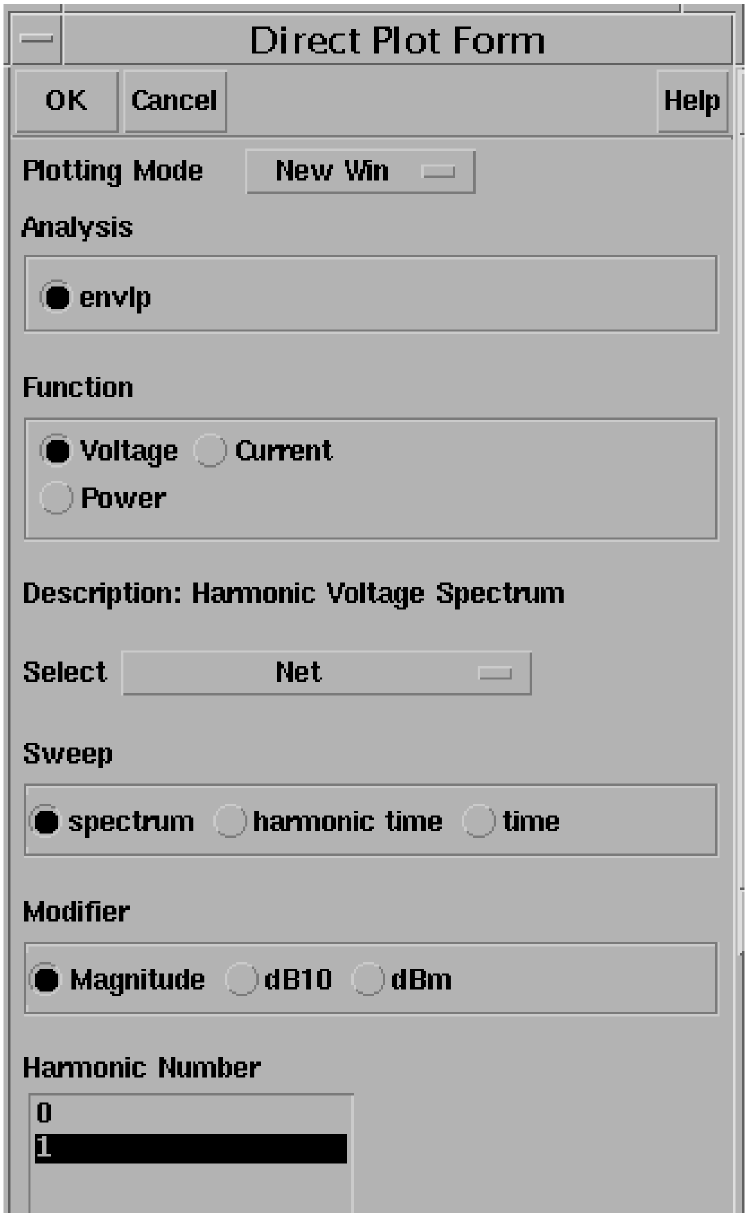
   d. Highlight *spectrum* for *Sweep*.

      The *Power Spectral Density Parameters* section opens at the bottom of the form.

   e. Highlight *Magnitude* for *Modifier*.

   f. Choose *1* for *Harmonic Number*.

The top of the Direct Plot form looks like the following.



3. In the Bottom of the Direct Plot form, enter information for the *Power Spectral Density Parameters*.:

   g. Press *Get From Data* to calculate the values for the *Time Interval* values, *From* is `0.0` and *To* is `0.0002669271`.

   h. Type `5M` for *Nyquist half-bandwidth*.

i. Type `.1M` for *Frequency bin width*.

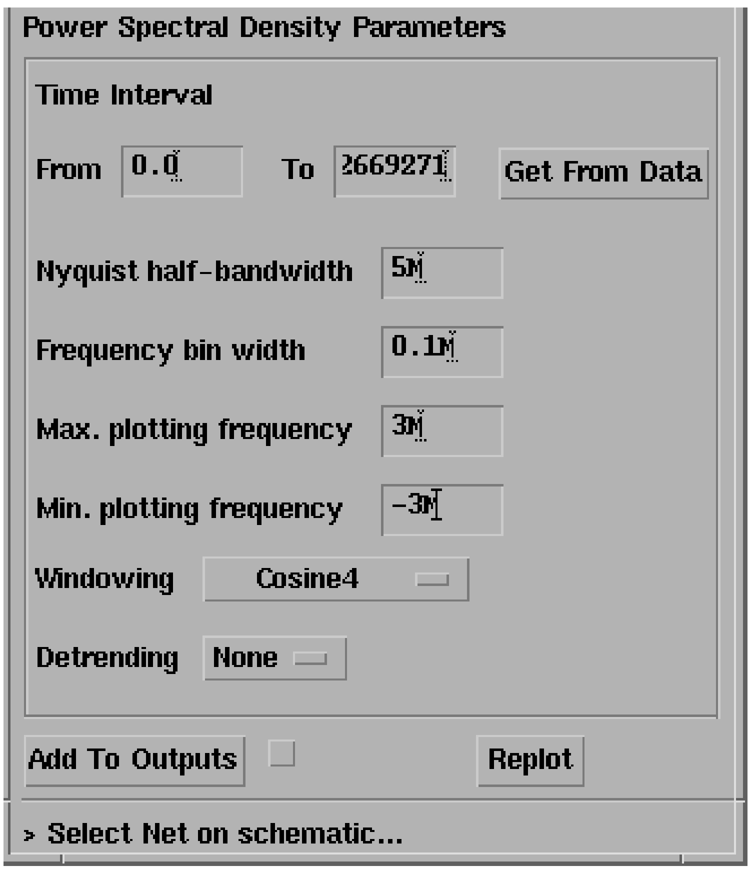
j. Type `3M` for *Max. plotting frequency*.

k. Type `-3M` for *Min. plotting frequency*.

l. In the *Windowing* cyclic field, select *Cosine4*.

m. In the *Detrending* cyclic field, select *None*.

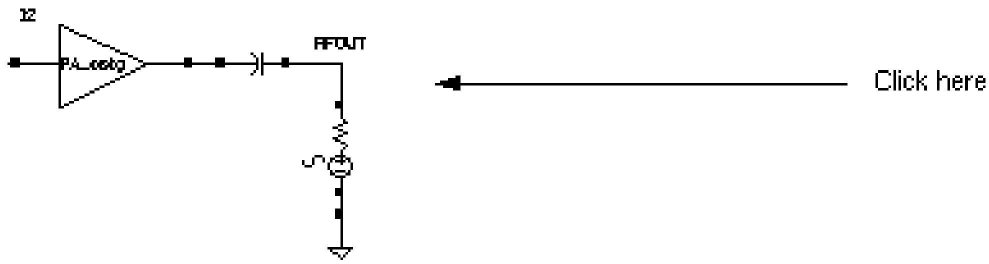
The bottom of the *envlp* Direct Plot form is as follows.



n. Following the prompt at the bottom of the form,

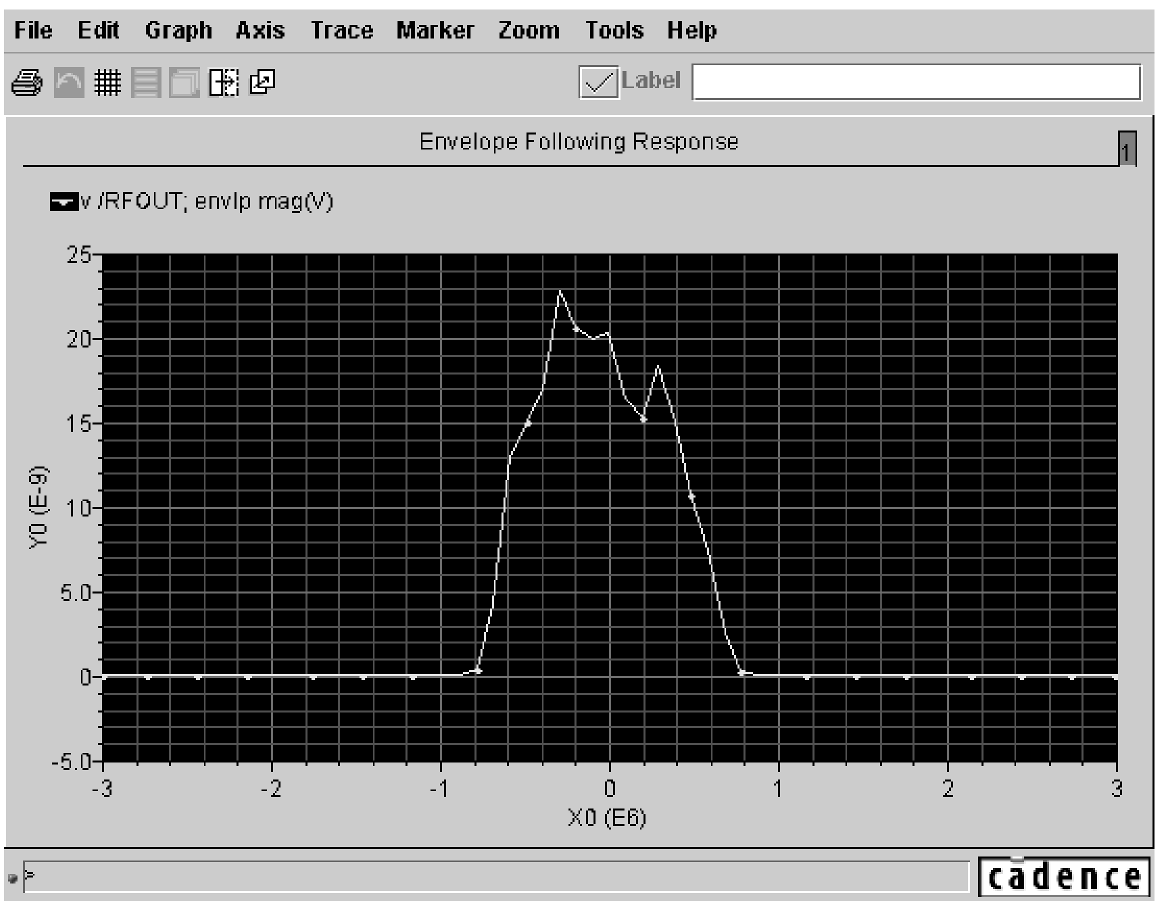
> *Select Net on schematic.*

Click on the transmitter output net (*/RFOUT*).



4. Click *Replot*.

The PSD plot displays.



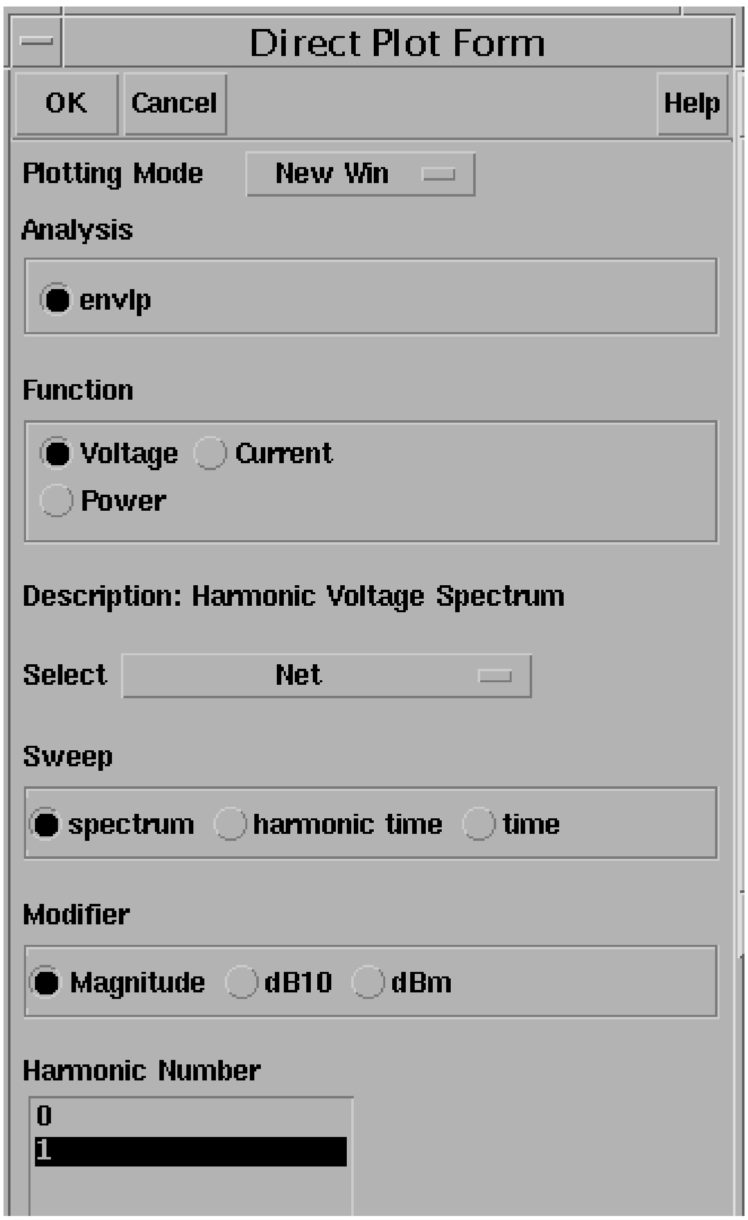
To add the modified PSD plot, make the following changes in the Direct Plot form.

1. In the Direct Plot form, make the following changes:

a. Choose *Append* for *Plotting Mode*. This will plot the modified PSD waveform in the same window.

b. Highlight *dB10* for *Modifier*.

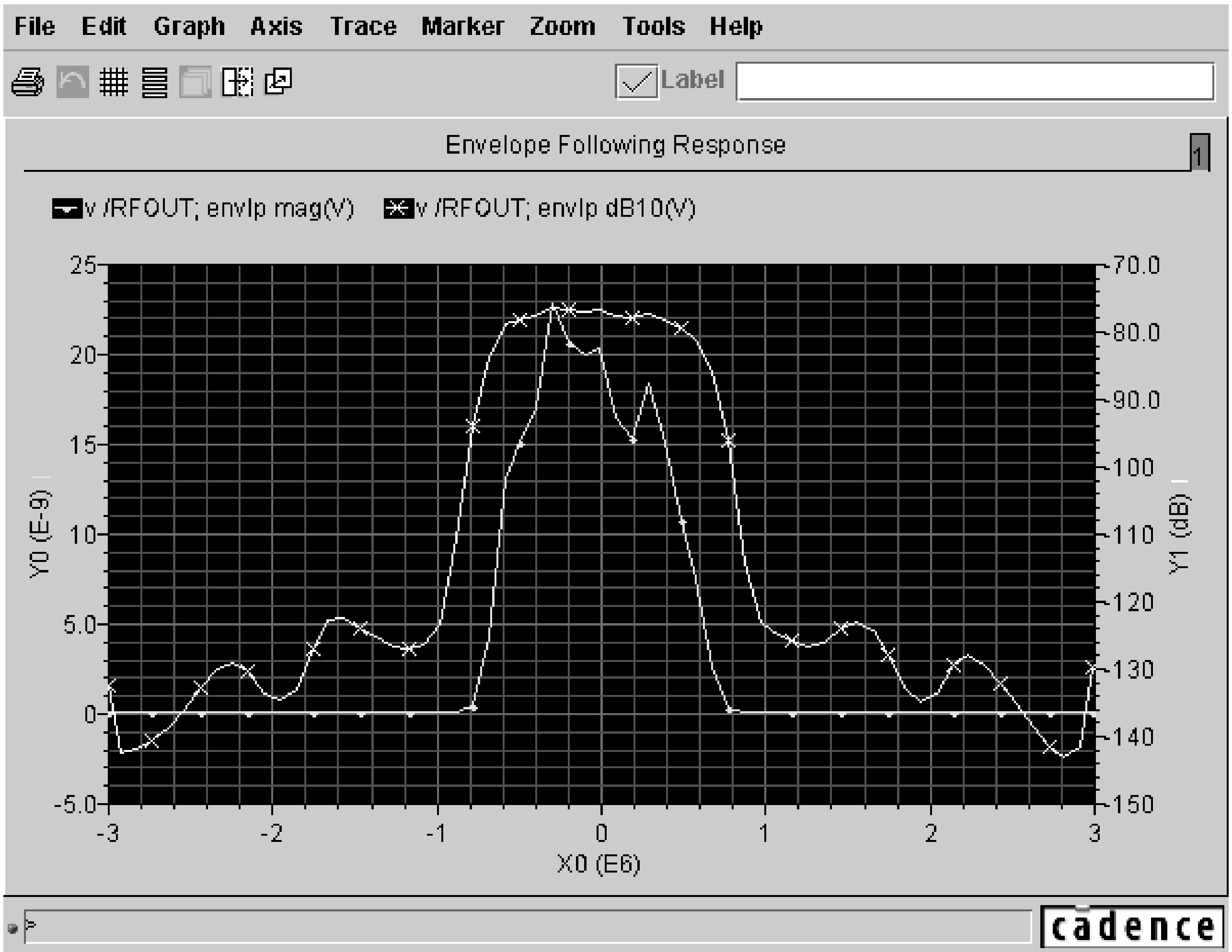
The top of the Direct Plot form looks like the following.



c. Click *Replot* at the bottom of the *envlp* Direct Plot form.

The modified PSD plot is appended to the original PSD plot, as shown in Figure 10-3.

Figure 10-3 Estimated PSD from the envlp Direct Plot Form



Compare Figure 10-3 with the PSD plot measured with the ACPR Wizard as shown in Figure 10-2 on page 737.

Reference Information for ACPR and PSD Calculations

The process outlined in this section is complex largely because the *envlp* analysis parameters are not directly related to the Waveform Calculator `psdbb` function arguments.

- The *envlp* parameters include nyquist half-bandwidth, frequency bin width and time interval.
- The *psdbb* function arguments are the total number of samples, the window size and the bin-width. All three parameters are in terms of the number of DFT time samples.

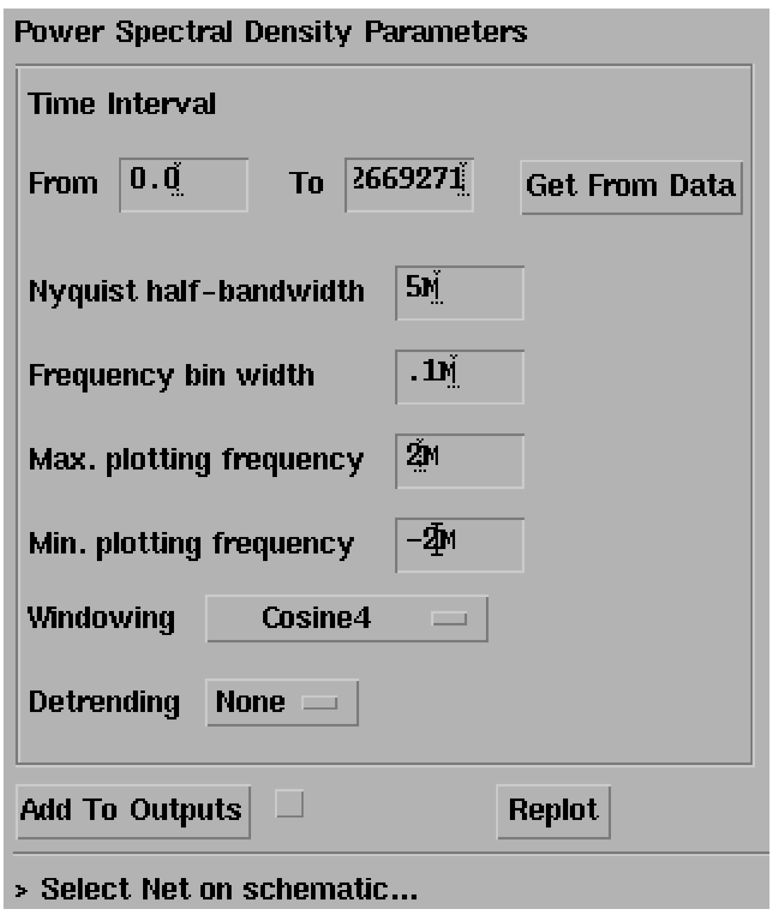
However, to make optimum use of the simulation data, it is necessary that the simulation parameters and the `psdbb` function arguments be compatible.

The Power Spectral Density (PSD) Parameters

When you select *spectrum* for *Sweep* in the Direct Plot form, the *Power Spectral Density Parameters* section opens at the bottom of the Direct Plot form. The *Power Spectral Density Parameters* section displays the analysis parameters that control the PSD estimate.

The *Power Spectral Density Parameters* section is shown in Figure 10-4.

Figure 10-4 PSD Parameters on the envlp Direct Plot Form



The PSD Parameters are described in Table 10-1.

Table 10-1 Power Spectral Density Parameters From the Direct Plot Form

PSD Parameter	Description
Time Interval	The *Time Interval* parameters specify the time record to analyze in the frequency domain. The time interval should be long enough to support the required frequency resolution. Use *Get From Data* to calculate the *From* and *To* values from the ACPR wizard data.
Nyquist half-bandwidth	The *Nyquist half-bandwidth* parameter is half the sampling frequency used in the DFTs. Make the nyquist half-bandwidth large enough to prevent aliasing. The true spectrum of the baseband signal should be negligible at this frequency and beyond it.
Frequency bin width	The *Frequency bin width* parameter specifies the required frequency resolution. ■ When this value is too small; the resulting PSD will look noisy and the PSD will have a jagged appearance. ■ When this value is too large; the resulting PSD might be softened when the spectrum should have sharp edges. *Windowing Function* presets include *Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hamming, Hanning, Kaiser,Parzen,Rectangular* and *Triangular.*
Windowing	The *Windowing* parameter specifies which windowing function to apply before performing the DFTs.
Detrending	The *Detrending* parameter has one of three values: *None*, *Mean*, or *Linear*.

If necessary, the waveform is first interpolated to generate evenly spaced data points in time. The data point spacing is the inverse of the DFT sampling frequency. The PSD is computed by

- Breaking the time interval up into overlapping segments
- Multiplying each segment, time point by time point, by the specified *Windowing* function.

Windowing reduces errors caused by a finite time record. It is impossible to work with an infinite time record. Direct use of an unwindowed finite time record is equivalent to multiplying the infinite record by a rectangular pulse that lasts as long as the data record. Multiplication in the time domain corresponds to convolution in the frequency domain. The Fourier transform of a rectangular pulse is a *sinc* function. Considering the frequency domain convolution, the side lobes of the sinc function cause parts of the true spectrum to leak into the frequency of interest, that is, the frequency of the main lobe. Ideally, the sinc function would be a Dirac delta function but that requires an infinite time record. Good window functions have smaller side lobes than the sinc function.

The DFT is performed on each windowed segment of the baseband waveform. At each frequency, the DFTs from all segments are averaged together. Fewer segments means fewer data points in the average at a particular frequency. The length of each segment is inversely proportional to the *Frequency bin width*, which is why a small *Frequency bin width* produces a jagged PSD. A smaller *Frequency bin width* means a longer time segment.

Fewer long segments fit into the given time interval so there are fewer DFTs to average together. In the extreme, there is only one segment and no averaging. Without averaging, the PSD is the square of the magnitude of the DFT of a stochastic process. At the other extreme, large *Frequency bin widths* produce lots of points to average at each frequency but there are fewer frequencies at which to average because fewer large bins fit into the Nyquist frequency. The PSD is smoother but it does not have as much resolution.

PSD is always estimated because the information riding on the carrier is a stochastic process and the Fourier transform of a stochastic process is ill defined. No matter how you chose to define the spectral nature of a stochastic process, it always involves an averaging process. Any empirically derived average is an estimate because you can never take an infinite number of samples.

The Waveform Calculator *psdbb* function performs a discrete Fourier transform (DFT) on the voltage curve which produces the power spectral density (PSD) curve. Integrate the PSD curve to calculate power in the channel.

The Waveform Calculator psdbb Function

The Waveform Calculator *psdbb* function, which estimates PSD, derives the function parameters it requires from the PSD parameters and values you supply in the envlp Direct Plot form.

- *Nyquist half-bandwidth*
- *Frequency bin width*
- *Time Interval*

See Table 10-1 for more information about the PSD parameters on the Direct Plot form.

The Waveform Calculator *psdbb* function parameters are

- The total number of samples
- The window size
- The bin-width

All three parameters are in terms of the number of DFT time samples.

The calculations defined in Table 10-2 generate the `psdbb` parameters. The calculated *psdbb* parameters are printed in the CIW window.

Table 10-2 Parameter Values Calculated by the psdbb Calculator Function

psdbb Calculation	Source of the Data
$L = To - From$	*To* and *From* are the values from the *Time Interval To* and *From* fields on the *envlp* Direct Plot form.
f_{max}	*Nyquist half-bandwidth* and *Frequency bin width* are values from these fields on the *envlp* Direct Plot form.
`#bins = floor(L*binwidth)`	`#bins >= 1`. Here, *floor* means *to take the integer part of*, i.e. truncate to the nearest integer.
$2^m * (\#bins) > 2 * L * f_{max}$	Compute the smallest `m`.
$windowsize=2^m$	
`#bins * windowsize`	Compute the number of samples.

You might want to use the `psdbb` function directly when strobing time-harmonic results to eliminate interpolation error. Use of the `psdbb` function is described in the waveform calculator documentation.

Calculating ACPR

This section describes how to Calculate the ACPR (Adjacent Channel Power Ratio) by hand. The ACPR Wizard performs these calculations for you.

ACPR is measured with respect to any x_1 and x_2, as y_1 - y_2. You can calculate the ACPR for any two x-axis values by subtracting their associated y-axis values.

For example, given the following two spectral parameters,

- The adjacent channel is 2.5 MHz from the carrier
- You choose to define ACPR in terms of power at just two frequencies,
  - 2.5 MHz
  - 0 MHz

Use the cursor to determine the RF output power at 2.5 MHz and 0 MHz frequencies. As you slide the cursor along the curve, you read the X and Y values off the top of the Waveform window:

```
X:2.5M    Y:-115.8        v /RFOUT;envlp mag(V)
```

and

```
X:0    Y:-77.33        v /RFOUT;envlp mag(V)
```

So

- The power at 2.5 MHz equals -115.8 dB
- The power at 0 MHz equals -77.33 dB

Remember that the horizontal scale is frequency offset from the carrier fundamental. The difference between these measurements equals -38.47 dB.

$$ACPR = -115.8 - (-77.33) = -38.47$$

For this definition of ACPR, ACPR = -38.47 dB.

Other definitions of ACPR are possible. You can compute them by adjusting the spectral parameters and applying the waveform calculator to the spectral plot.

Calculating PSD

For a constant baseband signal, calculate power spectral density as shown in Table 10-3.

Table 10-3 Power Spectral Density for a Constant Baseband Signal

Baseband signal	$1 + j \times q$ where $j = \sqrt{-1}$
Passband signal	$i \times \cos(w \times t) - q \times \sin(w \times t)$
Baseband signal power	$i \times i + q \times q \qquad volts \times volts$
Passband signal power	$i \times \frac{i}{2} + q \times \frac{q}{2} = (1/2) \times (baseband""power)$

Envelope Spectral analysis computes these values per Hz. You can also express these values as

$$(rms""\text{passband}""\text{volts}) \times (rms""\text{passband}""\text{volts}) / (Hz)$$

Envelope Following Time-Harmonic analysis computes peak volts because they can be directly compared with the modulating signals.

For this example

- You see a waveform displayed in the Waveform window as V^2 / (Hz) versus frequency.
- You can think of this as

$$(rms""\text{passband}""\text{volts}) \times (rms""\text{passband}""\text{volts}) / (Hz)$$

Important

The displayed spectrum is the estimated PSD of the complex envelope divided by two. The division by two is included because the envelope is expressed in units of peak carrier volts, but power in the carrier equals the square of the peak divided by two. It is convenient to express the envelope in peak units because you can then directly compare it against an input baseband signal.

PSD and the Transmitted Baseband Signal

When you measure ACPR, it is crucial that you drive the transmitter with the proper baseband signals. The baseband signals driving the transmitter dominate the transmitted PSD. In most cases, the baseband signals are produced by digital filters so the digital filters constrain the spectrum of the input baseband signal. Distortion in the transmitter causes the spectrum to grow where it should not, hence the need for an ACPR measurement.

It is not practical to model digital filters in SpectreRF because SpectreRF cannot simulate state variables inside Verilog®-A modules. Consequently, for now, you must pre-compute and store the baseband inputs and then read them into the SpectreRF analysis through *ppwlf* sources found in the *analogLib*, as shown in "Computing the Spectrum at the Adder Output" on page 751. The *ppwlf* sources also read SPW format, so you can also generate and record the input baseband waveforms using SPW.

The *rfLib* contains three sets of stored baseband waveforms, *cdma*, *dqpsk*, and *gsm*. These waveforms were created with the baseband signal generators in the *measurement* category of the *rfLib*.

If you want to measure ACPR with the noise floor much more than 40 dB below the peak of the output power spectral density, you must create baseband drive signals with a noise floor at or below the required noise floor. If you use a DSP tool such as SPW to create the signals, the filters in the baseband signal generator must operate perhaps hundreds of times faster than those in the actual generator.

Note: Sometimes an ACPR specification exceeds the ACPR of the baseband drive signals. To see if the transmitter meets specifications in that event, it must be driven with an unrealistic baseband signal. Otherwise the signals will not have enough resolution. The noise floor depends heavily on interpolation error.

Note: The Fourier analysis used to compute the power spectral density uses evenly spaced time points. If data does not exist at one of the Fourier time points, the Fourier algorithm must interpolate to create the missing Fourier time point.

You can strobe the harmonic time results to eliminate interpolation of the output but you cannot eliminate interpolation of the baseband drive signals. The only way you can reduce interpolation errors at the input is to use ultra-high-resolution drive signals so that no matter where the interpolation occurs, the error is small. For now, you must generate the ultra-high-resolution drive signals yourself.

Computing the Spectrum at the Adder Output

The following results were obtained by

1. Generating high resolution baseband signals in SPW
2. Storing the high resolution baseband signals
3. Reading the high resolution baseband signals into an *envlp* analysis through the *pwl* sources

The SPW FIR filters operate at 300 times the chip rate.

Figure 10-5 shows the spectrum at the output of the adder. In this circuit, the RF signal is undistorted at the adder output. The spectrum was computed three times. Each time with a different value for *number of samples*.

The envlp analysis was run with the following parameters:

- *reltol* set to `1e`$^{-5}$
- *strobeperiod* set to `100` ns

Note: If the *strobeperiod* does not equal an integer number of clock cycles, it is internally truncated to an integer number of clock cycles.

For the `psdbb` function

- The *Time Interval* was from `100 us` to `1 ms`
- The *window size* was `1024` samples.

  Note: *Window size* must be a power of 2. If you enter a value that is not a power of 2, the `psdbb` function will truncate the value to the nearest power of 2.
- The *windowing* function was *Hanning*.

With *strobeperiod* set to `100` ns (100 clock cycles), the simulation produced 9000 evenly spaced samples inside the time interval. The spectra were calculated using 8500, 8750, 9000, 9250, and 9500 points for the *number of samples*. Figure 10-5 clearly shows interpolation errors when the number of samples used in the `psdbb` function does not equal the actual number of samples.

Figure 10-5 Adder Output Spectrum Computed with Three Different Numbers of Samples

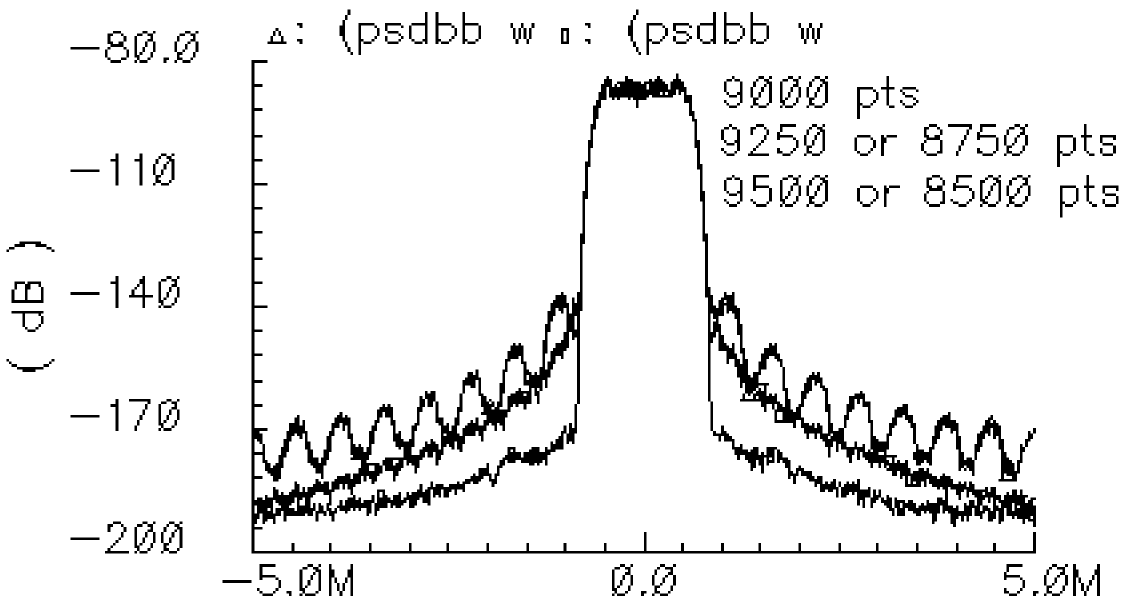
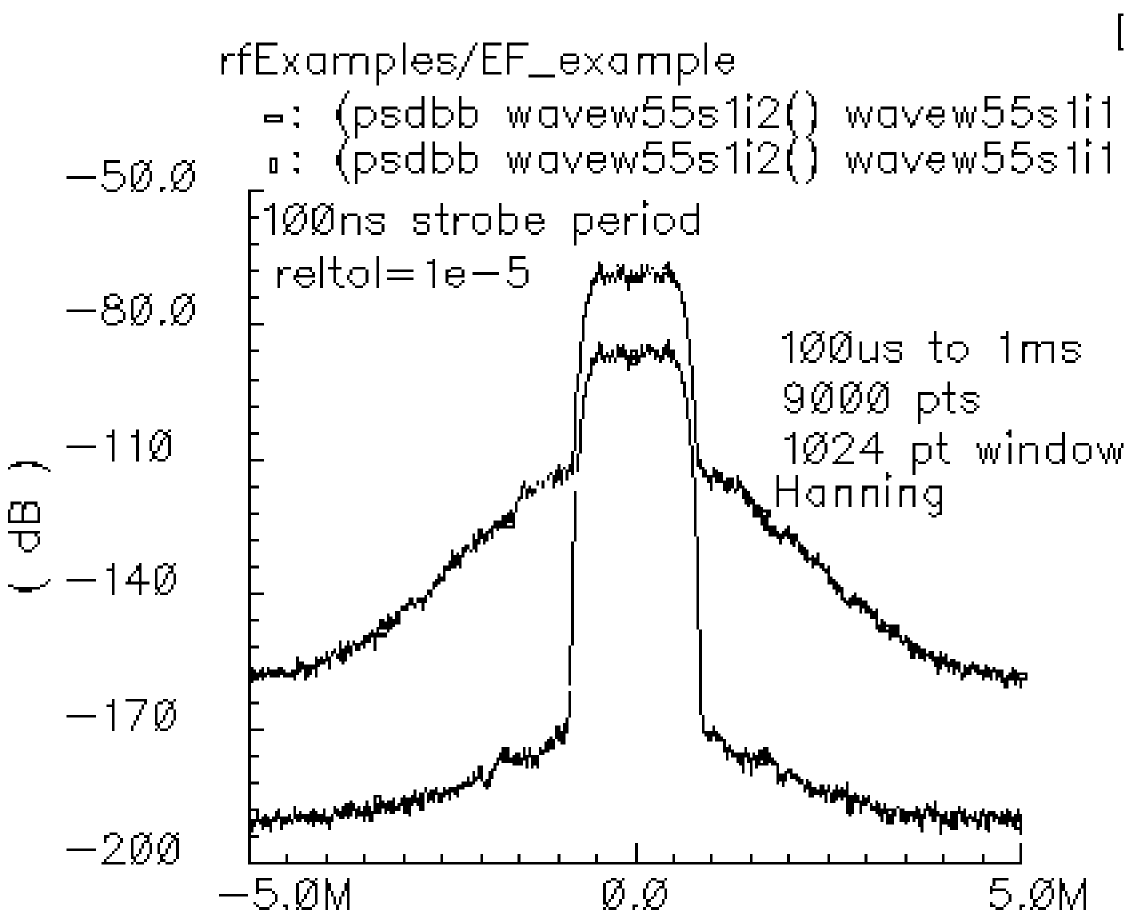


Figure 10-6 compares the input and output PSDs using the high resolution drive signals and the following set of `psdbb` parameters:

- *From* is `100` u
- *To* is `1` m
- *Number of Samples* is `9000`
- *Window size* is `1024`
- *Window Type* is *Hanning*

In Figure 10-6, the upper PSD plot is the power amplifier output and clearly shows spectral regrowth when compared to the input PSD plot.

Figure 10-6 Comparison of Input and Output PSD plots



Measuring Load-Pull Contours and Load Reflection Coefficients

This section describes how to generate load pull contours and how to determine whether you must redesign the input matching network for the optimal load.

A *load-pull contour* is a set of points on a Smith chart representing all the loads that dissipate a given amount of power. Load-pull contours help you match a load to a power amplifier for maximum power transfer. Just as a topographical map shows a hiker where the mountain peak lies and how steep the climb is, load-pull contours show which load dissipates the most power and how sensitive that power is to small load perturbations.

To properly use Load-Pull results, you must understand how the Load-Pull feature defines load reflection coefficients. Load reflection coefficients are computed using the PSS analysis.

A *load reflection coefficient* is computed from the load impedance, which is computed as the ratio of the voltage across the load to the current flowing into the load at the RF carrier frequency. Suppose the RF carrier is 1 GHz and the load waveforms are distorted. The impedance is not computed from small-signal perturbations about an operating point. Rather, the impedance is computed as the ratio of the 1 GHz components of the load voltage and current waveforms simulated by a PSS analysis. Because the load is passive and linear, the load reflection coefficient computed in this manner equals the small-signal, or incrementally computed, load impedance.

This is not necessarily true for the *input reflection coefficient* because the input circuitry can be non-linear and it can also contain input offset voltages and currents. However, because matching networks are usually designed only for the RF fundamental, defining the reflection coefficient as the ratio of fundamental Fourier components of the large signals is often justified.

Creating and Setting Up the Modified Circuit

This example tells you how to

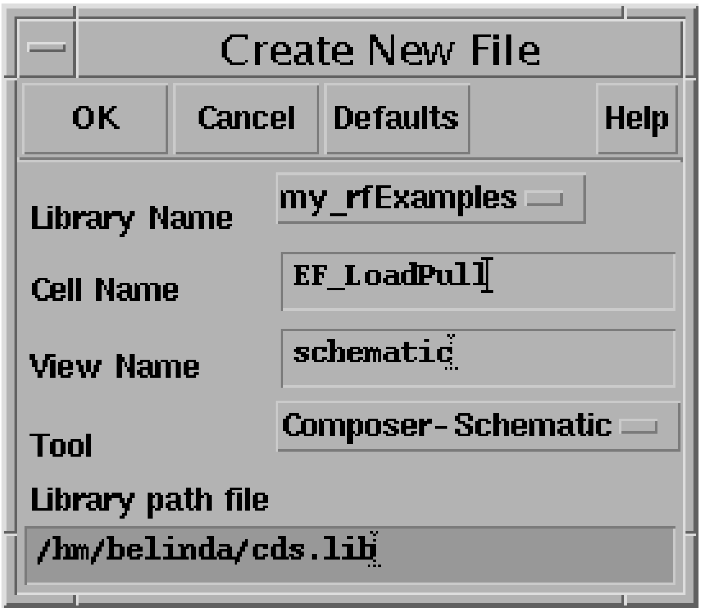
- Create *EF_LoadPull,* a modified copy of the *EF_example* schematic, for this example.
- Set up and run the necessary PSS and Parametric analyses
- Measure load-pull contours for the modified *EF_LoadPull* schematic
- Measure load reflection coefficients for the modified *EF_LoadPull* schematic

Before you start, make sure you have performed the setup procedures for the writable *rfExamples* library, as described in Chapter 3.

Creating a New Empty Schematic Window

1. In the CIW choose *File—New—Cellview*.

The Create New File form appears.



2. In the Create New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select *my_rfExamples*, the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* field, enter *EF_LoadPull*.

   c. In the *View Name* field, enter *schematic*.

   d. In the *Tool* cyclic field, select *Composer-Schematic*.

   e. Click *OK*.

   A new, empty Schematic window named *EF_LoadPull* opens.

Opening and Copying the EF_example Circuit

Now open another schematic window containing the *EF_example* circuit. Then copy the *EF_example* circuit into this empty *EF_LoadPull* Schematic window.

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

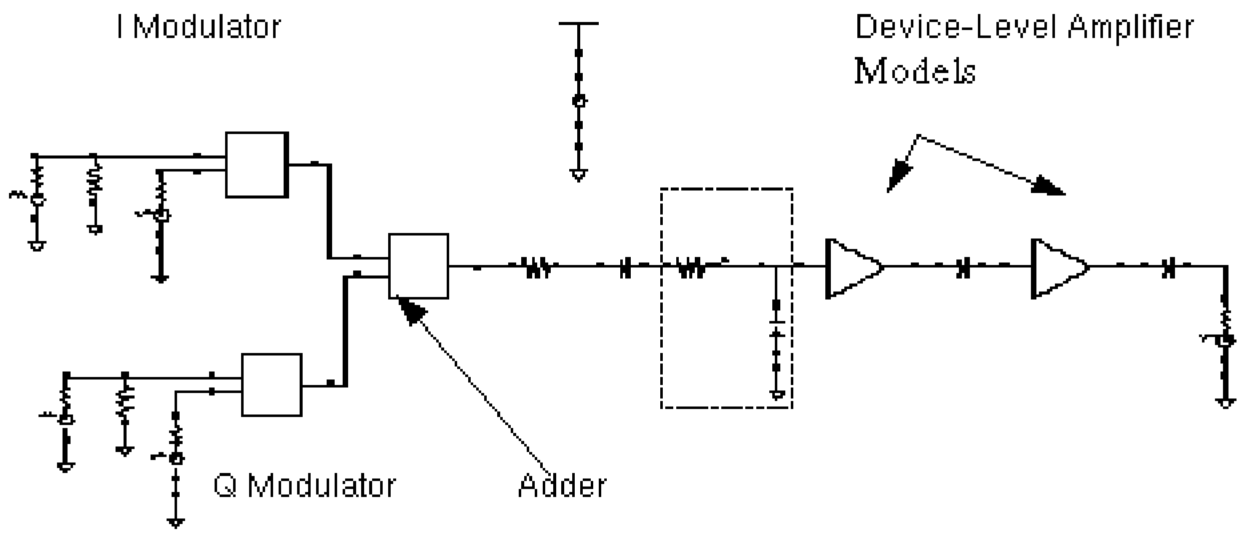
   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

   The Schematic window appears with the *EF_example* schematic. This is a simple direct-conversion transmitter with ideal I/Q modulators.



3. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts at the bottom of the Schematic window.

   a. Following the prompt,

      > *point at object to copy*

      left click and drag to create a box around the entire *EF_example* circuit.

      The *EF_example* components are highlighted in yellow.

   b. Following the prompt,

> *point at reference point for copy*

click inside the outlined elements.

c. Following the prompt,

> *point at destination point for copy*

move the cursor to the empty *EF_LoadPull* Schematic window and click there. This drags a copy of the *EF_example* circuit into the empty Schematic window.

The *EF_LoadPull* Schematic window now contains a copy of the *EF_example* schematic.

d. If necessary, choose *Window - Fit* to center the *EF_LoadPull* circuit in the Schematic window.

4. In the *EF_example* Schematic window, choose *Window - Close*.

The *EF_example* Schematic window closes.

Opening the Simulation Window for the EF_LoadPull Schematic

➤ Open the Simulation window from the *EF_LoadPull* Schematic window as described for the *EF_example* schematic in "Opening the Simulation Window" on page 685.

Setting up the Model Libraries for the EF_LoadPull Schematic

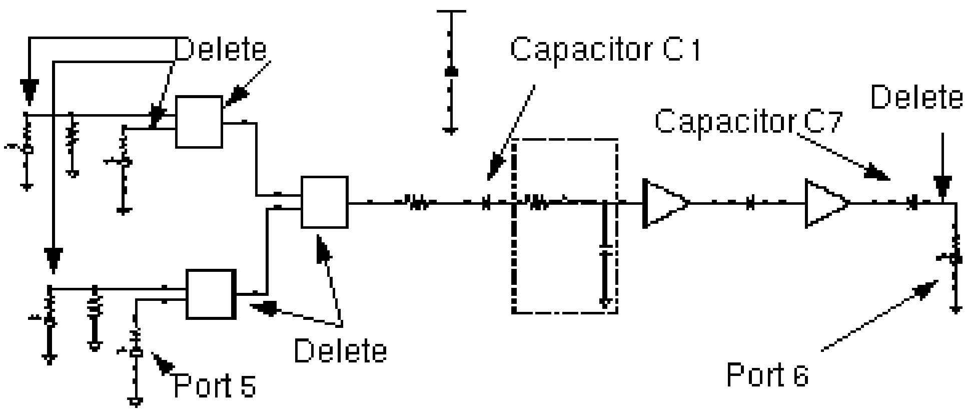
➤ Set up the Model Libraries for the *EF_LoadPull* Schematic as described for the *EF_example* schematic in "Setting Up the Model Libraries" on page 686.

Editing the EF_LoadPull Schematic

In the *EF_LoadPull* Schematic window, delete components and their associated connecting wires as shown in Figure 10-7.

Note: The final *EF_LoadPull* Schematic should look like Figure 10-8 on page 760.

Figure 10-7 Components and Wires to Delete for the EF_LoadPull Schematic



Note: If you need assistance with methods for editing the schematic, see the Virtuoso® Schematic Composer™ User Guide.

Delete Components and Wires from the EF_LoadPull Schematic

1. In the *EF_LoadPull* Schematic window, choose *Edit – Delete.*
2. Click on a component or wire to delete it.
3. Press the *Esc* key when you are done deleting.
4. In the *EF_LoadPull* Schematic window, choose *Edit – Move* to move both *PORT5* and *PORT6* as shown in Figure 10-8 on page 760.
   - ❑ Move *PORT5* closer to capacitor *C1*.
   - ❑ Move *PORT6* away from capacitor *C7* to make room for the *PortAdaptor* component you will add here.

   a. Left click and drag to create a rectangle around *PORT5* and its *GND*.

      PORT5 and the *Gnd* are highlighted in yellow.

   b. Click inside the yellow line and drag the outlined components to move them close to capacitor *C1* as shown in Figure 10-8 on page 760.

   c. Left click and drag to create a rectangle around *PORT6* and its *GND*.

      PORT6 and the *Gnd* are highlighted in yellow.

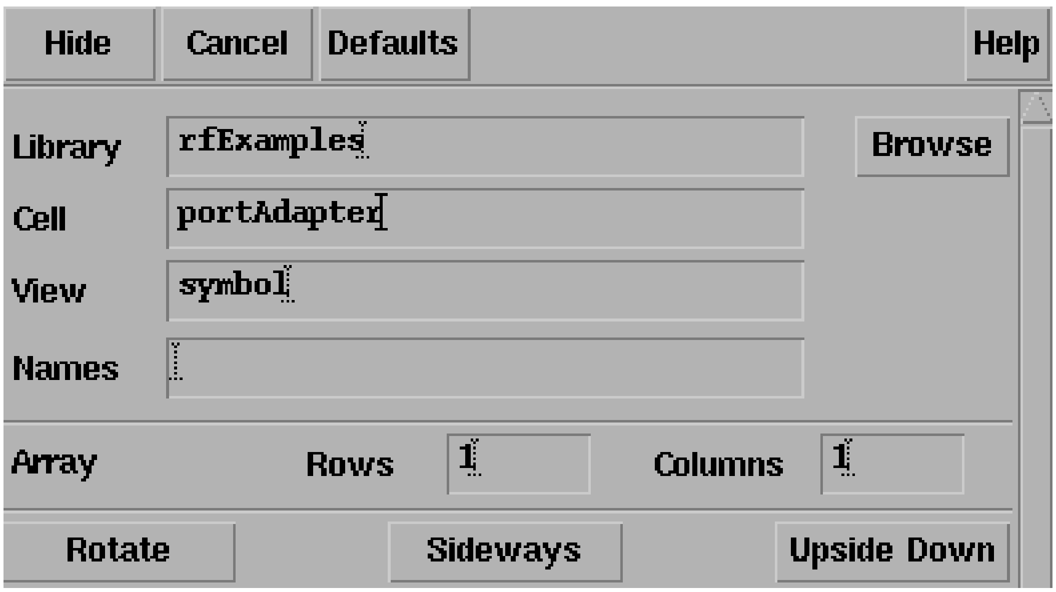
d. Click inside the yellow line and drag the components to move them away from capacitor *C7* as shown in Figure 10-8 on page 760. This makes room for the *PortAdapter*.

e. Choose *Window - Fit* to center the edited schematic in the window.

Place the PortAdaptor

1. In the *EF_LoadPull* Schematic window, choose *Add – Instance*.

   The Add Instance form appears.



2. In the Add Instance form.

   a. In the *Library* field, type `rfExamples`.

   b. In the *Cell* field, type `portAdapter`.

   c. In the *View* field, type `Symbol`.

   As you move your cursor from the form to the *EF_LoadPull* Schematic window, a copy of the *portAdapter* moves with the cursor. Left click to place the *portAdapter* as shown in Figure 10-8 on page 760.

   After you place the *portAdapter* in the schematic, press the *Esc* key to remove the *portAdapter* symbol from your cursor and close the Add Instance form.

Wire the Schematic

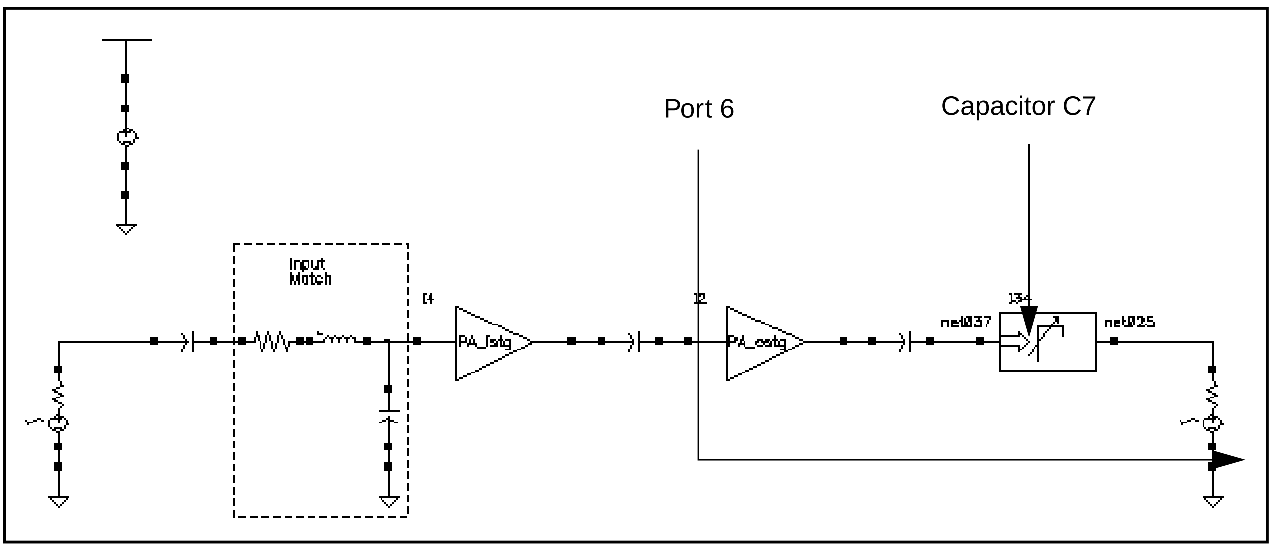
In the *EF_LoadPull* Schematic window,

- Connect *PORT5* to Capacitor *C1*.
- Connect the *portAdapter* between Capacitor *C7* and *PORT6*.

1. In the Schematic window, choose *Add - Wire (Narrow)* and do the following.
   a. Click the terminal on *PORT5* then click the terminal on *C1*.
   b. Click the terminal on *C7* then click the terminal on the *PortAdaptor.*
   c. Click the terminal on the *PortAdaptor* then click the terminal on *PORT6.*
2. Press the Esc key to stop wiring.

   The edited schematic looks like the one in Figure 10-8.

Figure 10-8 The EF_LoadPull Schematic



Edit CDF Properties for both the PortAdaptor and Port6

In the *EF_LoadPull* Schematic, the *portAdapter* and its terminating port, *Port6*, must have the same reference resistance.

1. In the *EF_LoadPull* Schematic window, select the *PortAdapter*.
2. In the *EF_LoadPull* Schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears with information for the *Port Adapter* displayed.

3. In the Edit Object Properties form, do the following and click *Apply*.

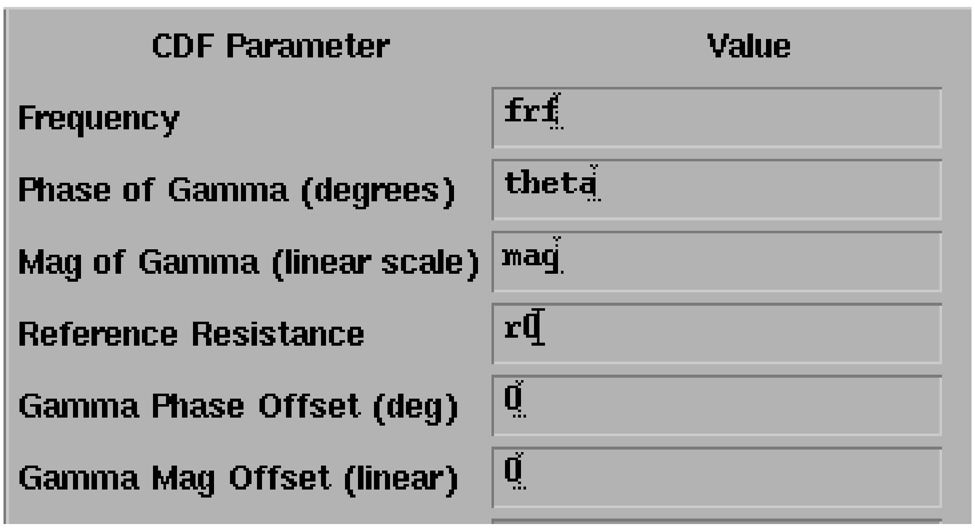
   a. Type *frf* for *Frequency*.

   b. Type *theta* for *Phase of Gamma (degrees)*.

   c. Type *mag* for *Mag of Gamma (linear scale)*.

   d. Type *r0* for *Reference Resistance*.

   The completed form looks like this.



CDF Parameter	Value
Frequency	frf
Phase of Gamma (degrees)	theta
Mag of Gamma (linear scale)	mag
Reference Resistance	r0
Gamma Phase Offset (deg)	0
Gamma Mag Offset (linear)	0

4. In the *EF_LoadPull* Schematic window, select *Port6*.

   The Edit Object Properties form changes to display data for *Port6*.

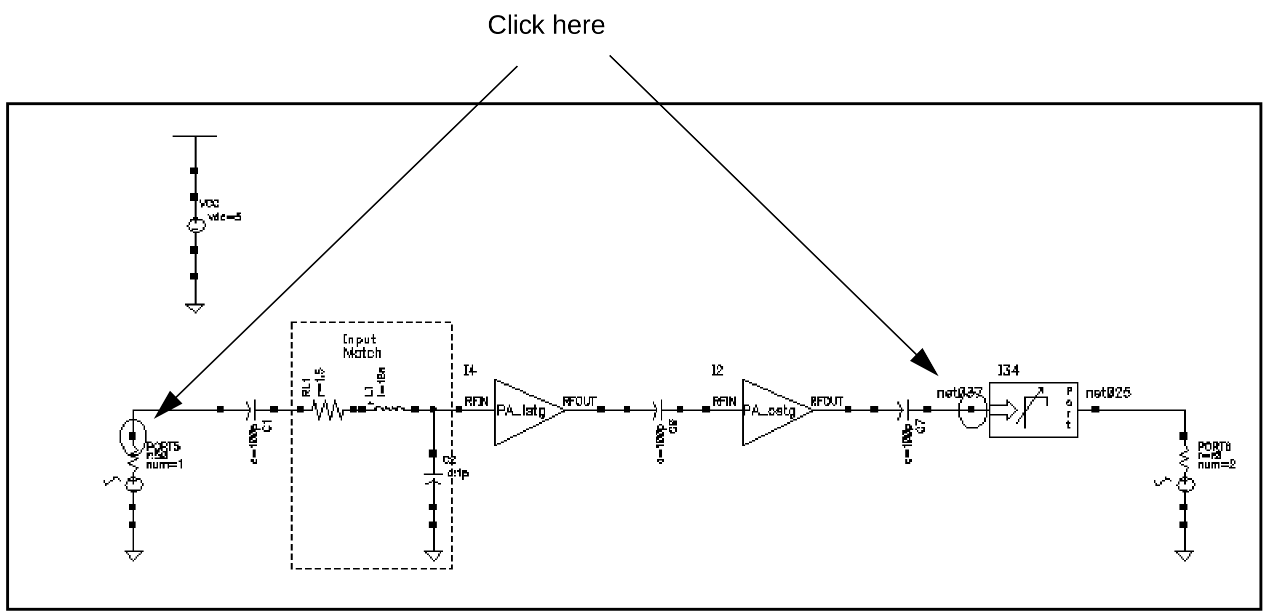
5. In the Edit Object Properties form for *Port6*, type *r0* for *Resistance* and click *OK*.

Select Outputs To Save

1. In the Simulation window for the *EF_LoadPull* Schematic, choose *Outputs – To Be Saved – Select on Schematic*.

2. In the *EF_LoadPull* Schematic window, click on each terminals that is circled in Figure 10-9.

Figure 10-9 Outputs to Save in the EF_LoadPull Schematic



After you click on a terminal, it is circled in the schematic as shown in Figure 10-9. The selected outputs are also displayed in the Simulation window Outputs area as shown in Figure 10-10.

Figure 10-10 Outputs Area in the Simulation Window

Outputs

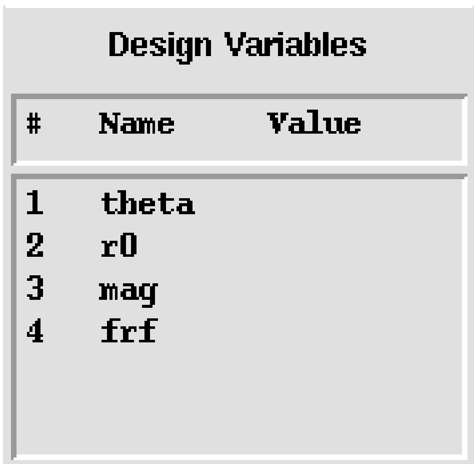
#	Name/Signal/Expr	Value	Plot	Save	March
1	PORT3/PLUS		no	yes	no
2	I3/out		no	yes	no

Editing Variables

1. In the Simulation window, choose *Variables – Copy from Cellview*.

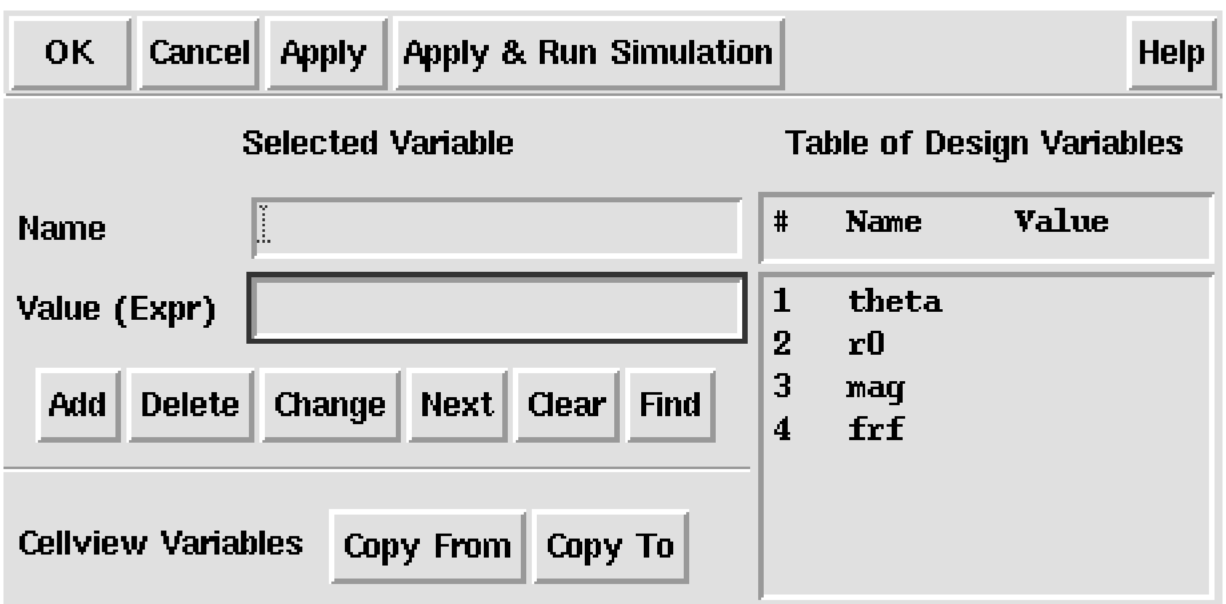
The *Design Variables* list box in the Simulation window changes to reflect the copied variables from the schematic as shown in Figure 10-11.

Figure 10-11 Design Variables Area Showing Variable Names



2. In the Simulation window, choose *Variables - Edit*.

   The Editing Design variables form appears.



3. In the Editing Design Variables form, associate the variable values with the variable names as listed in Table 10-4.

Table 10-4 Design Variable Values

Variable	Value
theta	`0`
r0	`50`
mag	`0`
frf	`1G`

In the Editing Design Variables form, do the following then click *OK*.

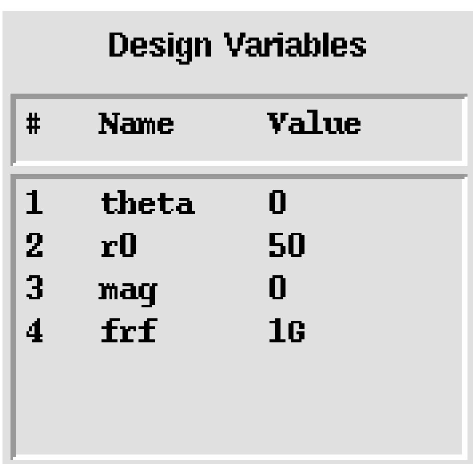
a. Highlight one of the variables In the *Table of Design Variables* list box.

The variable name displays in the *Name* field.

b. Type it's associated value from Table 10-4 into the *Value (Expr)* field.

c. Click *Change*.

The Design Variables list box in the Simulation window now displays both variable names and the value associated with each name.



Save the Changes to the EF_LoadPull Schematic

➤ In the Schematic window, choose *Design – Check and Save* to save the current state of the *EF_LoadPull* schematic.

Setting Up and Running the PSS and Parametric Analyses

This example tells you how to

- Set up the swept PSS analysis to sweep the design variable `theta`.
- Set up the Parametric analysis to sweep the design variable `mag`.
- Run the parametric and swept PSS analyses.
- Plot the results.

Performing the PSS Simulation

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

   The form changes to display options for PSS simulation.

3. In the Choosing Analyses form, do the following and then click *OK*.

   a. If information for *frf* is not displayed in the *Fundamental Tones* list box, type `frf` in the *Name* field below the list box and click *Clear/Add*.

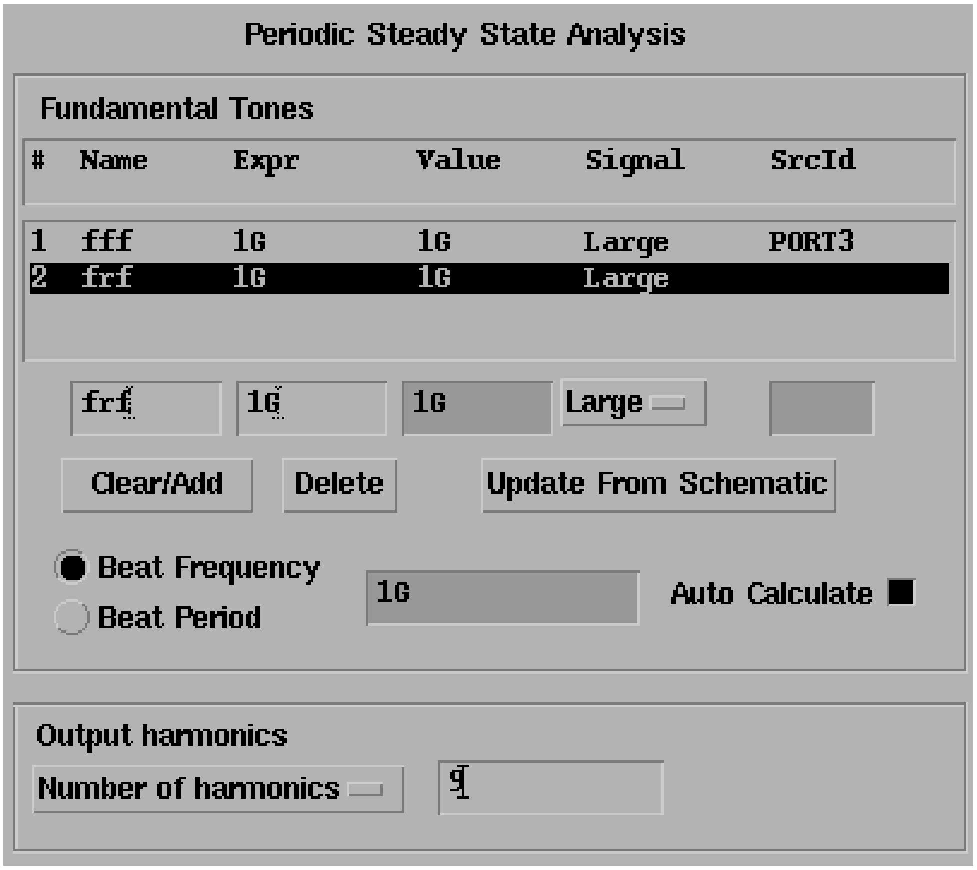
   b. Click *Update From Schematic* to update the values in the *Fundamental Tones* list box from those in the schematic.

      Note: Before you can *Update From Schematic*, you have must performed a *Design - Check and Save* on the design in the Schematic window.

   c. Click *Auto Calculate* to automatically calculate and enter a value in the *Beat Frequency* field.

   d. Choose *Number of harmonics* for *Output harmonics* and type 9 in the adjacent field.

The top of the PSS analysis form looks like this.



e. Highlight *moderate* for *Accuracy Defaults (errpreset).*

f. Highlight the *Sweep* button and choose *Variable* in the associated cyclic field.

g. Type `theta` for *Variable Name* (or click on *Select Design Variable*, highlight *theta* in the form that appears and click *OK*).

h. Choose *Start – Stop* for *Sweep Range*.

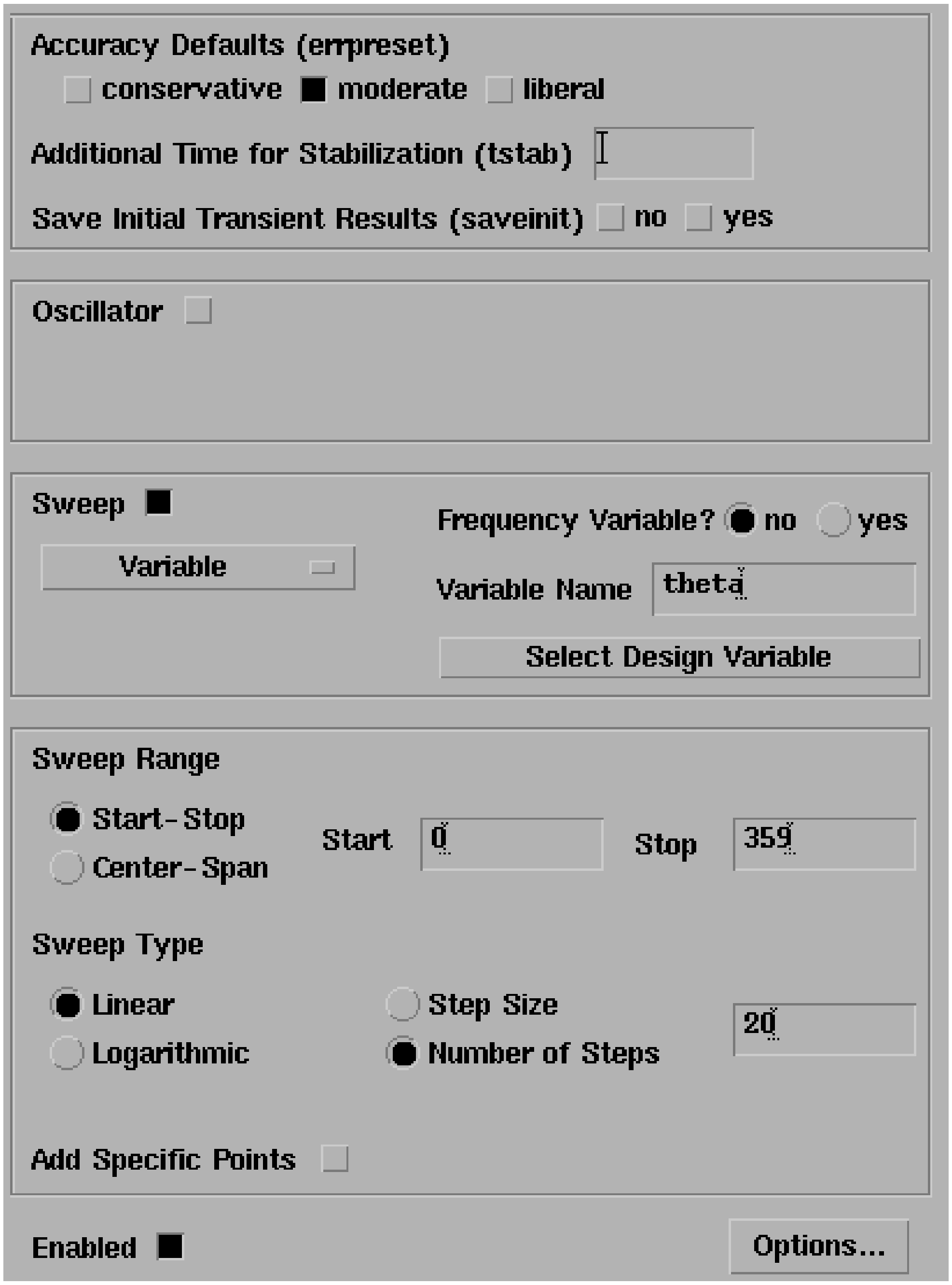
i. Type `0` for the *Start* value and `359` for the *Stop* value.

j. Choose *Linear* for *Sweep Type*.

k. Highlight *Number of Steps* and type `20` in the adjacent field.

l. Verify that *Enabled* is highlighted.

The bottom of the PSS analysis form looks like this.



Performing the Parametric Analysis

1. In the Simulation window, choose *Tools – Parametric Analysis*.

The Parametric Analysis form appears.

2. In the Parametric Analysis form, do the following:

   a. Type *mag* for *Variable Name*.

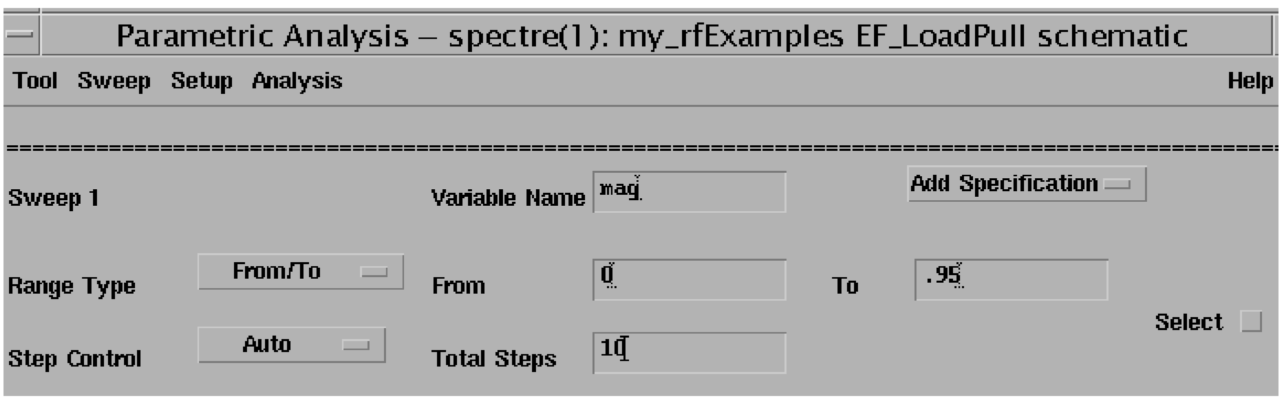
   b. Choose *From/To* in the *Range Type* cyclic field.

   c. Type `0` for *From* and `.95` for *To* in the adjacent fields.

   d. Choose *Linear* in the *Step Control* cyclic field.

   e. Type `10` in the adjacent *Total Steps* field.

   The completed form looks like this.



3. In the Parametric Analysis form, choose *Analysis – Start*.

   Look in the CIW for a message that says the simulation has completed successfully.

Displaying Load Contours

1. In the Simulation window, choose *Results – Direct Plot – Main Form*.

   The Direct Plot form appears.

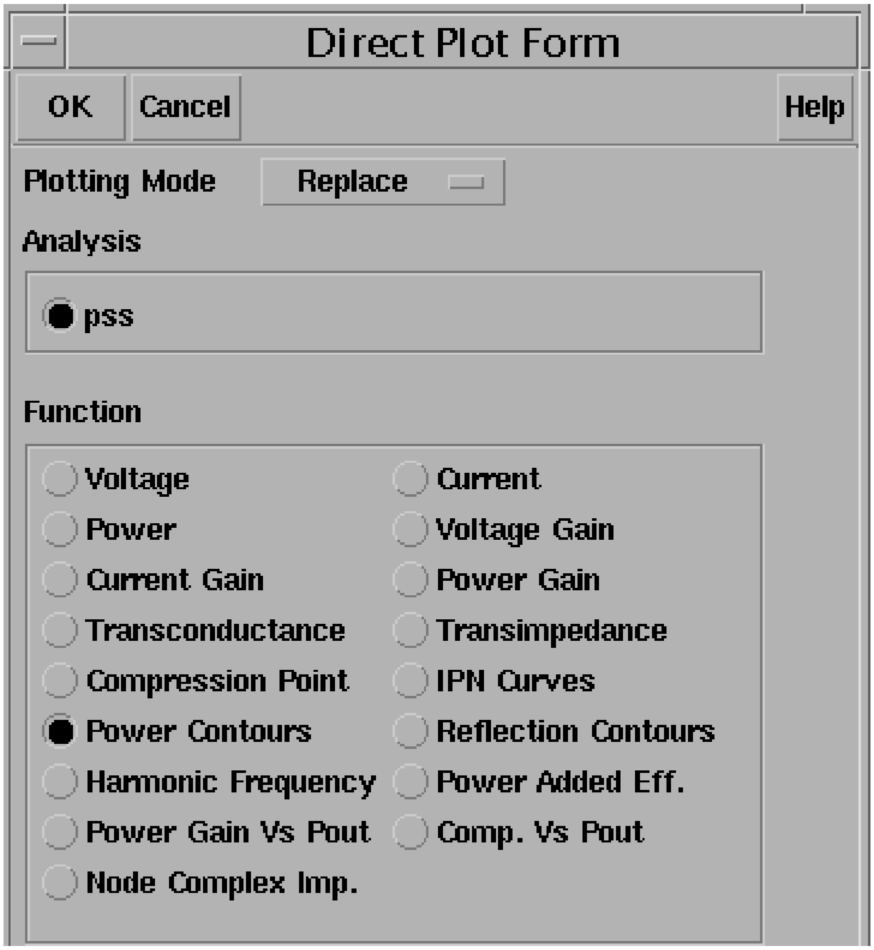
2. In the Direct Plot form, do the following:

   a. Select *Replace* in the *Plot Mode* cyclic field.

   b. Highlight *pss* for *Analysis*.

   c. Highlight *Power Contours* for *Function*.

The top of the Direct Plot form looks like the following.



d. Highlight *Magnitude* for *Power Modifier*.

e. Leave *Maximum Power* and *Minimum Power* blank.

f. Type `9` for *Number of Contours*, if necessary.

g. Type `50.0` for *Reference Resistance*, if necessary.

h. Following the prompt at the bottom of the form,

> *Select Output Harmonic on this form...*,

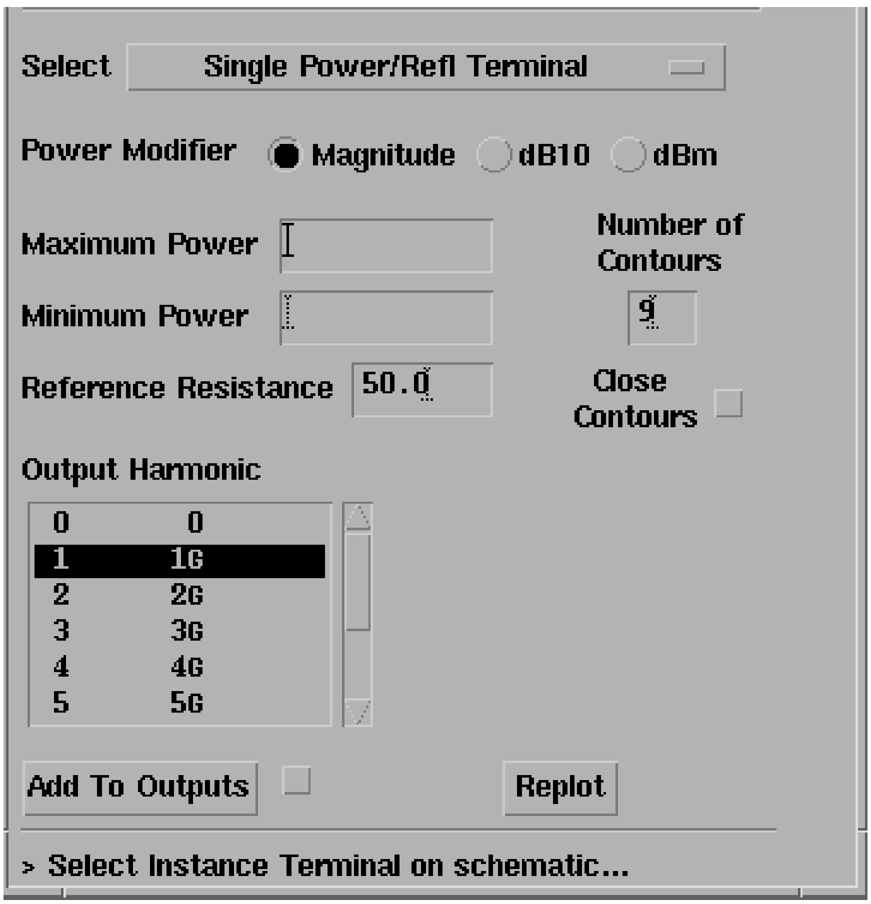
Highlight *1   1G* for *Output Harmonic*.

You would select a different harmonic if the PSS fundamental frequency were smaller than `1 GHz`. For example, if another part of the circuit is driven at `1.5 Hz`, the fundamental will be `500 MHz`. But because the part of the circuit where you want to do load-pull analysis operates at `1 GHz`, in order to plot the correct contours, the contours associated with `1 GHz`, you must specify the second harmonic of the fundamental, `2*500 MHz = 1 GHz`.

i. The *Select* cyclic field displays *Single Power/Refl Terminal* and the prompt at the bottom of the form displays

> *Select Instance Terminal on Schematic*.

The bottom of the Direct Plot form looks like the following.

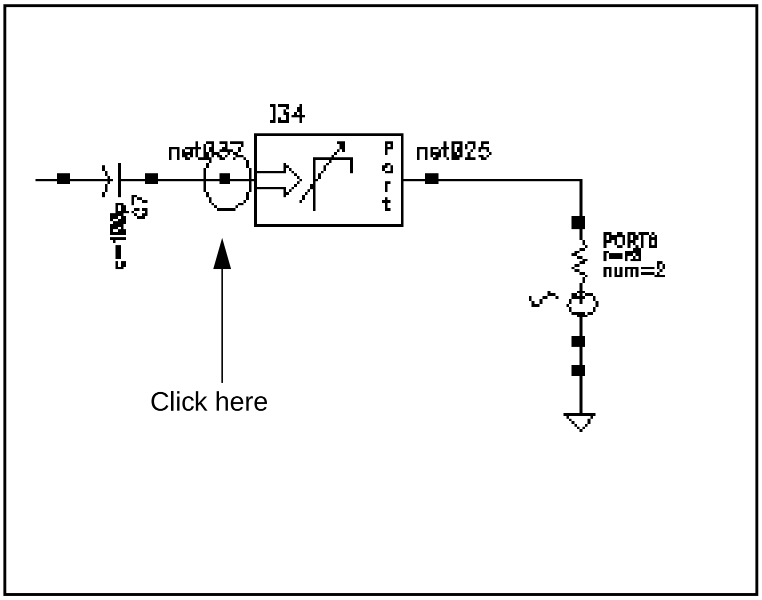


3. Follow the prompt at the bottom of the Direct Plot form,

> *Select Instance Terminal on Schematic*

In the Schematic window, click on the terminal circled in Figure 10-12.

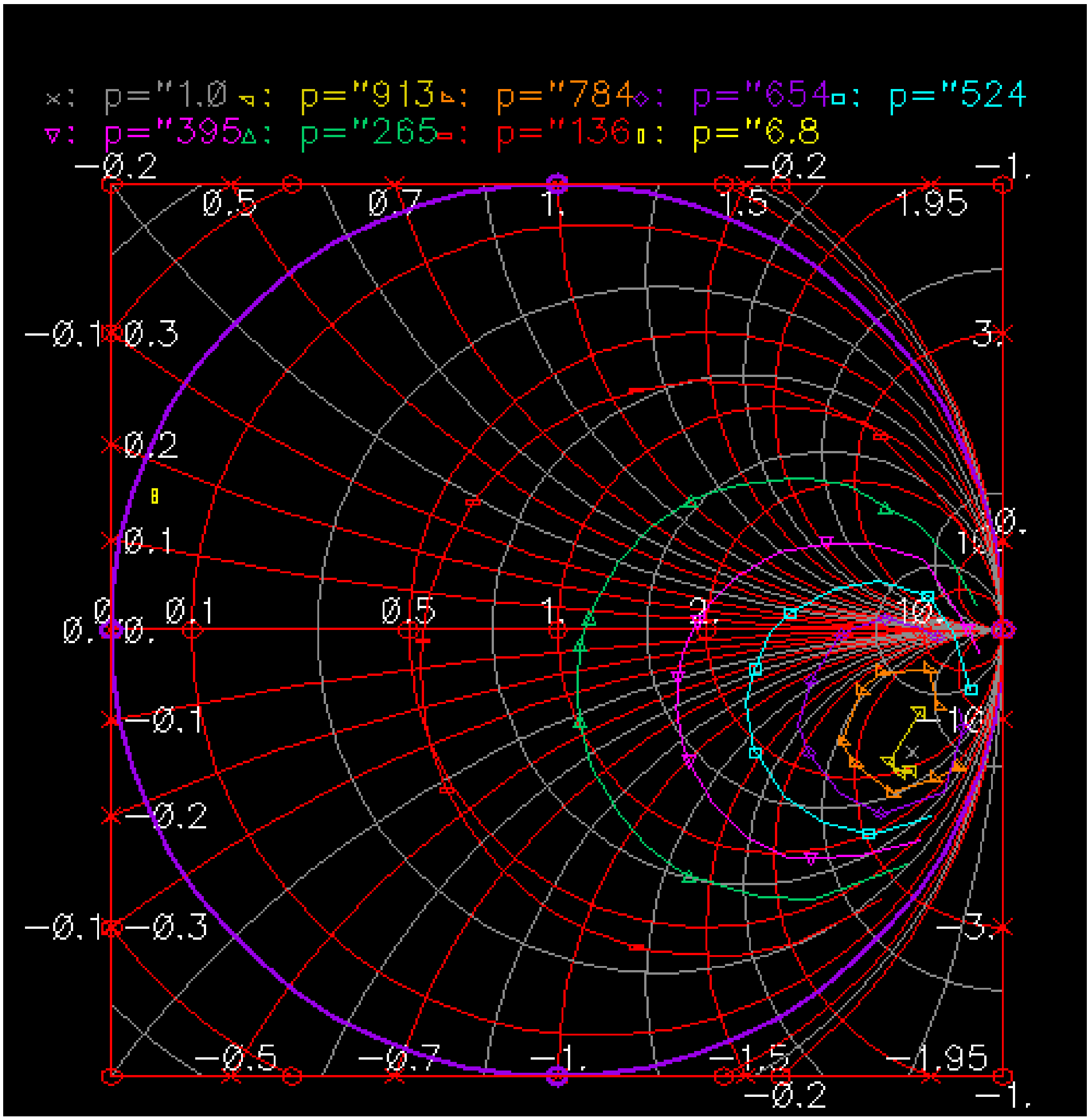
Figure 10-12 The Terminal at the Port Adapter



Note: Subsequent instructions call this terminal the *terminal at the port adapter*.

4. After you select the terminal, the plot for the load contours appears in the Waveform window. Figure 10-13 shows the entire plot.

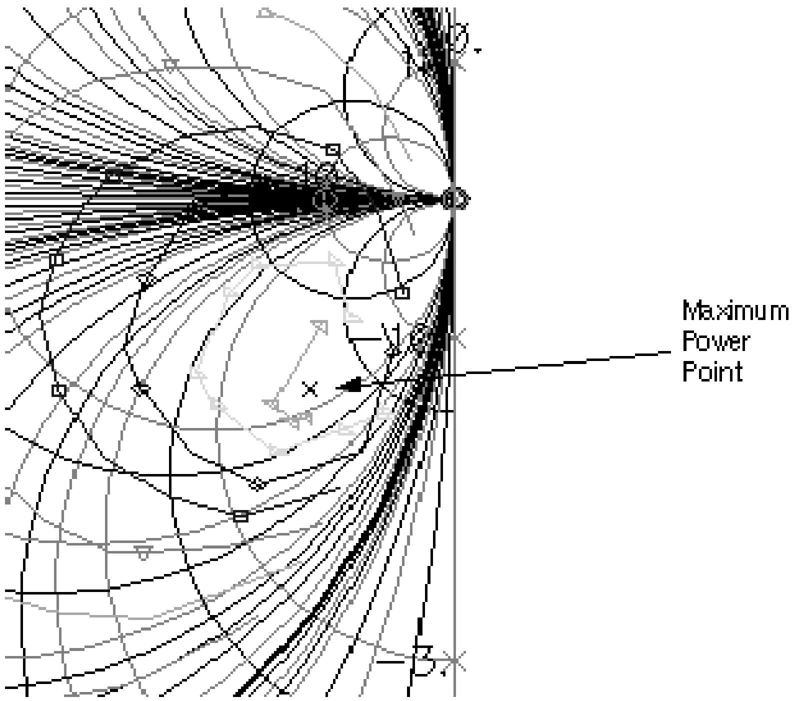
Figure 10-13 Complete Load Contour Plot



A small X appears at the maximum power point, which in this case lies near the center of the smallest constant power contour.

Figure 10-14, an enlarged area taken from Figure 10-13, more clearly shows a X that marks the maximum power point.

Figure 10-14 Enlargement Showing Maximum Power Point



If you place the cursor on the X, you can read the following information across the top of the Waveform window.

```
Real: 2.479   Imag: -4.739   Freq: -360   p=”1.047m”;Constant Power Contours
```

This indicates that a normalized load impedance of about 2.48-j4.738 dissipates the most power. The X appears at the maximum power point, which in this case lies near the center of the smallest constant power contour.

Adding the Reflection Contours to the Plot

You might want to maximize load power subject to a constraint on the magnitude of the amplifier’s input reflection coefficient. Such a constraint can prevent unstable interactions with the preceding stage.

You can overlay load-pull contours with contours of constant input reflection coefficient magnitude. The optimal load corresponds to the reflection coefficient that lies on the *largest power* load-pull contour and also lies on a constant input reflection coefficient contour that is within the constraint. Here, *largest power* means the contour corresponding to the largest amount of power delivered to the load.

To use the topographical map analogy again, this is like overlaying constant elevation contours with constant temperature contours. The optimum objective would be like trying to find the highest point on the mountain such that the temperature is above 60 degrees.

1. In the Direct Plot form, do the following:

   a. Select *Append* in the *Plot Mode* cyclic field.

   b. Highlight *pss* for *Analysis*.

   c. Highlight *Reflection Contours* for *Function*.

   d. In the *Select* cyclic field, select *Separate Refl and RefRefl Terminals*. The prompt at the bottom of the form changes to

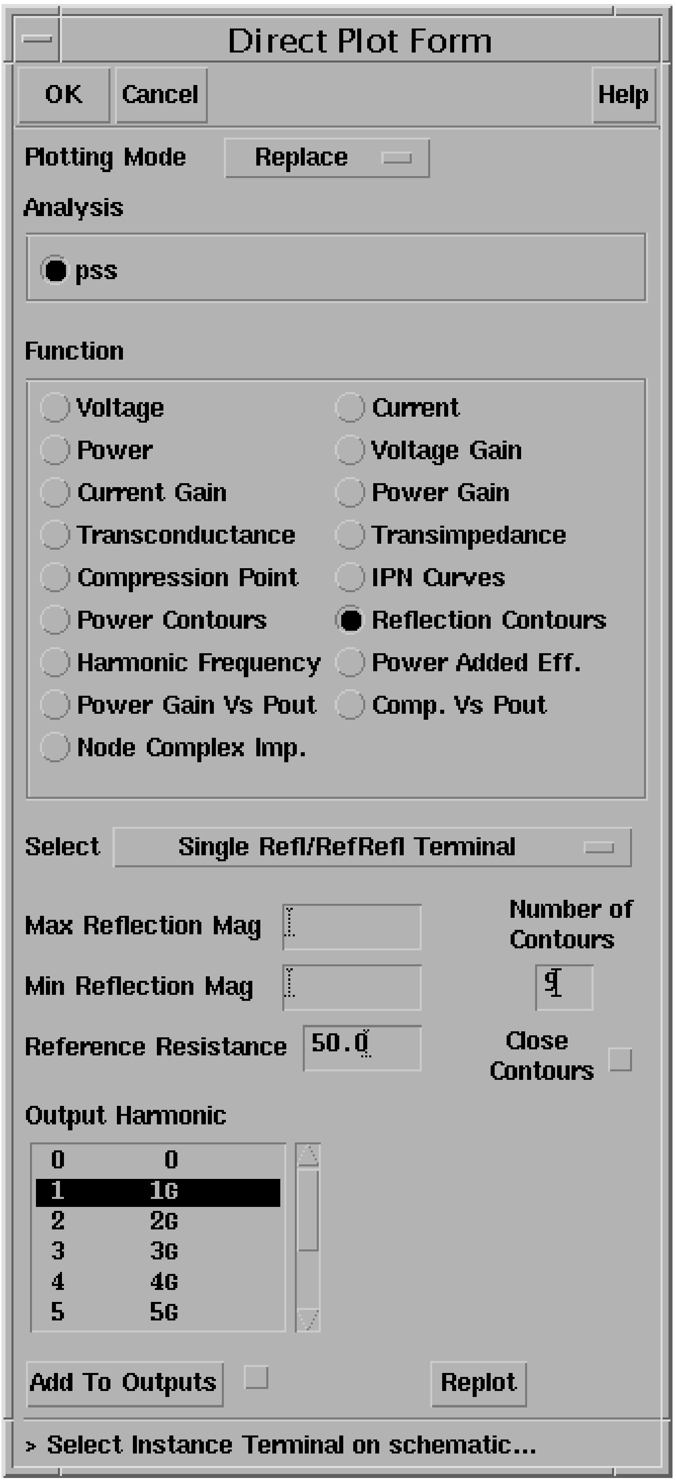
      > *Select Reflection Instance Terminal on Schematic*

   e. Type `9` for *Number of Contours*, if necessary.

   f. Type `50.0` for *Reference Resistance*, if necessary.

   g. Highlight *1   1G* for *Output Harmonic*.

The completed form looks like this.

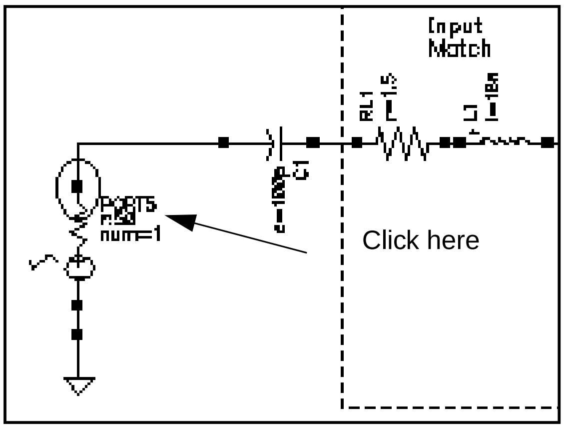


2. Follow the prompt at the bottom of the Direct Plot form,

> Select Reflection Instance Terminal on Schematic

In the Schematic window, click on the terminal circled in Figure 10-15.

Figure 10-15 The Terminal at Port 5



Note: Subsequent instructions call this terminal the *terminal at Port 5*.

The prompt on the bottom of the Direct Plot form changes to

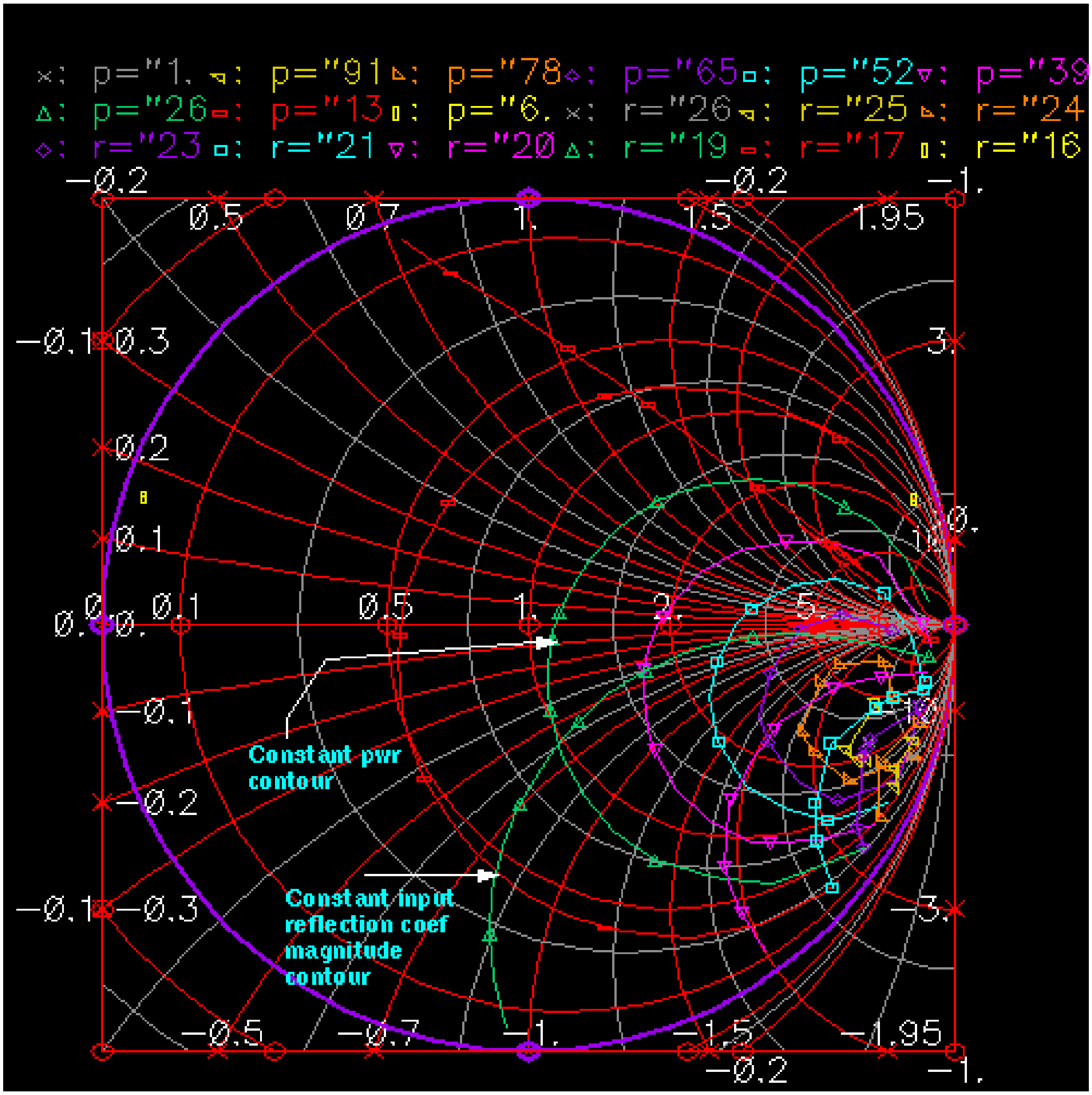
> Select Reflection Instance Terminal on Schematic

3. Follow the new prompt at the bottom of the Direct Plot form,

> Select Instance Terminal on Schematic....

In the Schematic window, click on the *terminal at the port adapter* (See Figure 10-12 on page 771).

The plot for reflection contours is added to the Waveform window.



Let's assume that for stability, the input reflection coefficient should be less than or equal to 0.218. For the curves shown above, the optimal normalized load impedance is approximately 3.908 -j5.073. Assuming you were free to draw smaller constant power contours, the true constrained optimal power point would occur when the constant power contour is just tangent to the constant input reflection coefficient contour of 0.218.

In the Direct Plot Form for *Reflection Contours*, *Separate Refl & RefRefl Terminals* is selected to measure *Reflection Contours* as contours of constant reflection (magnitude) as seen from the power amplifier's input (measured at *port 5,* the input port) with respect to the reflection of the power amplifier's load (measured at the input to the port adapter.

In this case, in response to the first prompt

> *Select Reflection Instance Terminal on Schematic*

select the input to the power amplifier, the terminal of *port 5* in the schematic.

In response to the second prompt

> *Select Instance Terminal on Schematic*

select the output of the power amplifier, the input terminal to the *port adapter* in the schematic.

So the *port adapter's* reflection (both `mag` and `phase`) is swept, which is the same as sweeping the power amplifier's load. In the Smith chart you see input reflection plots as seen by *port 5*.

The reflection contour plot shows the effect of the power amplifier's load on the input reflection while the power contour plot shows the effect of the power amplifiers's load on the power gain.

For reflection contours, the following terminal selections apply.

1. For *Single Refl/RefRefl Terminal* select one terminal.

   One reflection coefficient of that terminal is computed as

   `gamma1` = (*Z* - *Z0*) / (*Z*+*Z0*)

   where *Z* is the large signal impedance at the fundamental. The resulting Smith chart plots contours where `gamma1` is constant.

2. For *Single Refl/RefRefl Term and ref Term* select two terminals.

   This is the differential case for number 1. The different voltages of the two nets are used in the large signal impedance calculation.

3. For *Separate Refl and RefRefl Terminals* select two terminals

   Two reflection coefficients for each terminal are computed as

   `gamma1` = (*Z1* - *Z0*) / (*Z1*+ *Z0*)

   `gamma2` = (*Z2* - *Z0*) / (*Z2*+ *Z0*)

where *Z1* and *Z2* are the large signal impedance for each terminal at the fundamental. The resulting Smith chart plots gamma2 and contours where `gamma1` is constant.

4. For *+ - Refl and + - Ref Refl Terminals* select 4 terminals

   This is the differential case for number 3.

For power contours, the following terminal selections apply.

1. For *Single Power/Ref Terminals* select one terminal.

   The power (*P1*) and the reflection coefficient (`gamma1`) of that terminal are computed. The resulting Smith chart plots `gamma1`and contours where `P1` is constant.

2. For *Single Power/Refl Term and ref Term*

   This is the differential case for number 1.

3. For *Separate Power and Refl Terminals* select 2 terminals

   The power of the first terminal (*P1*) and the reflection coefficients of the second terminal (`gamma2`) are computed. The resulting Smith chart plots `gamma2` and contours where *P1* is constant.

4. For *+ - Power and + - Refl Terminals*

   This is the differential case for number 3.

Are Constant Power Contours the Same as Constant Gain Contours?

In general, *constant power contours* are not the same as *constant power gain contours*. The simulator does not maintain any input impedance match while it sweeps the load to generate load-pull contours. The resulting contours are not constant power gain contours, but are only constant power contours. However, if the input reflection coefficient and input power do not change much over the load sweep range, the generated contours are good approximations of constant gain contours. By plotting constant power contours in the input reflection coefficient plane, you can see how both change when you sweep load reflection coefficient. If the input power does not vary with load, the constant power contours are also constant power gain contours. If the input power does vary with the load, the constant power contours tell you how much you have to change the input matching network during the load sweep to maintain a constant input power.

1. In the Direct Plot form, do the following:

   a. Highlight *Replace* for *Plot Mode*.

   b. Highlight *Power Contours for Function*.

c. Highlight *Single Power/Ref Terminal*.

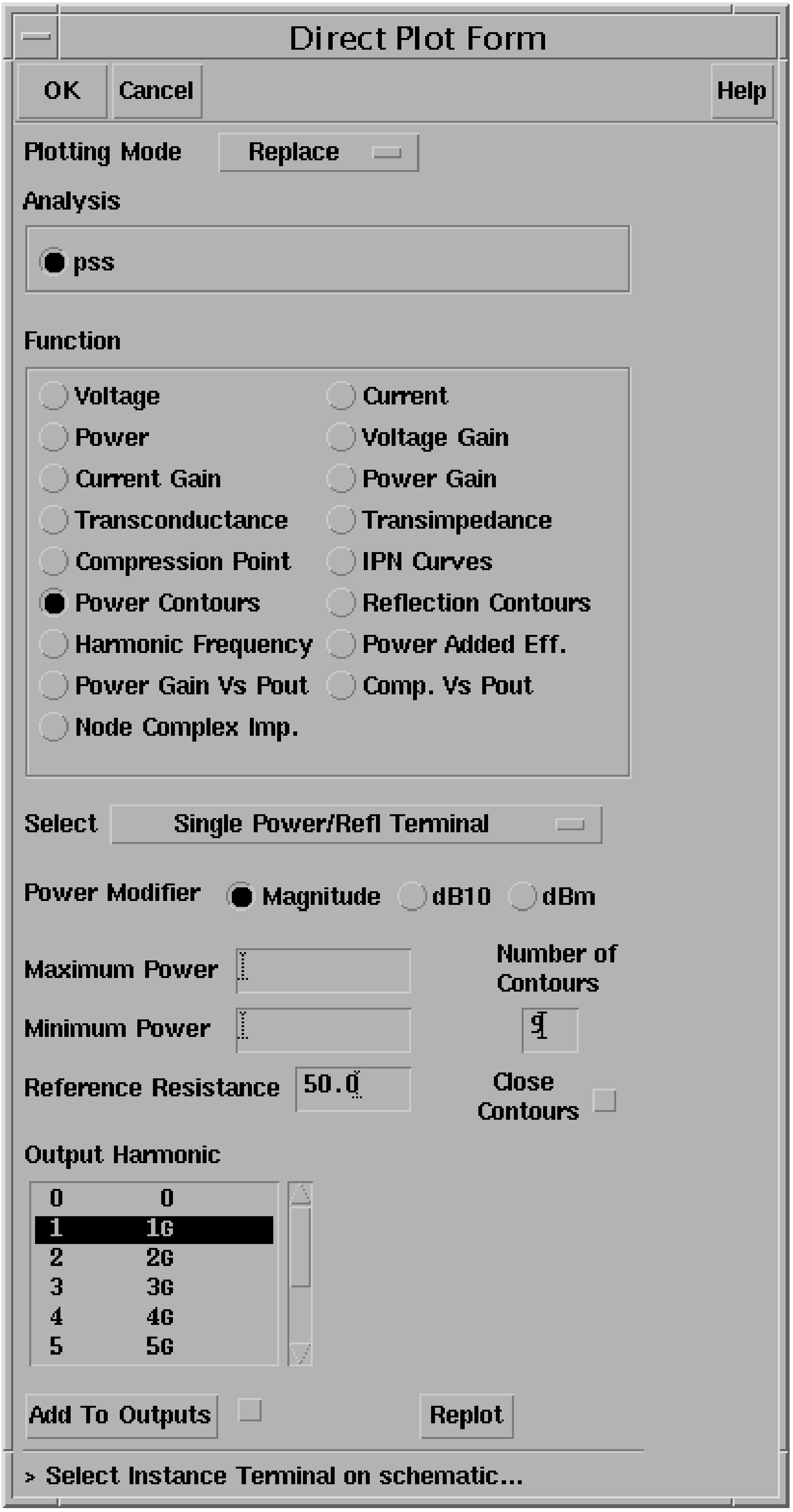
d. Highlight Magnitude for Power Modifier.

e. Type 9 for *Number of Contours*, if necessary.

f. Type `50.0` for *Reference Resistance*, if necessary.

g. Highlight 1 G for *Output Harmonic*.

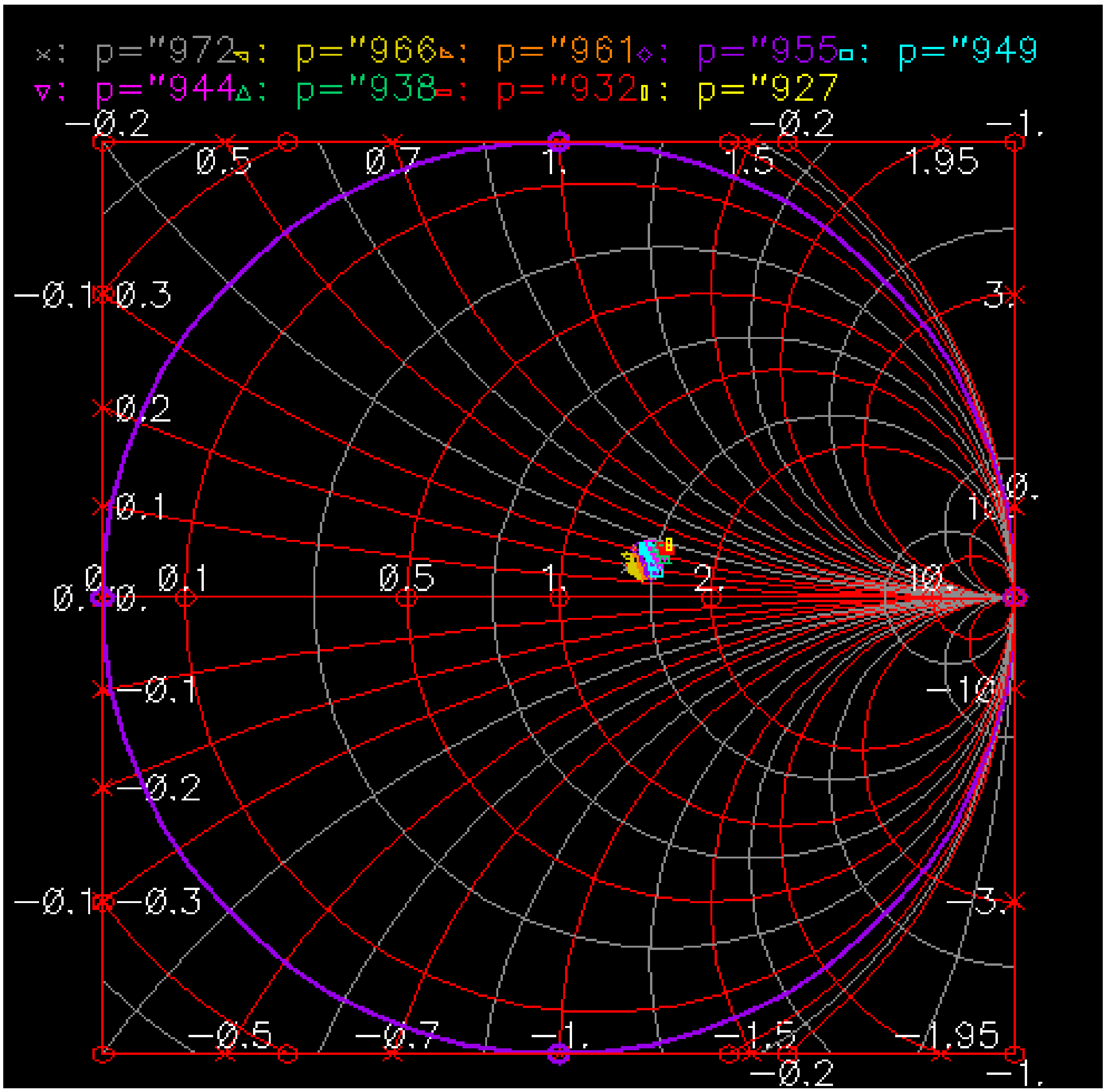
The completed form looks like this.



2. In the Schematic window, click on the terminal at Port 5 (See Figure 10-15 on page 776).

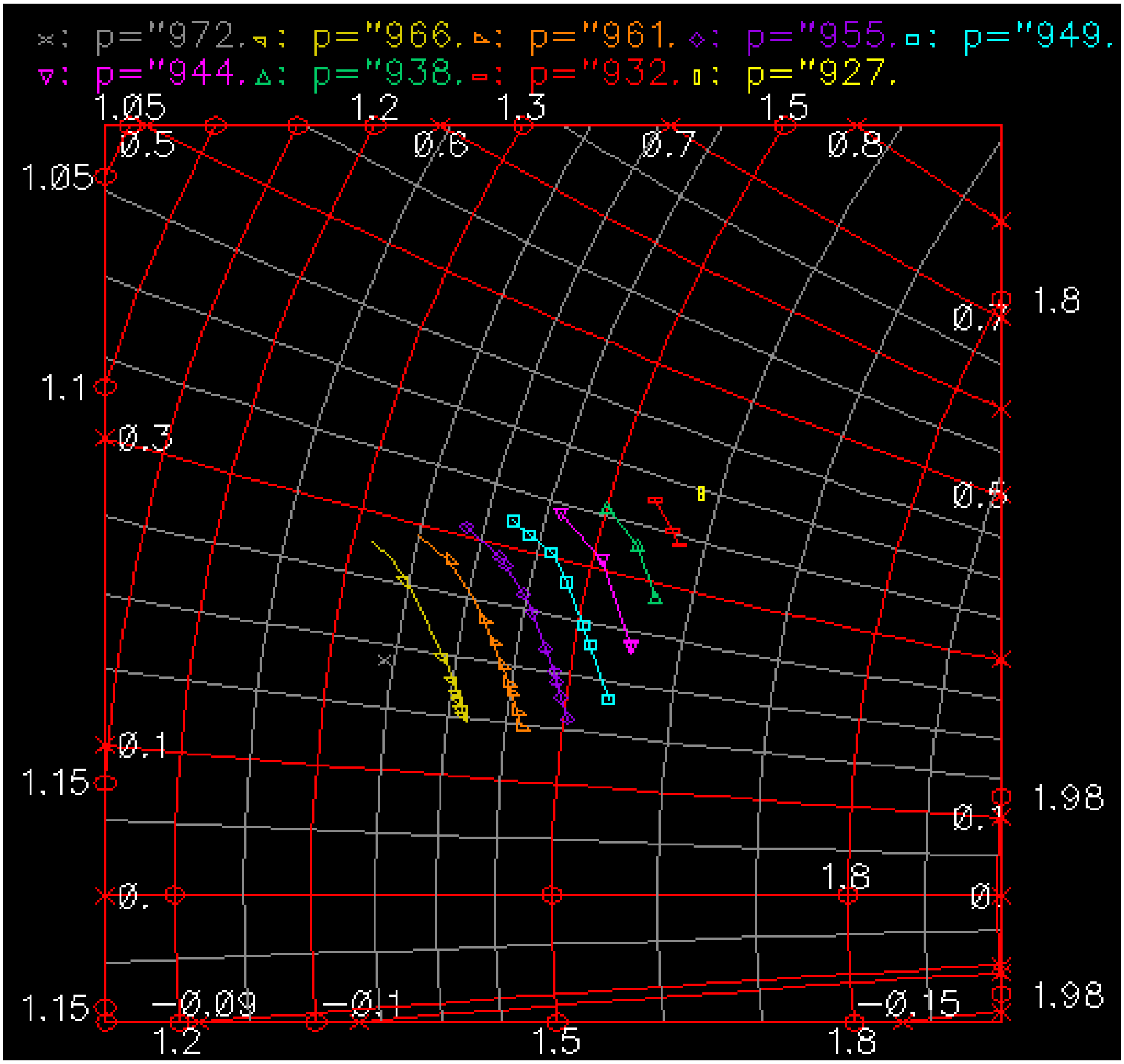
   The plot of input reflection coefficients appears as in Figure 10-16.

Figure 10-16 Smith Chart of Input Reflection Coefficients Generated by Sweeping Load



3. In the Waveform window, choose *Zoom – ZoomIn* and then click and drag with the mouse to form a rectangle that includes the area you want to see.The magnified view is shown in Figure 10-17.

Figure 10-17 Magnified View of the Constant Power Contours



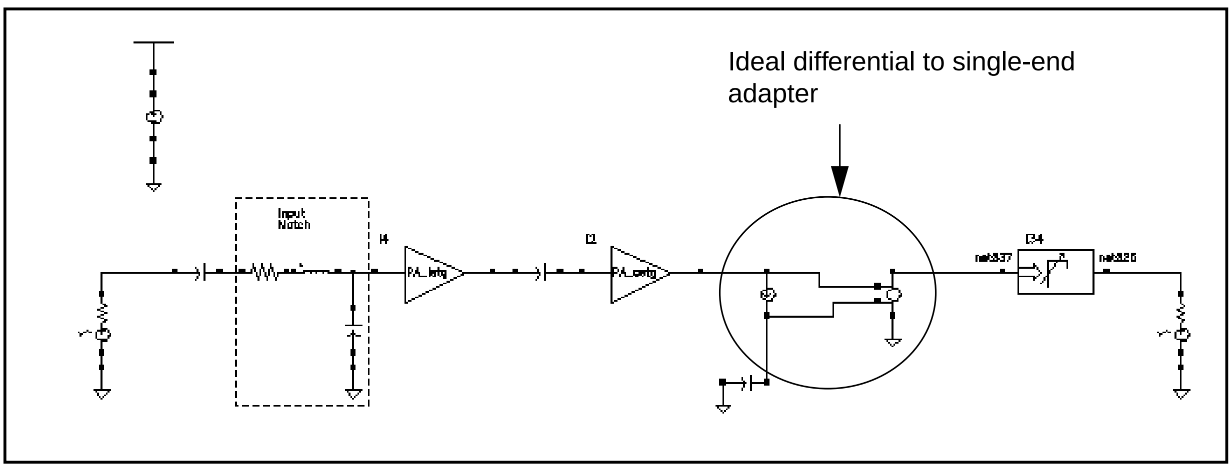
The real part of the input impedance varies from about 1.3 to 1.6 in normalized units and the imaginary part varies from 0.15 to 0.4 in normalized units. The input power varies from 927 nW to 967 nW. You find these numbers by placing the cursor on the end points of the outside contours. You read the impedance and power/reflection for that cursor location at the top of the window.

The numbers vary slightly as the load is swept. The objective here is to determine how much the input reflection coefficient and the input power vary as the load varies. If the input quantities vary significantly, then the generated load pull contours do not coincide with the constant power gain contours because the input power does not remain constant. The amount of variation in the input reflection coefficient tells you by how much you would have to re-tune the input matching network to maintain constant input power as you sweep the load. You have to decide whether this much variation matters.

Moving to Differential Mode

Up until now, the examples used in this chapter have been single-ended. Since many circuits are differential, Figure 10-18 shows a simple way to modify the example circuit to make it differential.

Figure 10-18 Simple Differential Circuit



If your circuit has one or more differential ports, you can translate between single-ended and differential circuitry using linear-dependent controlled sources. The extra circuitry added to Figure 10-18 shows a modified version of the circuit where the output has been modified to make it differential. The transition from single-ended to differential output was made using the controlled sources *F0* and *E1* (circled in Figure 10-18).

The controlled sources, *cccs* and *vcvs*, are available in *analogLib*. The gain of *cccs* is -1 to be consistent with the polarity of the *vcvs*. The *vcvs* (*E1*) is the *Name of voltage source* parameter in the *cccs*.

Using S-Parameter Input Files

In this example, you create an S-parameter data file then you enter it back into the original circuit. The S-parameter data file replaces that portion of the circuit originally used to create the S-parameters. To enter tabulated S-parameters from some other source, record the data in the format shown in the S-parameter data file and select the appropriate numerical options in the nport.

Important

To run a SpectreRF analysis that includes the nport component, make sure you use the rational fitting option.

This example shows you how to

- Set up and simulate the `s.param.first` circuit to create an S-parameter data file (`sparam.practice`) for the `s.param.first` schematic
- Add an *n2port* device in the frequency domain to a modified copy of the `s.param.first` schematic.
- Run another simulation of the modified circuit where the *n2port* component reads in the S parameter data file
- Plot and compare the two sets of data produced by the second simulation.
  - The output of the original branch of the circuit
  - The output of the modified branch of the circuit where the *n2port* component replaced the components in the original branch. The n2port reads in the `sparam.practice` S-parameter file you created.

  When you superimpose the two plots, it is clear that the S-parameter file produces the same plot as the original components.

Setting Up the EF_example Schematic for the First Simulation

This example requires that you

- Open the *EF_example* schematic
- Create a new, empty cellview named `s.param.first`
- Copy part of the *EF_example* schematic and paste it in the `s.param.first` schematic
- Simulate the new schematic.

This first simulation generates the S-parameters of a linear time-invariant part of the original circuit (*EF_example*). The S-parameter file is stored so you can use it in the second simulation to import S-parameters.

Opening the EF_example Schematic

1. Open the *EF_example* schematic as described in “Opening the EF_example Circuit in the Schematic Window” on page 684.

   The *EF_example* schematic is the original schematic used in the first Envelope Following example in this chapter. Leave the *EF_example* schematic open in the background.

Creating and Editing the New Schematic

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

2. In the New File form, do the following:

   a. Choose *my_rfExamples* for *Library Name,* the editable copy of *rfExamples*.

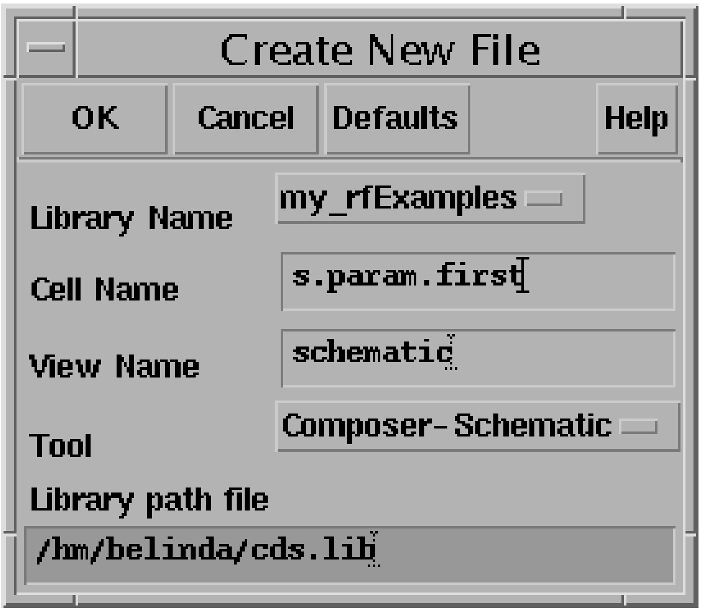
      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

   b. In the *Cell Name* field, type `s.param.first,` a name for the new view.

   c. In the *View Name* field, type `schematic`.

   d. Choose *Composer-Schematic* for *Tool*.

e. Click *OK*.



After you click *OK,* the new empty Schematic window for *s.param.first* appears.

3. In the *EF_example* Schematic window, copy the part of the schematic shown in Figure 10-19 and paste it into the new Schematic window.

a. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts at the bottom of the Schematic window.

b. Following the prompt,

> *point at object to copy*

left click and drag to create a box around the part of the *EF_example* circuit indicated in Figure 10-19 on page 788.

The selected *EF_example* components are highlighted in yellow.

c. Following the prompt,

> *point at reference point for copy*

click inside the outlined elements.

d. Following the prompt,

> *point at destination point for copy*

move the cursor to the *s.param.first* schematic window and click there. This copies the selected part of the *EF_example* circuit into the empty Schematic window.

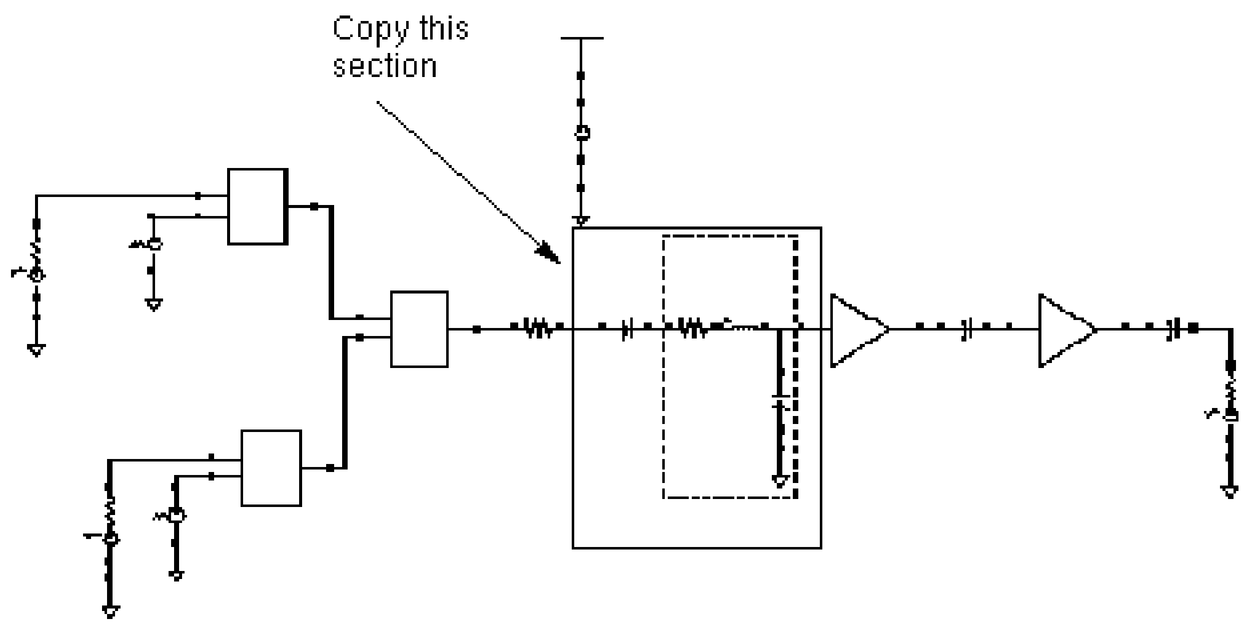
The empty Schematic window now contains the portion of the *EF_example* schematic shown in Figure 10-19 on page 788.

e. If necessary, choose *Window - Fit* to center the copied section of the *EF_example* circuit in the new Schematic window.

4. In the original *EF_example* Schematic window, choose *Window - Close*.

The *EF_example* Schematic window closes.

Figure 10-19 Portion of EF_example Schematic to Copy

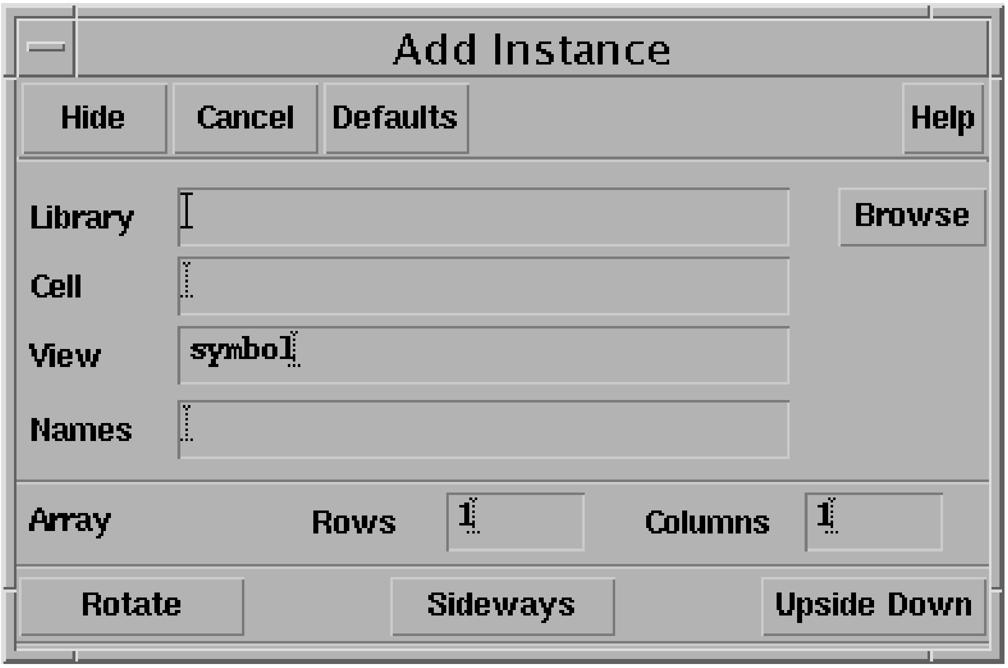


Note: If you need assistance with methods for editing the schematic, see the *Virtuoso® Schematic Composer™User Guide*.

Adding Components to the Schematic

1. In the `s.param.first` Schematic window, choose *Add – Instance.*

The Add Instance form appears.



2. In the Add Instance form, click on *Browse*.

   The Library Browser – Add Instance form appears.

3. In the Library Browser – Add Instance form, do the following:

   a. Select the *analogLib* and highlight *Show Categories*.

   b. Scroll down and select *Sources* in the *Category* list box.

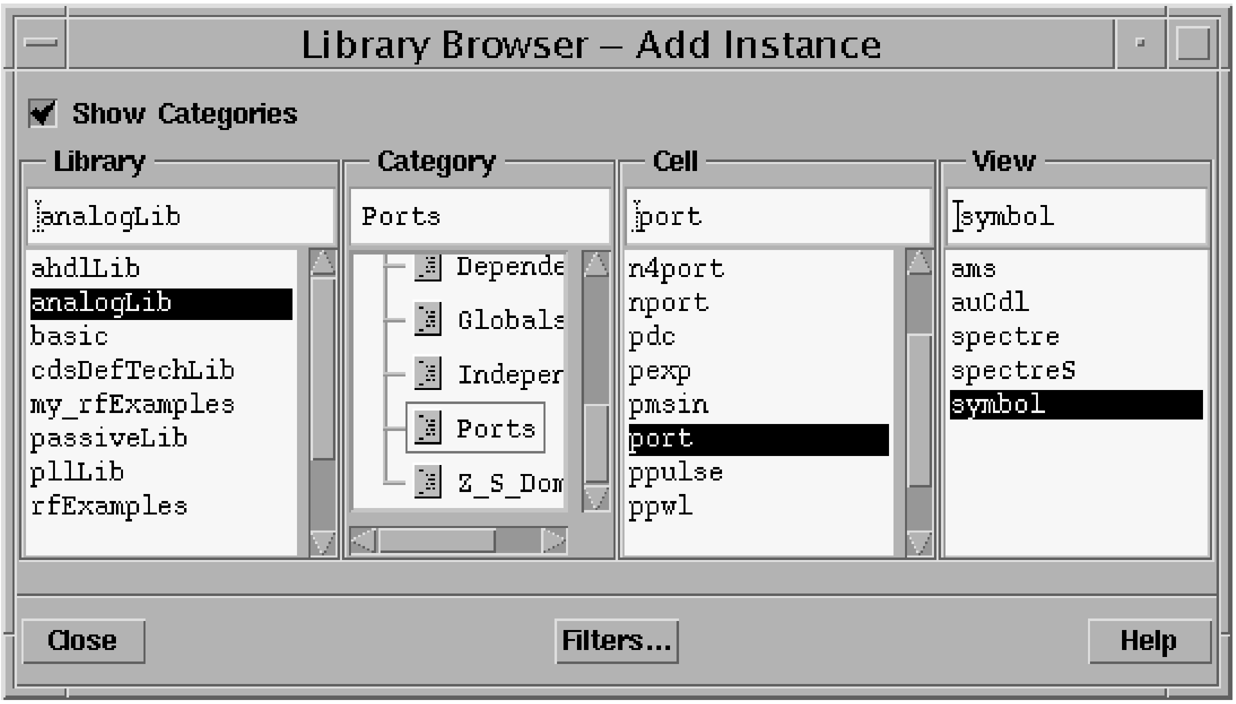
      The *Category* list box expands to display several subcategories under *Sources*.

   c. Scroll down and select the *Ports* subcategory under *Sources*.

   d. Select *port* for *Cell*.

   e. Select *symbol* for *View*.

The Library Browser – Add Instance form now looks like the one below:



As you move the cursor into the new Schematic window, a copy of the port component moves with the cursor.

4. Place two *port* symbols in the new Schematic window as shown in Figure 10-20 on page 791.

5. In the Library Browser – Add Instance form, do the following:

   a. Select *analogLib* for *Library*.

   b. Select *Everything* for *Category*.

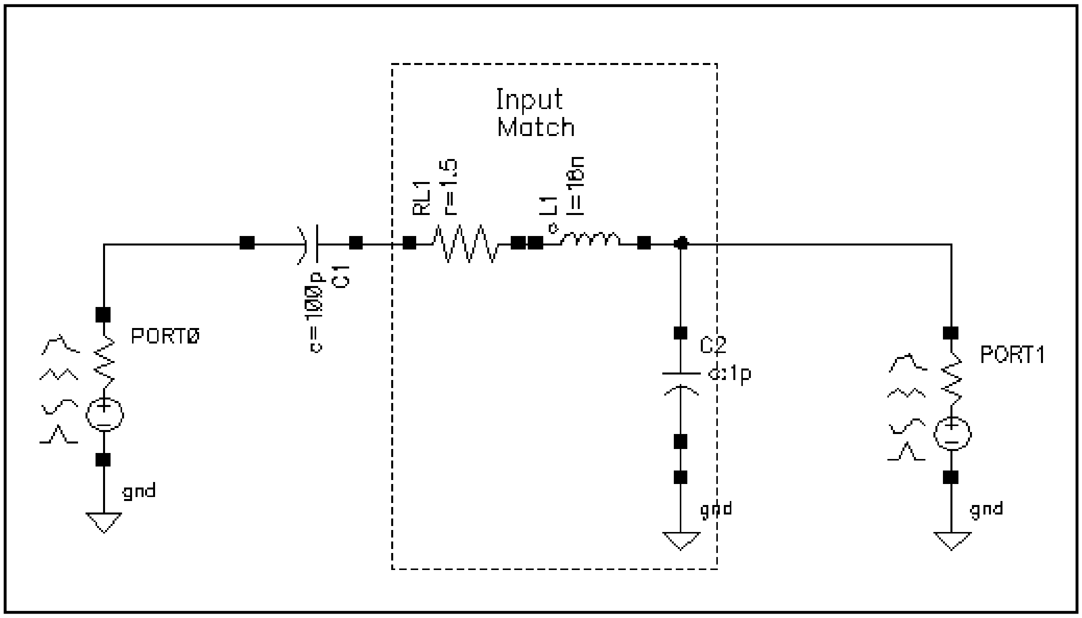
   c. Select *gnd* for *Cell*.

   d. Select *symbol* for *View*.

6. Place two *gnd* components in the Schematic window as shown in Figure 10-20 on page 791. Press *Esc* when you are done.

7. In the Schematic window, select *Add – Wire (narrow)* and wire up the new components in the *s.param.first* Schematic window. Press *Esc* when you are done.

Figure 10-20 Placement of Ports in the New Schematic Window



8. In the new Schematic window, choose *Edit – Properties – Objects*.

9. For each of the *port* components in the new schematic, do the following in the Edit Object Properties form:

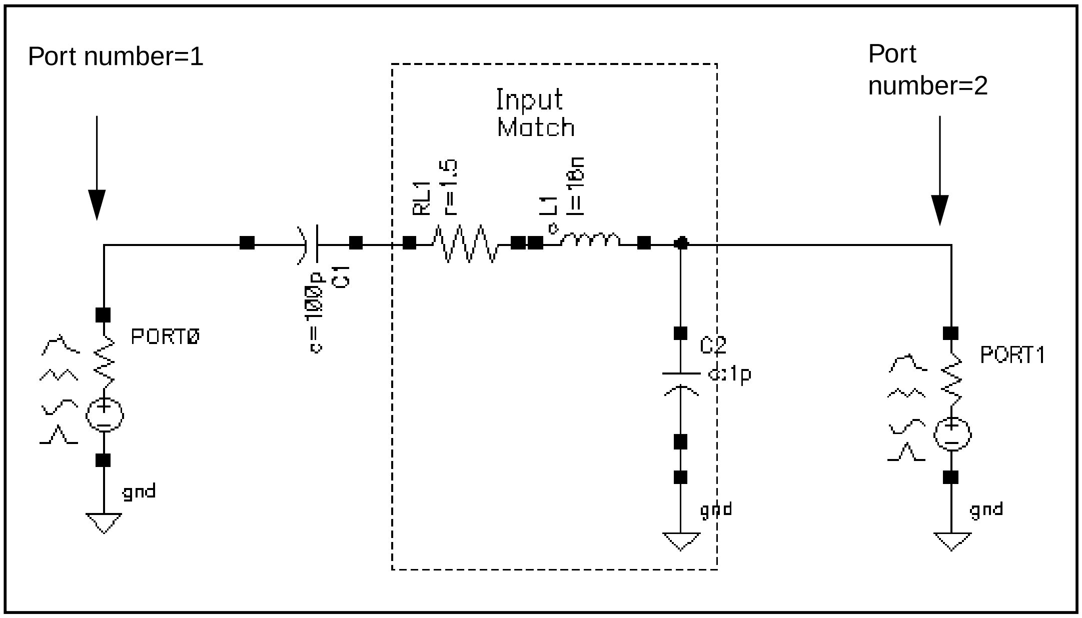
   a. Select the *port*.

   b. Type the appropriate number in the *Port number* field.

      Figure 10-21 on page 792 shows the appropriate number to type. Leave the Port resistances at their default of 50 Ohms.

   c. Click on *Apply*.

Figure 10-21 Port Numbering in New Schematic



10. In the Edit Object Properties form, click *OK*.

11. In the new Schematic window, choose *Design – Check and Save*.

Setting Up the s.param.first Schematic

1. In the *s.param.first* Schematic window, open the Simulation window as described in "Opening the Simulation Window" on page 685.

2. In the Simulation window, set up the model libraries as described in "Setting Up the Model Libraries" on page 686.

Running the SP Simulation

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, do the following:

   a. Highlight *sp* for *Analysis*.

   b. Highlight *Frequency* for *Sweep Variable*.

   c. Highlight *Start – Stop* for *Sweep Range*.

   d. Type `100K` for *Start* and `10G` for *Stop*.

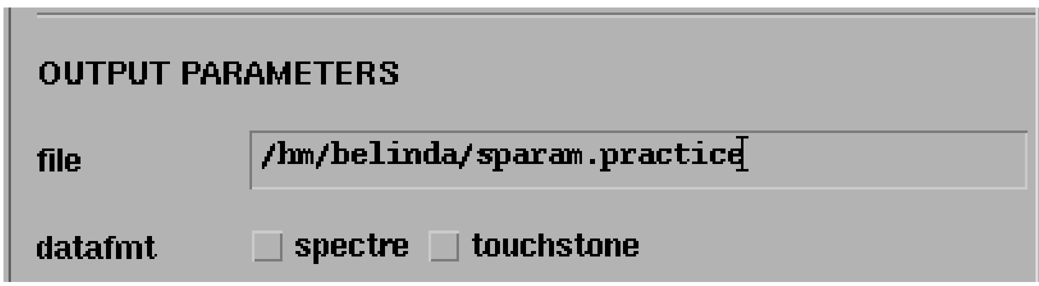
   e. Choose *Automatic* for *Sweep Type*.

   f. Highlight *no* for *Do Noise*.

3. In the Choosing Analyses form, click on *Options.*

   The S-parameter Options form appears.

4. In the OUTPUT PARAMETERS section of the S-parameter Options form, type the path for the output S-parameter file in the *file* field. Then click *OK*.



5. In the Choosing Analyses form, click *OK*.

6. In the Simulation window, select *Simulation – Netlist and Run*.

7. Check the CIW for a message that says the simulation completed successfully.

   The sp analysis wrote an S-parameter file, `sparam.practice`, to the directory you indicated.

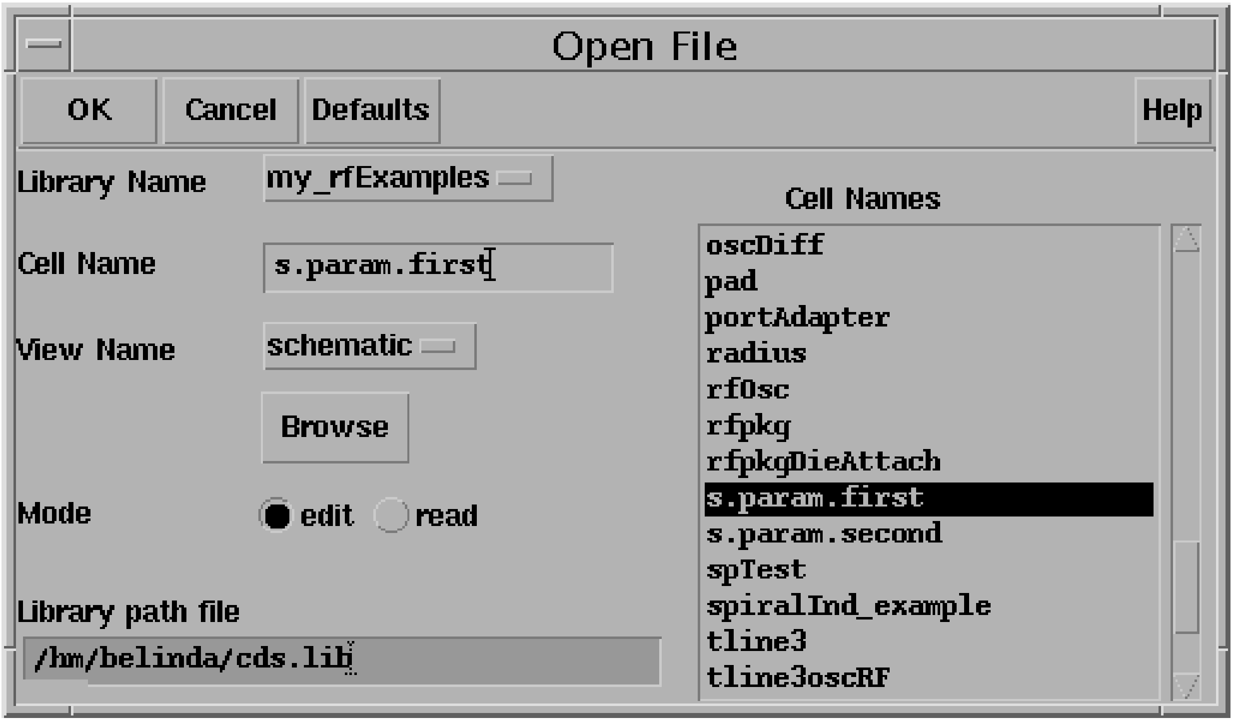
The S-Parameter File

Use the S-parameter output file you created as input for the next simulation. You can use a text editor to open and examine the format of the S-parameter output file. You can use this format to create S-parameter files from other sources.

Setting Up and Running the Second sp Simulation

In this simulation, use the S-parameter output file, `sparam.practice`, from the first simulation as input to the second simulation. The S-Parameter file models the components in the `s.param.first` Schematic.

1. If necessary, open the `s.param.first` schematic.



2. Add components to the *s.param.first* schematic you created for the SP analysis, as described in Table 10-5. Place the components as shown in Figure 10-22. See "Adding Components to the Schematic" on page 788 for information on adding components to a schematic.

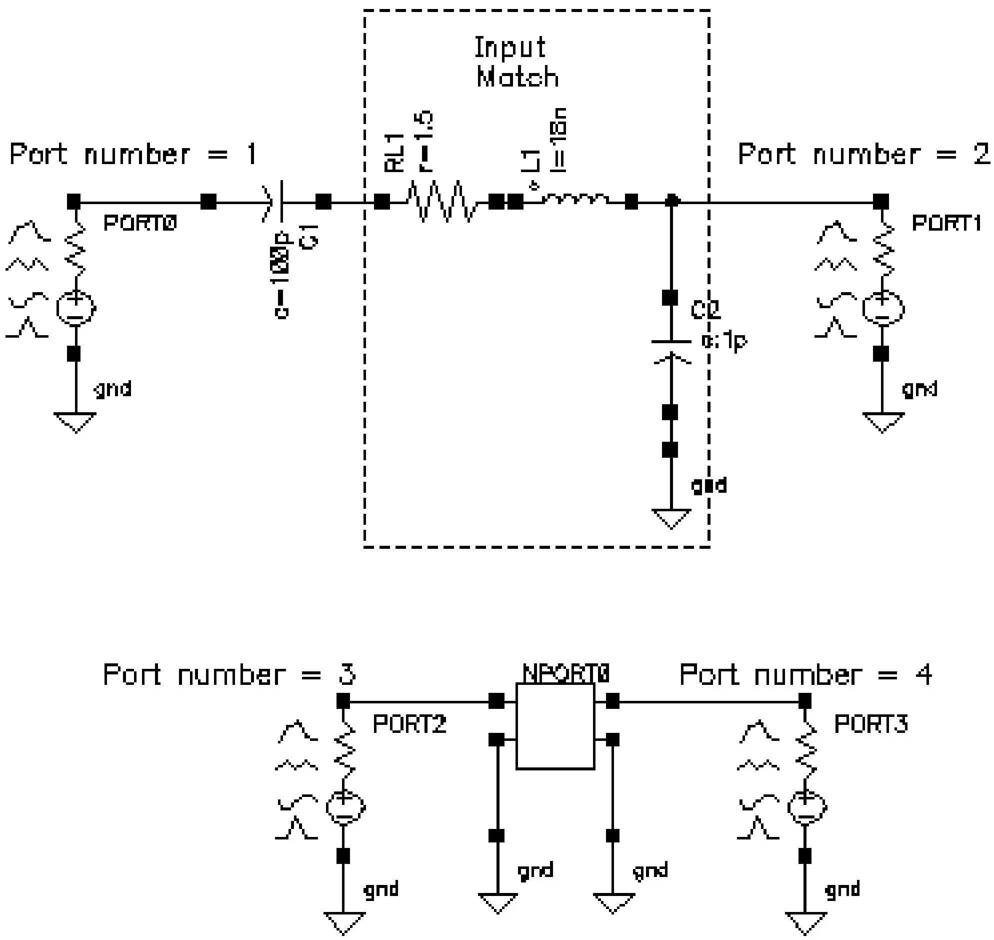
.

Table 10-5 Components to Add to the Schematic

Library	Category	Subcategory	Cell	View	Number to Add
analogLib	Sources	Ports	n2port	symbol	1
analogLib	Sources	Ports	port	symbol	2
analogLib	Everything	None	gnd	symbol	4

3. Wire the components as shown in Figure 10-22.

Figure 10-22 Wired s.param.first Schematic



4. Edit the properties on the two new ports to assign port numbers 3 and 4 as shown in Figure 10-22. Then edit the properties on the new *n2port*.

a. In the Schematic window select the *port* on the lower left.

b. In the Schematic window, choose *Edit – Properties – Objects.*

c. In the Edit Object Properties form, do the following:

❍ Type 3 in the *Port number* field.

Figure 10-22 on page 795 shows the appropriate number to type. Leave the *Port resistance* at the default of 50 Ohms.

❍ Click *Apply*.

❍ In the Schematic window, select the *port* on the lower right.

- Type 4 in the *Port number* field.

  Leave the *Port resistance* at the default of 50 Ohms.

- Click *Apply*.

d. In the Schematic window, select the *n2port*.

The Edit Object Properties form changes to display information for the *n2port*.

e. In the Edit Object Properties form, do the following and click *OK*:

- In the *S-parameter data file* field, type the absolute path to the S-parameter file you created in the first simulation.

  `/hm/belinda/sparam.practice`

- Choose *rational* for *Interpolation method*.

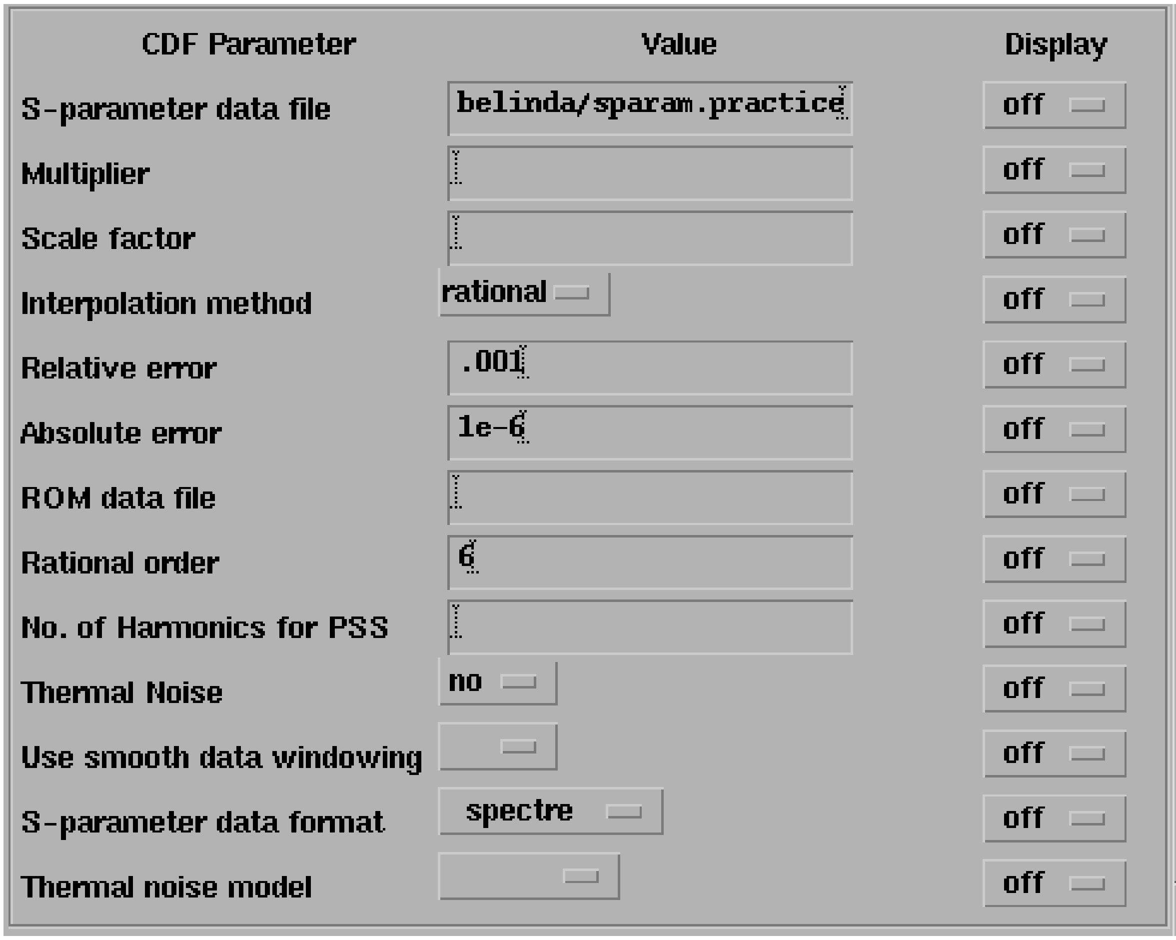
  The form changes to let you add additional information.

- Type `.001` for *Relative error*.
- Type `1e`$^{-6}$ for *Absolute error*.
- Type 6 for *Rational order*.

Note: It is mandatory that you use the *Rational* Interpolation method if you plan to run SpectreRF.

- Select *no* for *Thermal Noise*.
- Select *spectre* for *S-parameter data format*.

The completed Edit Object Properties form for the *n2port* looks like this.



CDF Parameter	Value	Display
S-parameter data file	belinda/sparam.practice	off
Multiplier		off
Scale factor		off
Interpolation method	rational	off
Relative error	.001	off
Absolute error	1e-6	off
ROM data file		off
Rational order	6	off
No. of Harmonics for PSS		off
Thermal Noise	no	off
Use smooth data windowing		off
S-parameter data format	spectre	off
Thermal noise model		off

5. In the Schematic window, choose *Design – Check and Save*.

6. In the Simulation window, choose *Simulation – Netlist and Run*.

Plotting Results

1. In the Simulation window, choose *Results – Direct Plot – Main Form.*

The Direct Plot form appears.

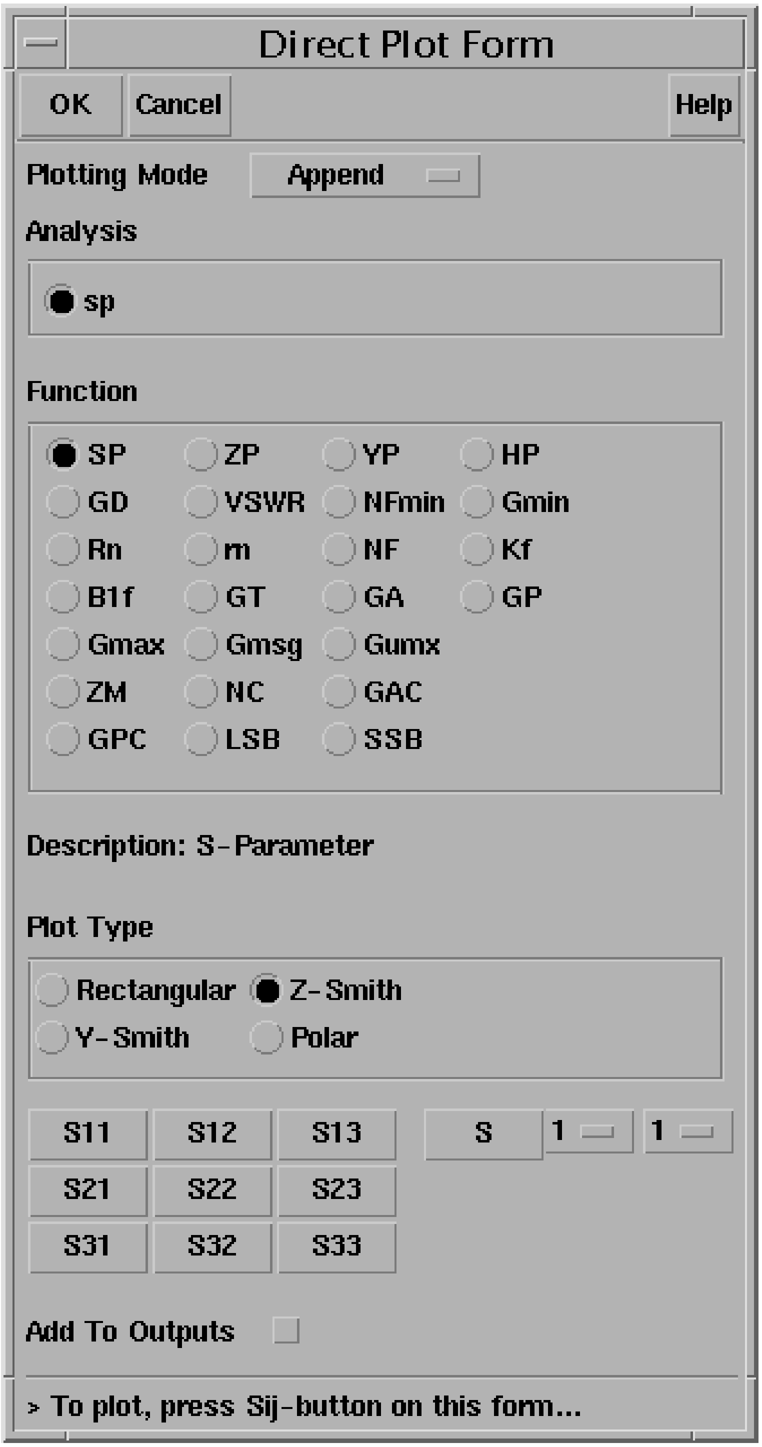
2. In the Direct Plot form, do the following:

a. Choose *Replace* for *Plot Mode*.

b. Highlight *SP* for *Function*.

c. Highlight *Z-Smith* for *Plot Mode*.

The Direct Plot form looks like the following.



d. Click *S11*.

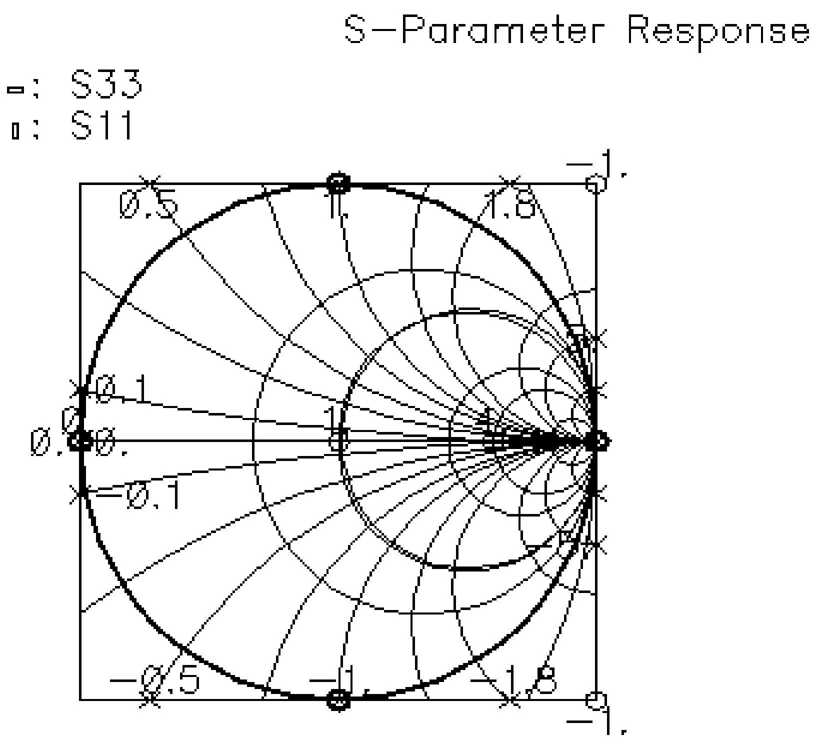
The plot for S11 appears in the Waveform window.

3. In the Direct Plot form, do the following:

a. Choose *Append* for *Plot Mode*.

b. Click on *S33*.

The plot for `S33` is appended to the `S11` plot. The two plots lie one on top of the other which shows that the two plots are identical. Thus the results produced by the first simulation are the same as those produced by the second simulation which used the S-parameter input file.



You can also check other plots for equivalency. For example, you can plot S21 and S43.

- First plot S21 as described in “Plotting Results” on page 797.

- Then plot S43. Choose Append for Plot Mode. Then plot S43, choose 4 and 3 in the cyclic fields on the right. Then click S.

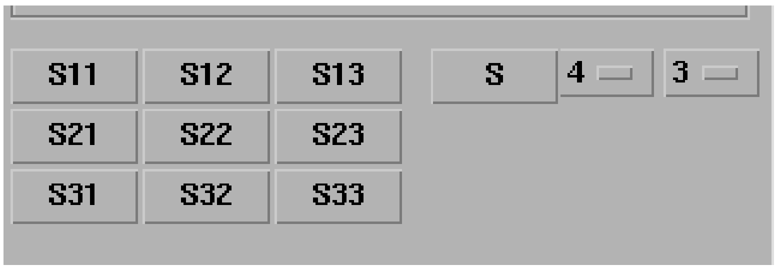
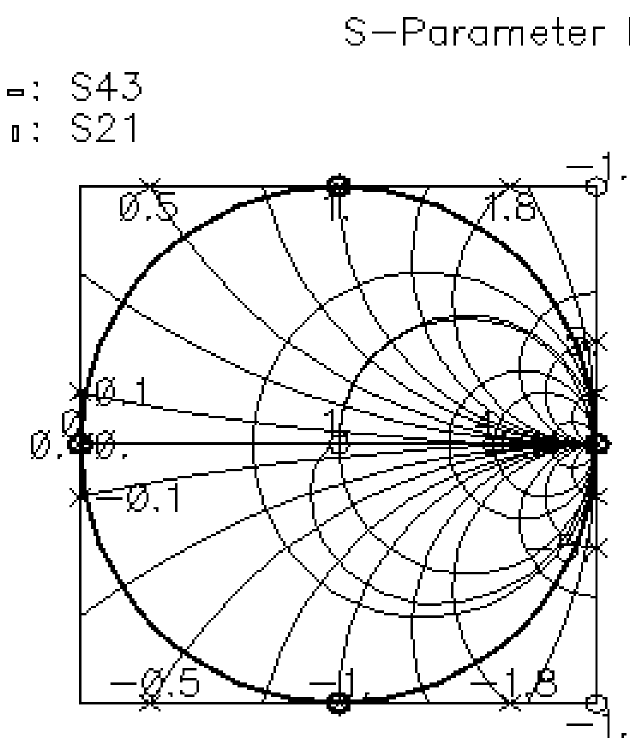


Figure 10-23 is the plot produced by appending `S21` to the `S43` plot.

Figure 10-23 Plot Showing S21 and S43



Using an S-Parameter Input File with a SpectreRF Envlp Analysis

This section uses the `s.param.practice` S-parameter data file as input to an Envelope Following Analysis. Again, it compares results for an *envlp* analysis using the S-parameter file to results for an *envlp* analysis using the original components.

Setting Up the Schematic

1. If necessary, open the *EF_example* schematic. Use *File – Open* in the CIW.
2. In the *EF_example* schematic. choose *Design – Save As* in the Schematic window.

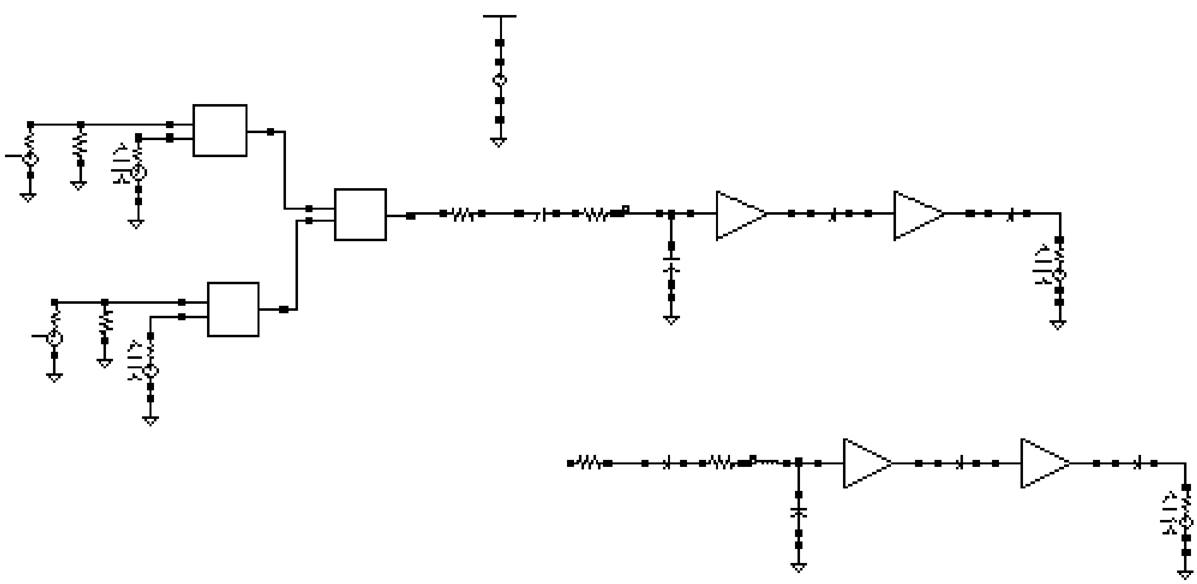
   The Save As form appears.
3. In the Save As form, do the following:

   a. In the *Library Name* field, type the name of your local, editable copy of the *rfExamples* library.

   b. In the *Cell Name* field, type *EF_example_copy,* the name for the copy of the *EF_example* schematic.

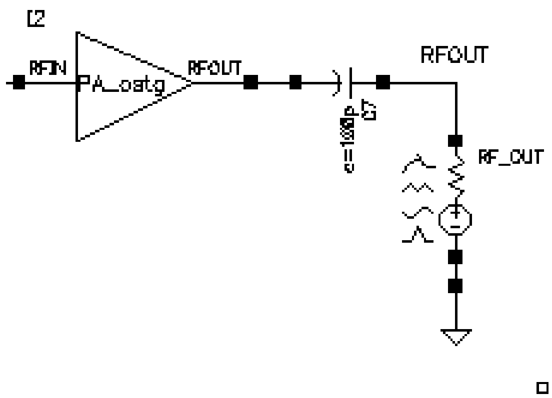
   c. Click *OK*.

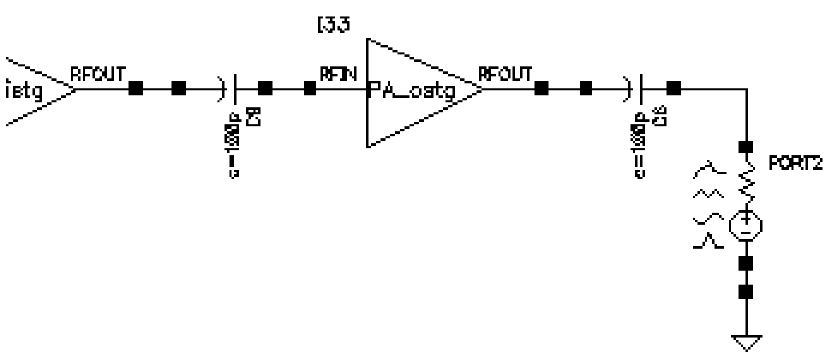
   You now have a copy of the *EF_example* schematic called *EF_example_copy*. In the *EF_example* Schematic window, choose *Window – Close* to close the *EF_example* schematic.
4. In the *EF_example_copy* Schematic window, copy everything to the right of the adder and paste the copy below the original.



Important

Be sure to remove the *RFOUT* label from the duplicate branch or the two branches will be shorted together at their outputs.





5. In the lower branch of the schematic, delete the circuitry you used to create the S-parameter file and replace it with an *n2port* component.

a. In the Schematic window, choose *Edit – Delete* and follow the prompts at the bottom of the Schematic window.

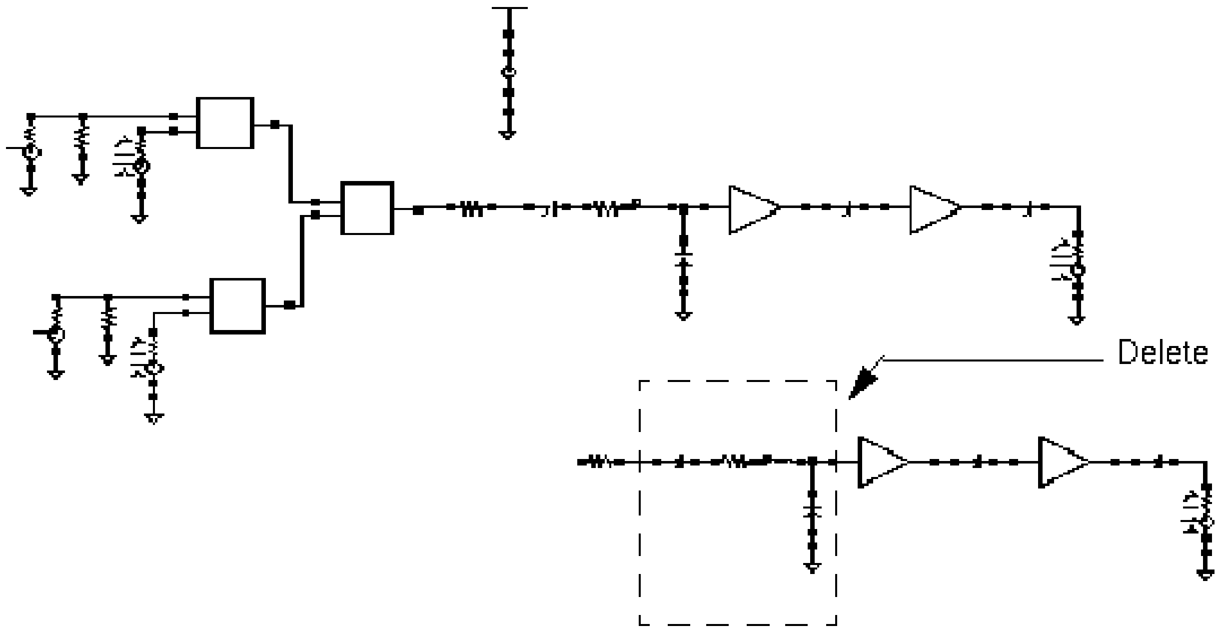
b. Following the prompt,

> *point at object to delete*

left click and drag to create a box around the indicated part of the *EF_example_copy* circuit.

The selected *EF_example_copy* components are deleted.

c. Press *Esc* to stop deleting.



6. Replace the deleted components in the *EF_example_copy* schematic with the a copy of the *n2port* component and the two attached *gnd* (ground) components from the *s.param.first* schematic.The *n2port* and *gnd* components are shown in Figure 10-22 on page 795.

a. In the CIW, choose *File – Copy* to open the *s.param.first* schematic.

b. In the *s.param.first* schematic.window, choose *Edit – Copy* to copy the *n2port* and two *gnd* components. Follow the prompts at the bottom of the *s.param.first* schematic window. (Close the *s.param.first* schematic window when you are done.)

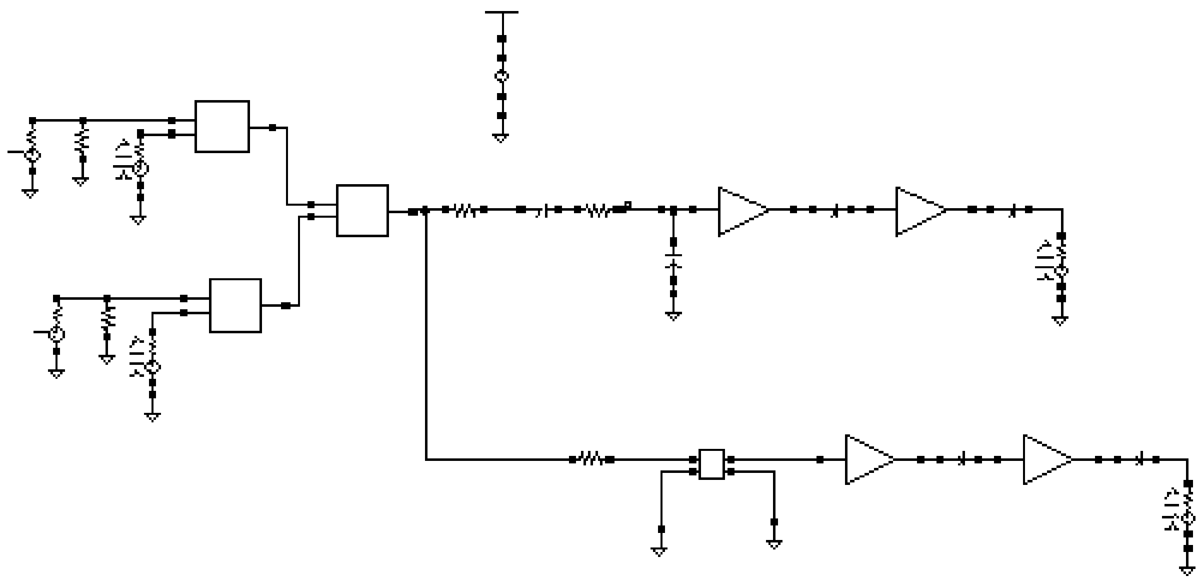
c. As you move the cursor into the *EF_example_copy* Schematic window, copies of the *n2port* and *gnd* components move with the cursor.

❍ Place the components in the *EF_example_copy* Schematic window as shown in Figure 10-22 on page 795.

❍ Press *Esc* to stop the copy operation.

7. In the Schematic window, select *Add – Wire (narrow)* and wire up the new components in the *EF_example_copy* Schematic window. Press *Esc* when you are done.

Figure 10-24 The EF_example_copy Schematic with n2port Component in Place.



8. In the Schematic window, select the *n2port* component and choose *Edit – Properties – Object*. Then verify that the *n2port* component has the following properties.

   a. The absolute path to the S-parameter data file displays in the S-parameter data file field. In this example,

      `/hm/belinda/sparam.practice`

   b. *Interpolation method* is *Rational*.

   c. *Relative error* is `.001`.

   d. *Absolute error* is `1e`$^{-6}$.

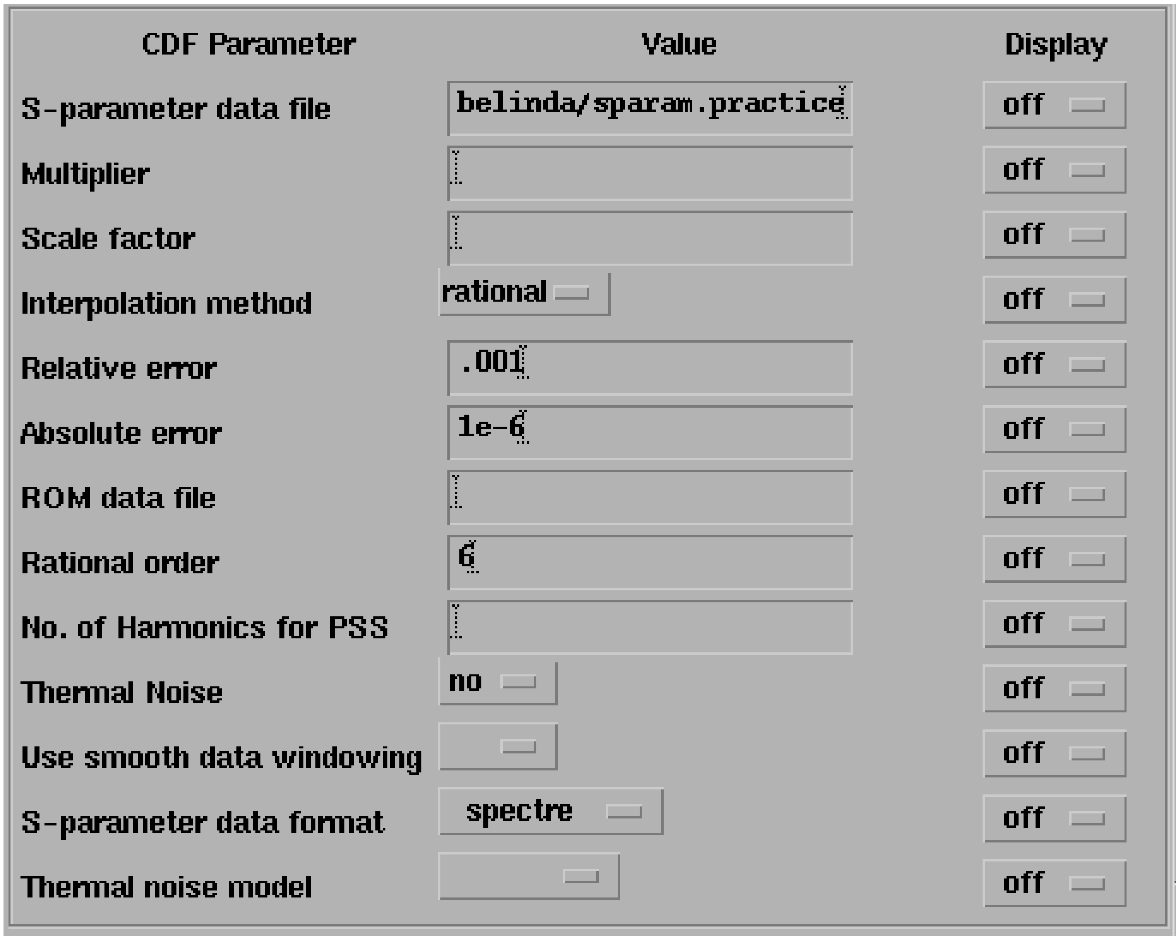
   e. *Rational order* is `6`.

      Note: It is mandatory that you use the *Rational* Interpolation method if you plan to run SpectreRF.

   f. *Thermal Noise* is *no*.

   g. *S-parameter data format* is *spectre*.

The completed Edit Object Properties form for the *n2port* looks like this.



9. In the Schematic window, choose *Design – Check and Save*.

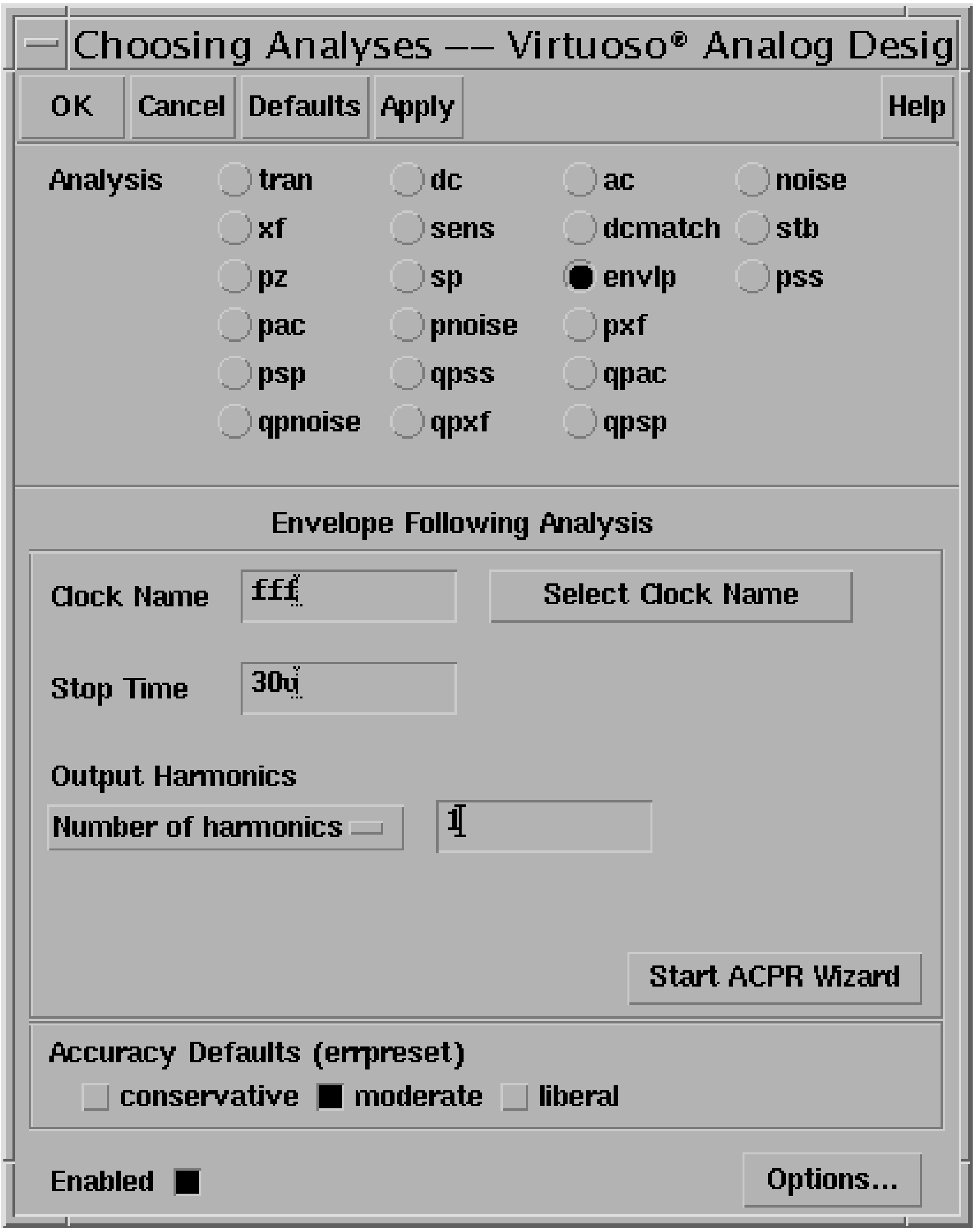
Setting Up and Running the Simulation

Set up and run an *envlp* analysis as follows.

- "Opening the Simulation Window" on page 685
- "Setting Up the Model Libraries" on page 686
- "Editing PORT0 and PORT1 in the Schematic Window" on page 687
- "Setting Up an Envelope Following Analysis" on page 689.

Note: When you set up the *envlp* analysis, use 30u for the *Stop Time* value.

The completed *envlp* Choosing Analysis form looks like this.



➤ In the Simulation window, choose *Simulation – Netlist and Run*.

Check the CIW for a message that says the simulation completed successfully.

Displaying the envlp Following Results

1. In the Simulation window, choose *Results – Direct Plot – envlp*.

   The Direct Plot form appears.

2. In the Direct Plot form, do the following:

   a. Choose *Replace* for *Plot Mode*.

   b. Highlight *envlp* for *analysis.*

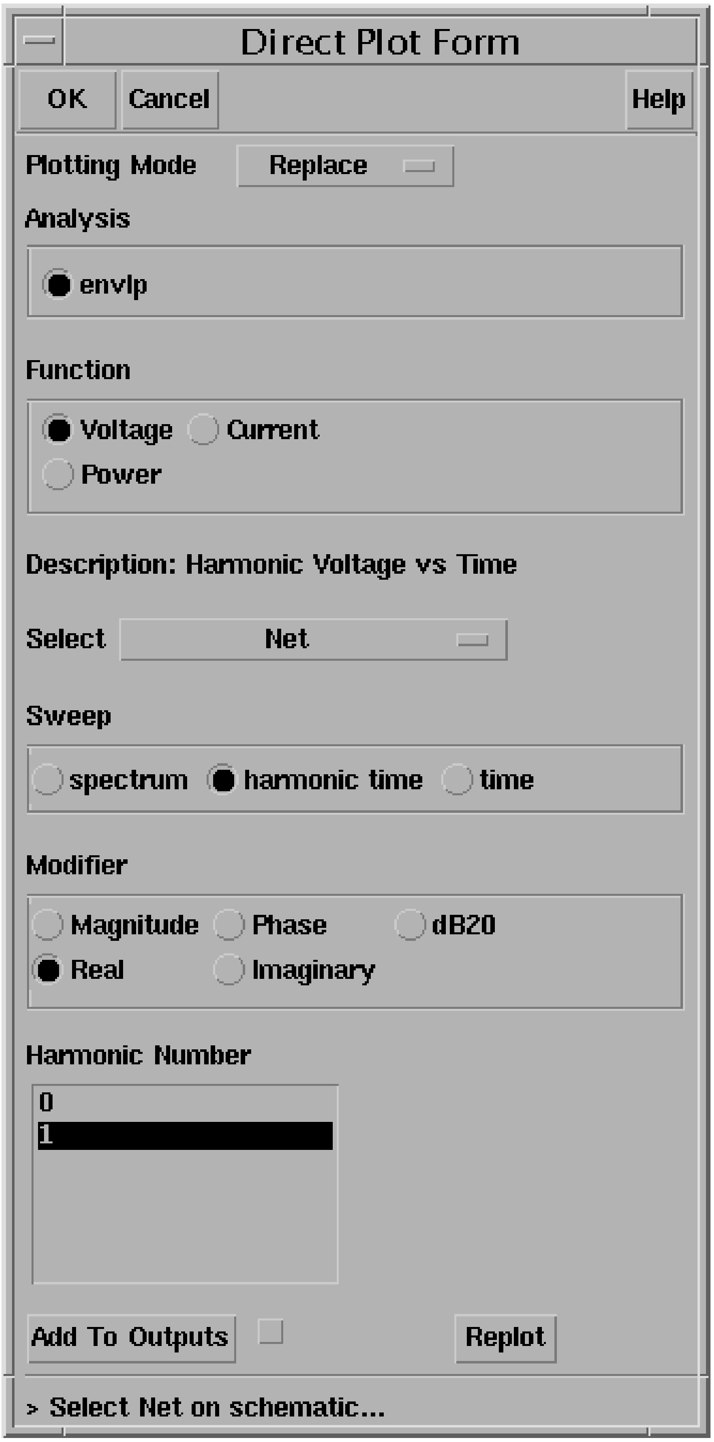
   c. Highlight *Voltage* for *Function*.

   d. Highlight *Harmonic Time* for *Sweep*.

   e. Highlight *Real* for *Modifier*.

   f. Choose *1* for *Harmonic Number*.

The completed form looks like this.



3. In the Schematic window,

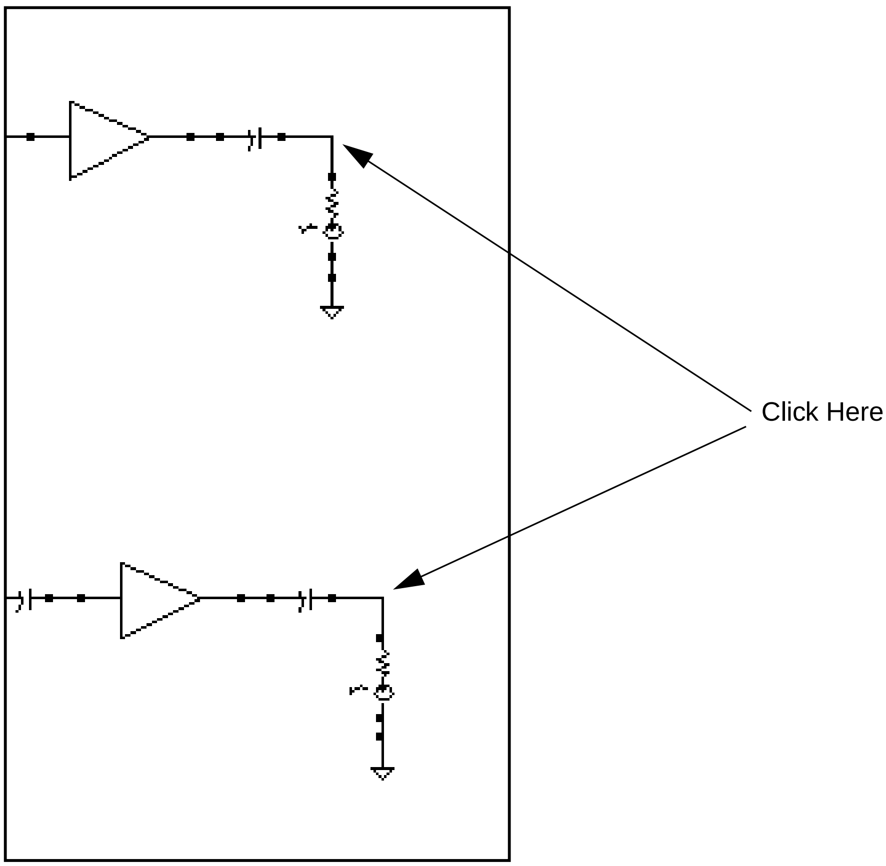
a. Click on one of the output nodes as shown in “Output Nodes” on page 809.

b. Change the *Plot Mode* to *Append.*

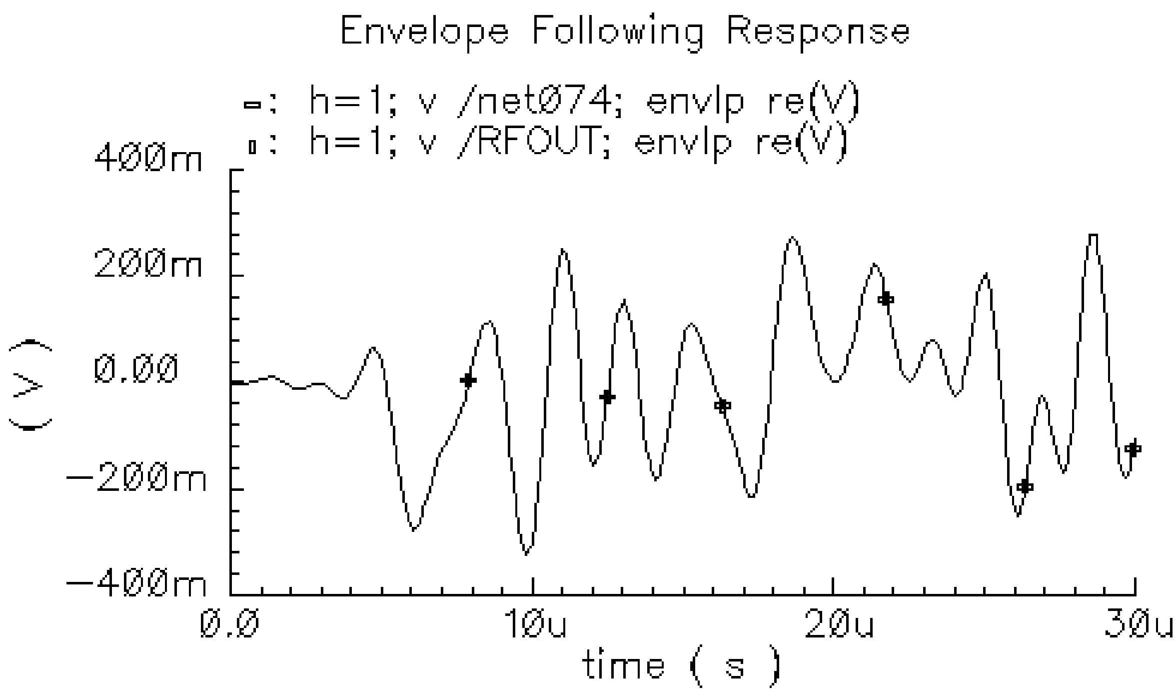
c. Click on the other output node as shown in “Output Nodes” on page 809.

You can also confirm that the imaginary parts of the waveforms match.

Figure 10-25 Output Nodes



The coincident plots in the Waveform window show that the output is the same at both nodes. When the S-parameters are correct, the *n2port* device accurately simulates the circuitry represented by the S-parameters.



Measuring AM and PM Conversion for the PAC and PXF Analyses

This section describes AM and PM small signal characterization in SpectreRF. AM/PM conversion measurements allow you to investigate the amplitude and phase characterization of RF circuits. The modulated PAC and PXF analyses calculate conversion gain and other characteristics between AM, PM, and SSB sources and AM, PM, and SSB outputs.

AM/PM conversion computes transfer functions and gain measurements involving AM and PM inputs and outputs. In general, there are three possible types of inputs and outputs for which you might want to compute transfer functions.

- Unmodulated or single sideband (SSB)
- Amplitude modulated sinusoids (AM)
- Phase modulated sinusoids (PM)

The complete analysis can calculate 9 cross conversion metrics from 3 types of inputs (AM, PM and SSB) to 3 types of outputs (AM, PM and SSB). Use the *Modulated Analysis* section in the PAC and PXF Choosing Analysis forms to set up your analyses.

- *Input Type* (for PAC analysis) and *Output Type* (for PXF) allow you to choose whether to measure all 9 modulated conversions (choose *SSB/AM/PM*) or only 3 conversions (choose *SSB*).
- *Output Modulated Harmonic List* (for PAC output modulations) and *Input Modulated Harmonic List* (for PXF modulated sources) specify a vector of harmonic indexes. You can type the indexes separated by spaces or you can choose them from a scrolling list.
- *Input Modulated Harmonic* (for PAC) and *Output Modulated Harmonic* (for PXF) specify a single harmonic index for PAC input source modulation or for PXF output modulation. You can type the index or you can choose it from a scrolling list. This choice appears when you select the *SSB/AM/PM Input Type* or *Output Type*.

  For *SSB Input Type* (PAC) or *Output Type* (PXF), specify an *Output Upper Sideband* or *Input Upper Sideband.* You can type the sideband index or you can choose it from a scrolling list.
- For PAC analysis, select the *Modulated Input Source* from the schematic.

After you run the simulation, use the Direct Plot form to plot the modulated analysis results.

Creating and Setting Up the EF_AMP Circuit

This example illustrates how to

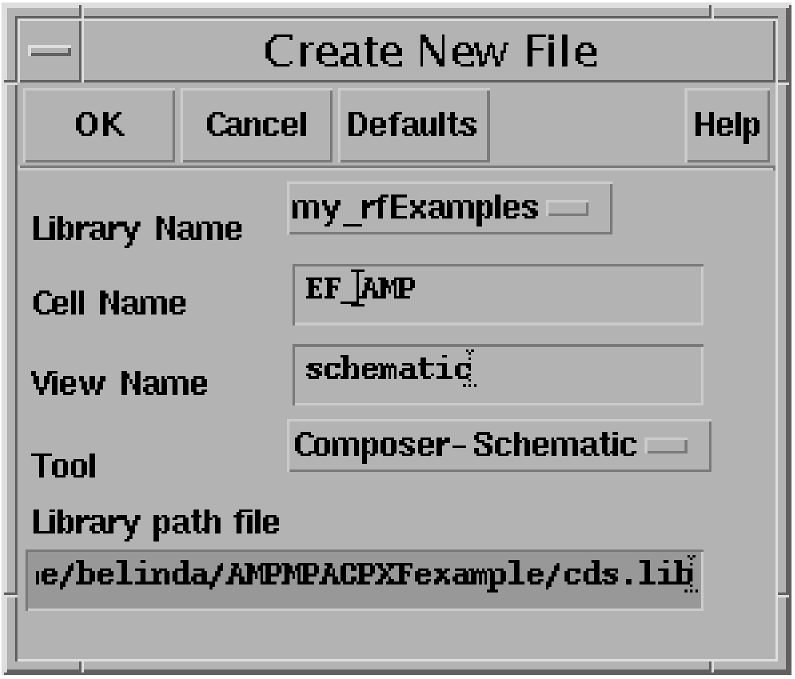
- Create a modified copy of the *EF_example* schematic, *EF_AMP*, for this example.
- Set up and run the necessary PSS, modulated PAC, and modulated PXF analyses.
- Compute transfer functions and gain measurements and display the resulting information with the Direct Plot form.

Before you start, perform the setup procedures described in Chapter 3. Next, open the *EF_example* schematic in another window and copy the *EF_example* circuit into this empty EF_AMP Schematic window.

Creating a New Empty Schematic Window

1. In the CIW choose *File—New—Cellview*.

The Create New File form appears.



2. In the Create New File form, do the following:

a. Choose *my_rfExamples* for *Library Name*.

Select *my_rfExamples*, the editable copy of the *rfExamples* library you created following the instructions in <u>Chapter 3</u>.

b. In the *Cell Name* field, enter *EF_AMP*.

c. In the *View Name* field, enter *schematic*.

d. In the *Tool* cyclic field, select *Composer-Schematic*.

e. Click *OK*.

A new, empty Schematic window named *EF_AMP* opens.

Opening and Copying the EF_example Schematic

Copy of the *EF_example* schematic into the empty *EF_AMP* Schematic window. Then edit *EF_AMP* before simulating.

Create the EF_AMP Schematic

1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, do the following:

   a. Choose *my_rfExamples* for *Library Name*.

      Select the editable copy of the *rfExamples* library you created following the instructions in Chapter 3.

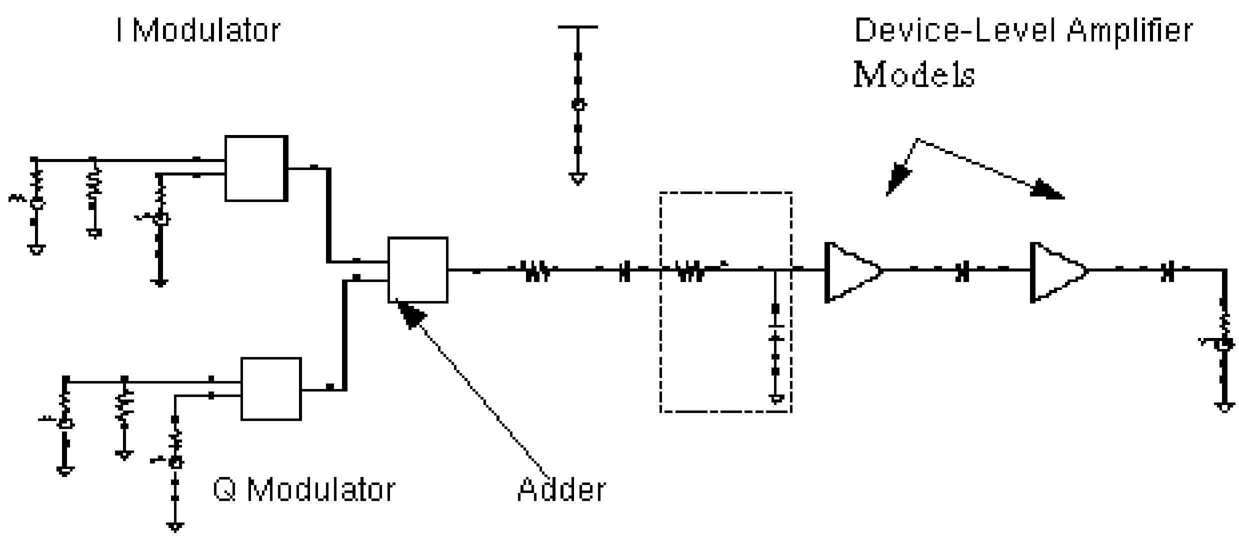
   b. In the *Cell Name* list box, highlight *EF_example*.

   c. Choose *schematic* for *View Name*.

   d. Highlight *edit* for *Mode*.

   e. Click *OK*.

   The Schematic window appears with the *EF_example* schematic.



3. In the *EF_example* Schematic window, choose *Edit - Copy* and follow the prompts.

4. Following the prompt at the bottom of the Schematic window,

   > *point at object to copy*

   left click and drag to create a box around the entire *EF_example* circuit.

   The *EF_example* components are highlighted in yellow.

5. Following the prompt,

   > *point at reference point for copy*

   click inside the outlined elements.

6. Following the prompt,

   > *point at destination point for copy*,

   move the cursor to drag a copy of the entire circuit into the *EF_AMP* Schematic window and click there.

   The *EF_AMP* Schematic window now contains a copy of the *EF_example* schematic.

7. If necessary, choose *Window - Fit* to scale and center the circuit in the *EF_AMP* Schematic window.

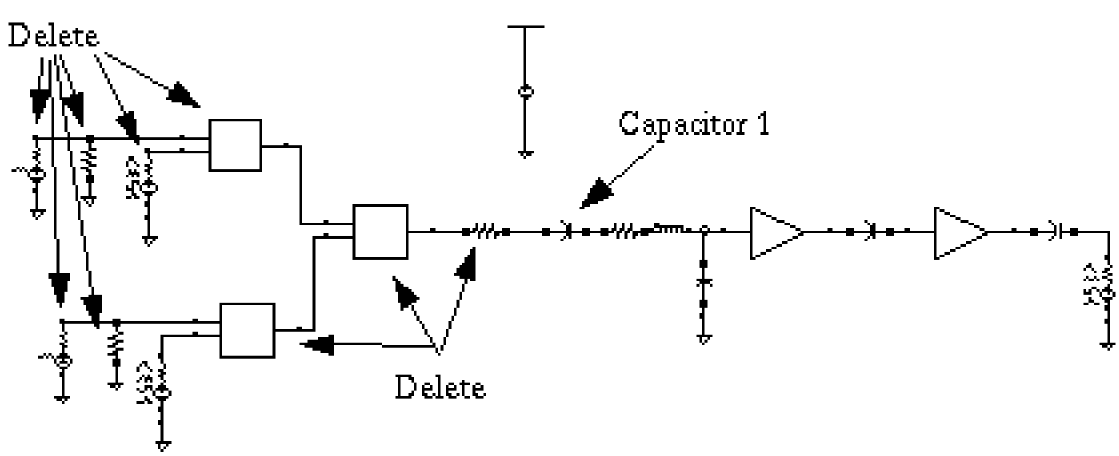
8. In the *EF_example* Schematic window, choose *Window - Close*.

   The *EF_example* Schematic window closes.

Editing the EF_AMP Schematic

In the *EF_AMP* Schematic window, delete components and their associated connecting wires as shown in Figure 10-26. The final *EF_AMP* Schematic should look like Figure 10-27 on page 816.

Figure 10-26 Components and Wires to Delete from the EF_LoadPull Schematic



Note: If you need assistance with methods for editing the schematic, see the Virtuoso® Schematic Composer™ User Guide.

Delete Components and Wires

1. In the *EF_AMP* Schematic window, choose *Edit – Delete.*
2. Click on a component or wire to delete it.
3. Press the *Esc* key when you are done deleting.
4. In the Schematic window, choose *Edit – Move* to move both *PORT 3* and *VCC* as shown in "The EF_LoadPull Schematic" on page 760.
   a. To move *PORT 3* closer to capacitor *C!,* click and drag to create a rectangle around *PORT 3* and its ground.

      PORT 3 and the *Gnd* are highlighted in yellow.
   b. Click and drag the components to move them close to capacitor C1 as shown in "The EF_LoadPull Schematic" on page 760.
   c. To move *VCC* below and between the two device-level amplifier models, click and drag to create a rectangle around *VCC*.

      VCC is highlighted in yellow.
   d. Click and drag the components to move them below and between the two device-level amplifier models as shown in "The EF_LoadPull Schematic" on page 760.
5. Choose *Window - Fit* to center the edited schematic in the window.

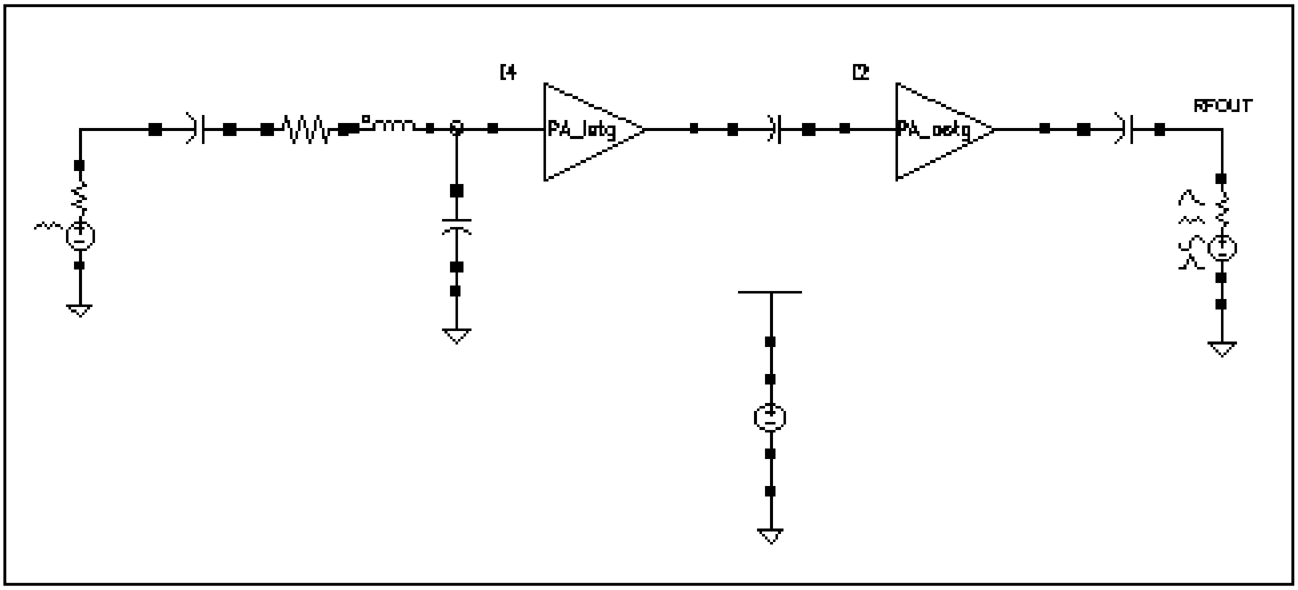
Wire the Schematic

In the Schematic window, connect *PORT3* to Capacitor *C1*.

1. In the Schematic window, choose *Add - Wire (Narrow)* and do the following.
   a. Click the terminal on *PORT 3* then click the terminal on *C1*.
   b. Press the Esc key to stop wiring.
2. Choose *Window -- Fit* to center the edited schematic in the window.

   The edited schematic looks like the one in Figure 10-8.

Figure 10-27 The Edited EF_AMP Schematic



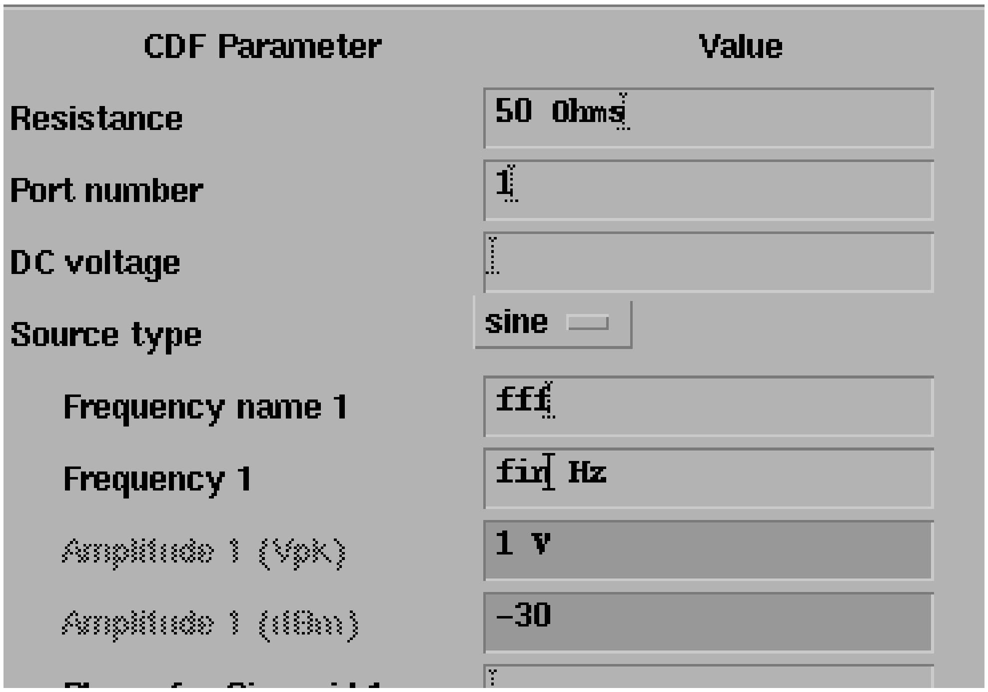
Edit CDF Properties for PORT3 and RFOUT

Edit CDF properties for both *PORT 3* and *RF_OUT*.

1. In the Schematic window, select *PORT 3*.
2. In the Schematic window, choose *Edit – Properties – Objects*.

   The Edit Object Properties form appears with information for *PORT 3* displayed.
3. In the Edit Object Properties form, do the following and then click *Apply*.
   a. Type *fin* for *Frequency 1*.

The completed form looks like this.

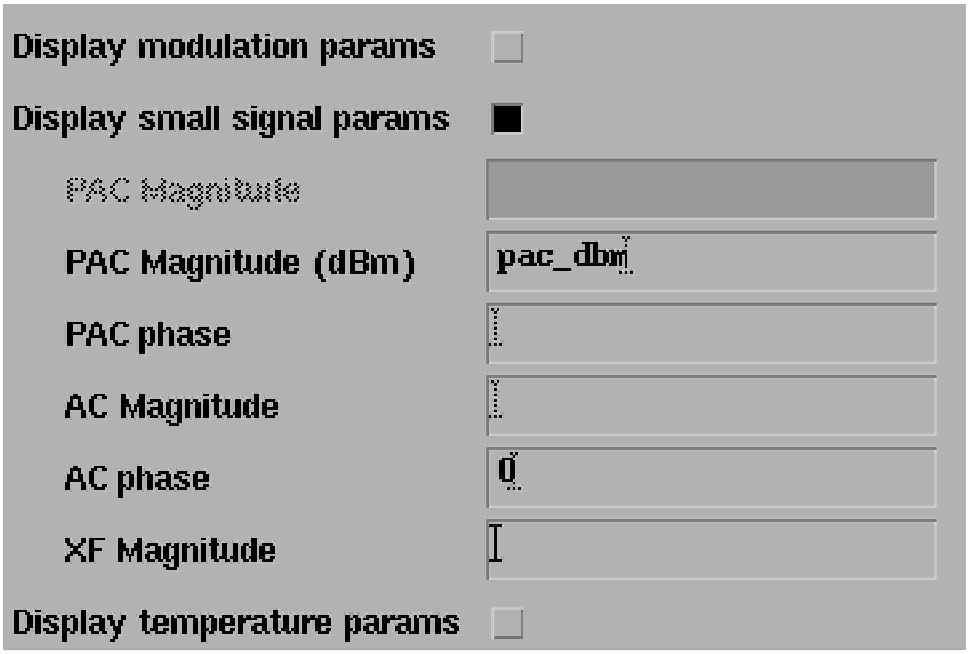


b. Highlight *Display small signal params*.

The small signal parameters section of the form opens up.

c. Type *pac_dbm* for *PAC Magnitude (dBm)*.

The completed form looks like this.



4. In the Schematic window, select *RF_OUT*.

The Edit Object Properties form changes to display data for *RF_OUT*.

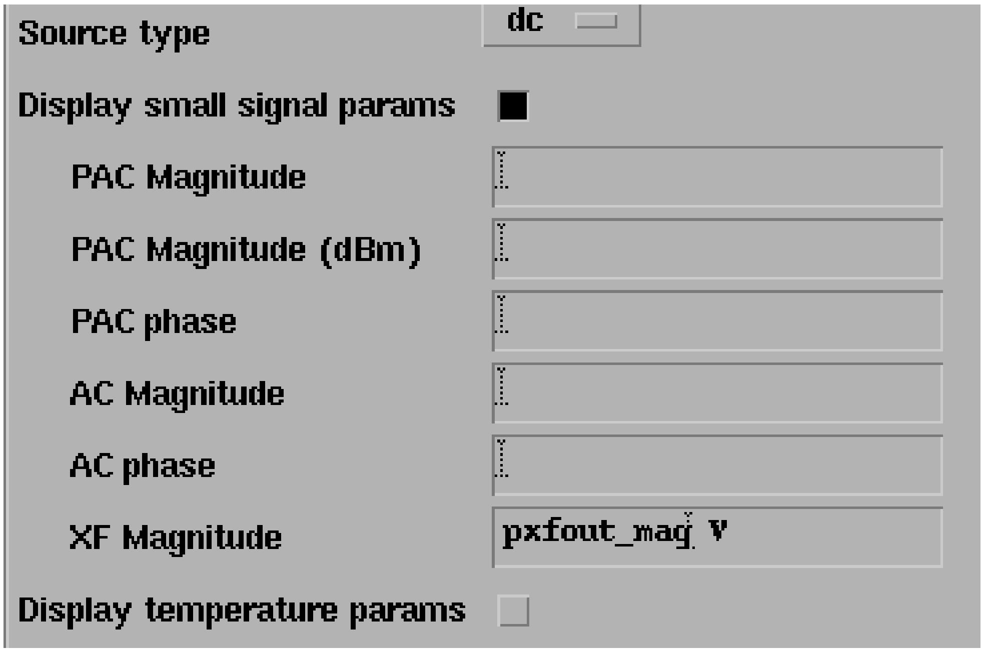
5. In the Edit Object Properties form, do the following and then click *OK*.

a. Highlight *Display small signal params*.

The small signal parameters section of the form opens up.

b. Type *pxfout_mag* for *XF Magnitude*.

The completed form looks like this.

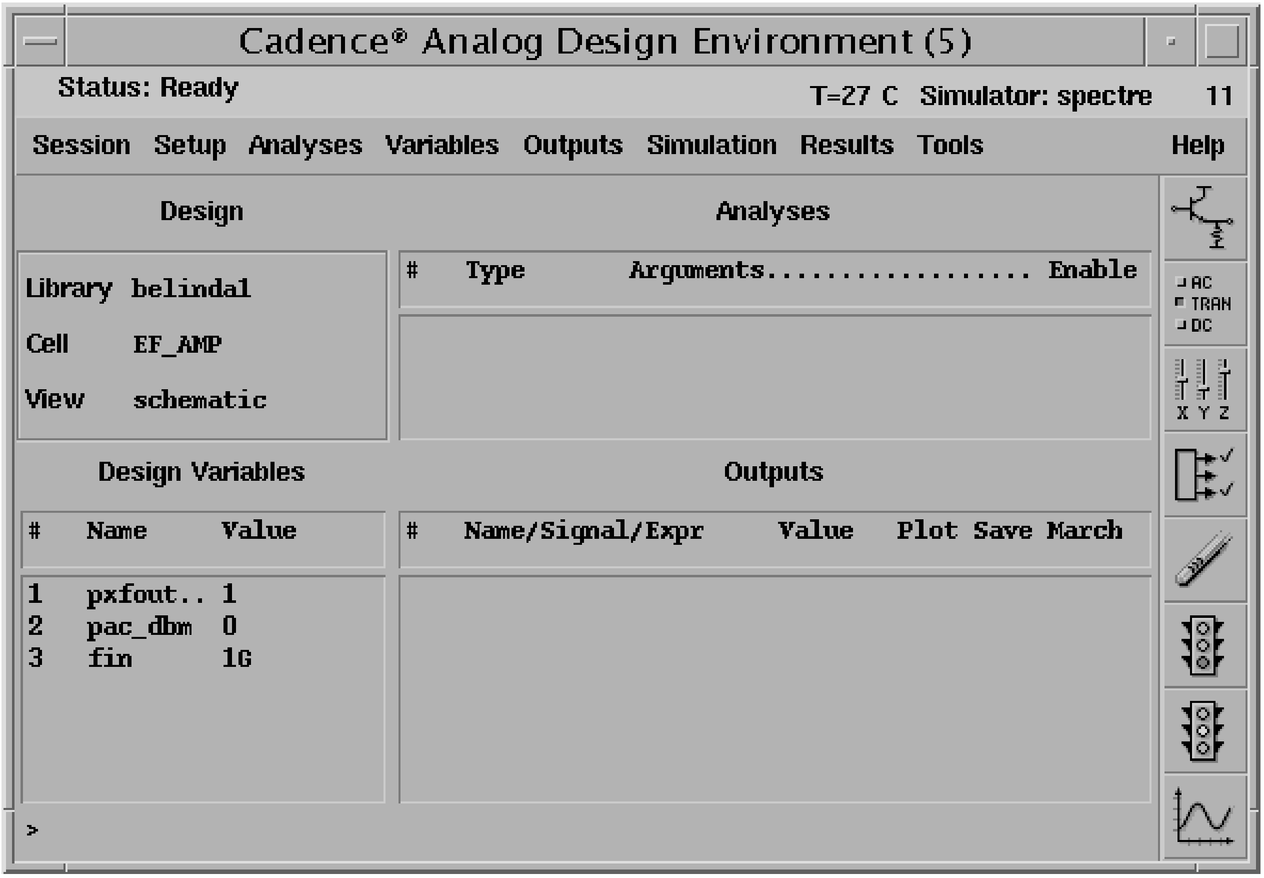


6. In the Schematic window, choose *Design - Check and Save*.

Opening the Simulation Window for the EF_AMP Circuit

1. In the EF_AMP Schematic window, choose *Tools – Analog Environment*.

The Simulation window opens.



You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing *EF_example* in the Choosing Design form.

Setting up the Model Libraries for the EF_AMP Circuit

1. In the Simulation window, choose *Setup - Model Libraries.*

   The Model Library Setup form appears.

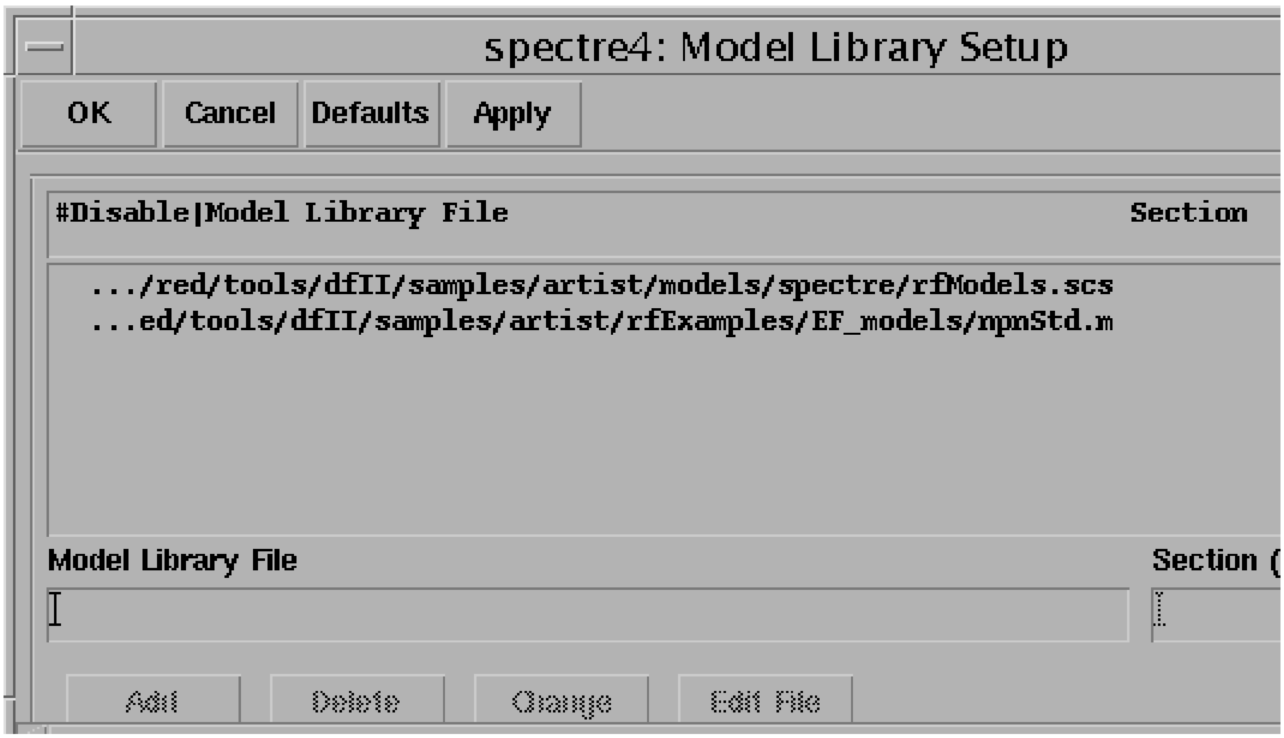
2. In the *Model Library File* field, type the following path, where *CDSHOME* is the installation directory for the Cadence software.

```
<CDSHOME>/tools/dfII/samples/artist/rfExamples/EF_models/npnStd.m
```

3. Click on *Add*.

4. In the *Model Library File* field, type the full path to the model file including the file name, `<CDSHOME>/tools/dfII/samples/artist/models/spectre/rfModels.scs`.
5. Click on *Add*.

   The Model Library Setup form looks like the following.

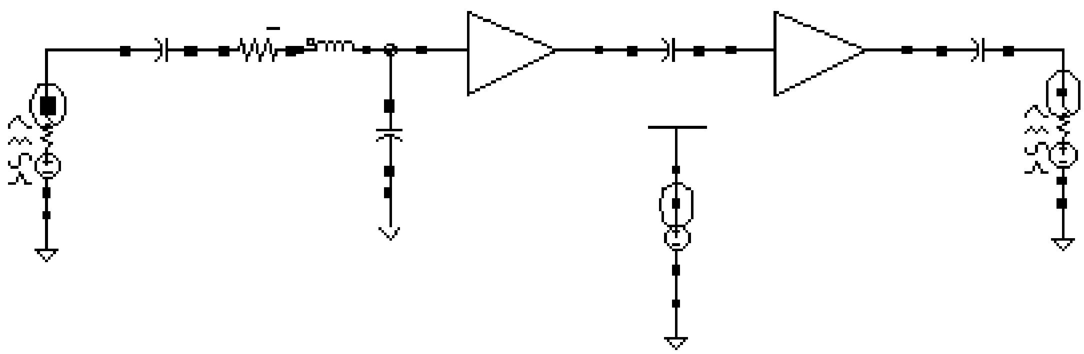


6. Click *OK*.

Selecting Outputs To Save

1. In the Simulation window, choose *Outputs – To Be Saved – Select on Schematic*.
2. In the Schematic window, click on the terminals that are circled in Figure 10-28.

Figure 10-28 Outputs to Save in the EF_AMP Schematic



After you click on a terminal, it is circled in the schematic. The selected outputs are also displayed in the Schematic window Outputs area as shown in Figure 10-29.

Figure 10-29 Outputs Area in Simulation Window

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	RF_OUT/PLUS		no	yes	no
2	VCC/PLUS		no	yes	no
3	PORT3/PLUS		no	yes	no

3. In the Simulation window, choose *Outputs-Save All*.

The Save Options form displays.

4. Set the following values

- ❑ *save* to *allpub*
- ❑ *currents* to *selected*
- ❑ *useprobes* to *yes*

5. Click *OK*.

Setting Up and Running the PSS, PAC Modulated and PXF Modulated Analyses

Setting up the PSS Analysis

1. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

2. In the Choosing Analyses form, highlight *pss*.

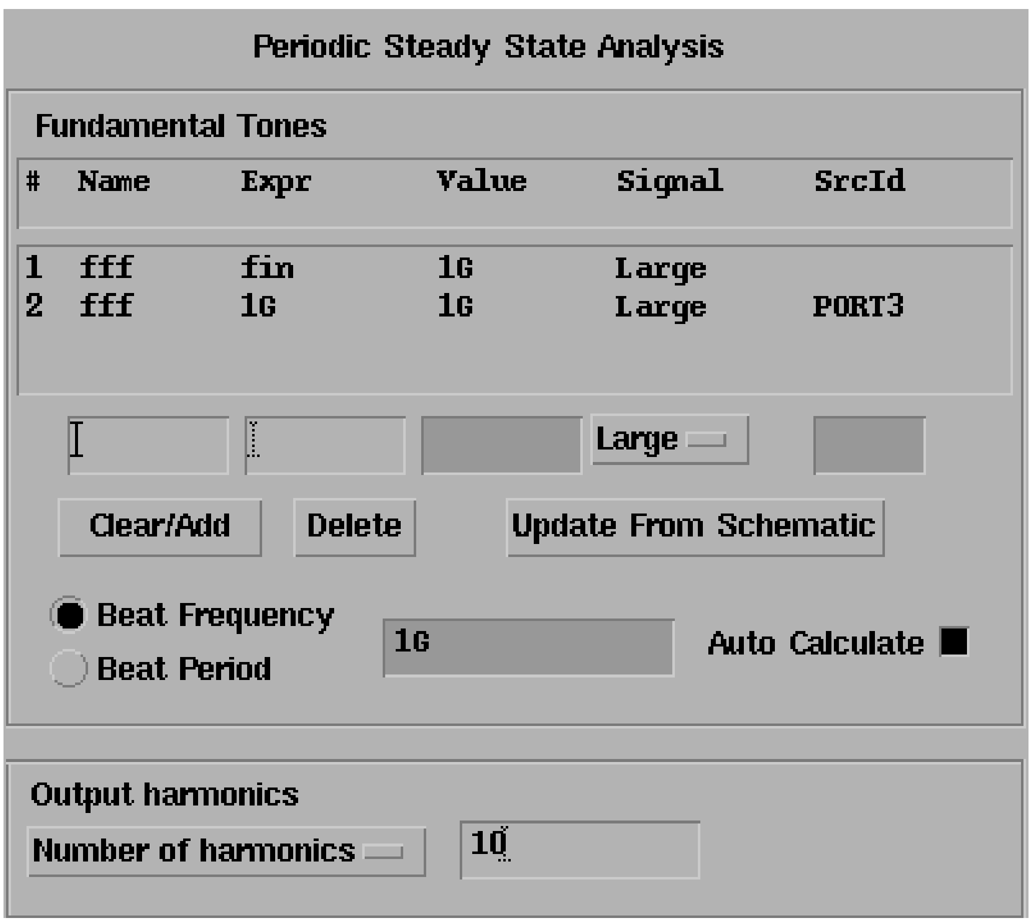
   The form changes to display options for PSS simulation.

3. In the Choosing Analyses form, do the following and then click *Apply*:

   a. Highlight the *Auto Calculate* button to automatically calculate and enter the value `1 G` in the *Beat Frequency* field.

   b. Choose *Number of harmonics* for *Output harmonics* and type `10` in the adjacent field.

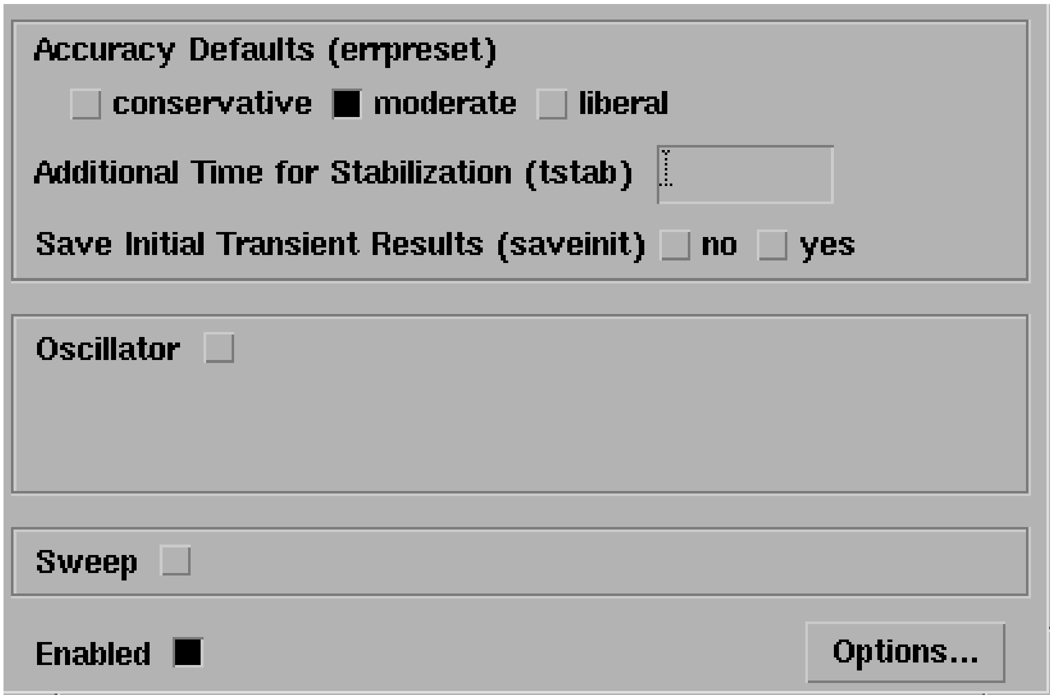
      The top of the PSS analysis form looks like this.



   c. Highlight *moderate* for Accuracy Defaults *(errpreset)*.

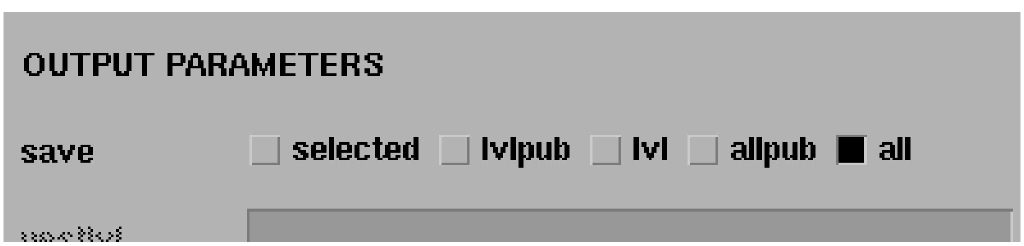
   d. Verify that *Enabled* is highlighted in the PSS Choosing Analysis form.

The bottom of the PSS Choosing Analysis form looks like this.



- **e.** Click the *Options* button to display the Options form for the PSS analysis.
- **f.** In the Options form, scroll to the *Output Parameters* section, highlight *all* for *save* and click *OK*.

  The *Output Parameters* section of the Options form looks like the following.



Setting up the PAC Modulated Analysis

1. In the Choosing Analyses form, highlight *pac*.

   The form changes to display options for PAC simulation.

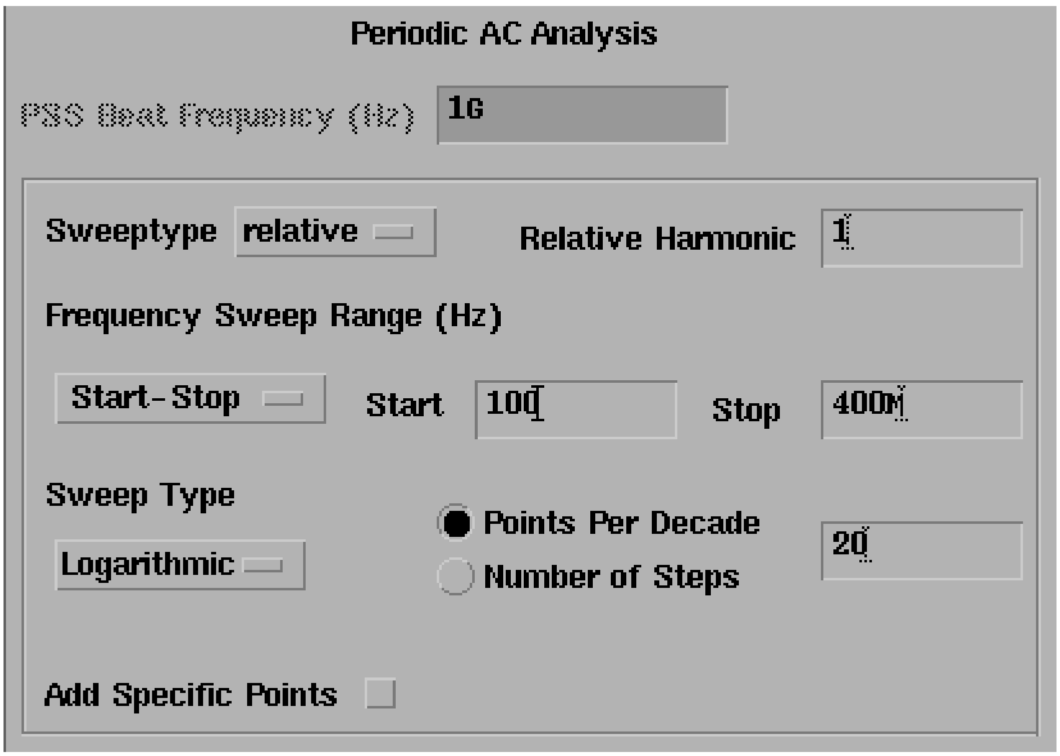
2. In the Choosing Analyses form, do the following and then click *Apply*:

a. For *Sweeptype*, select *Relative* in the cyclic field and type `1` in the *Relative Harmonic* field.

b. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `100` in the *Start* field and `400M` in the *Stop* field.

c. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `20` in the field.

The top of the PAC analysis form looks like this.

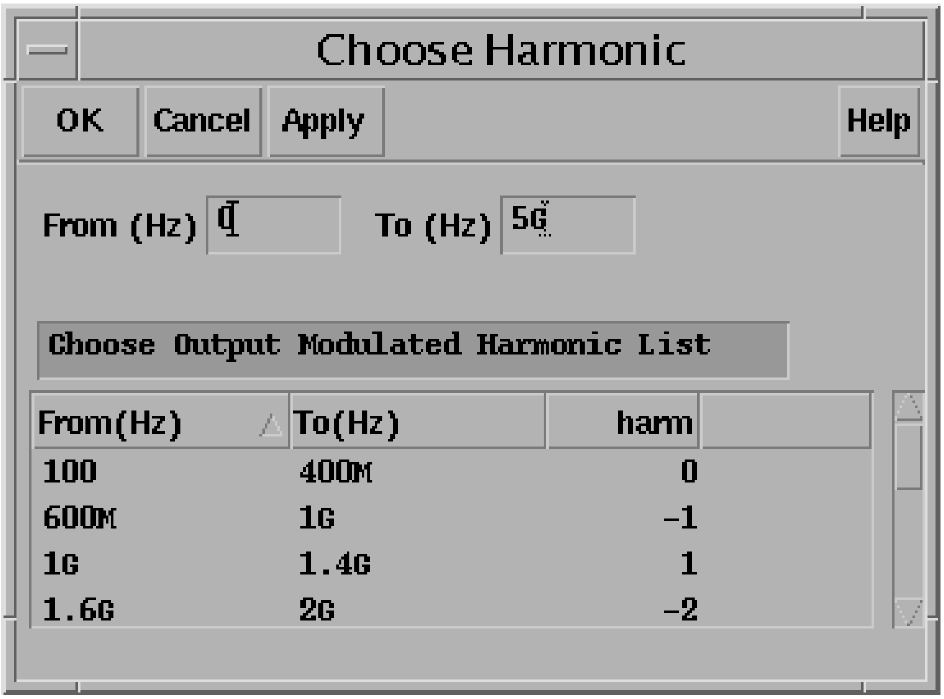


d. For *Sidebands*, select *Maximum sideband* in the cyclic field. Type `3` in the field.

e. Highlight *Modulated Analysis* to open the modulated analysis section of the PAC Choosing Analysis form.

f. For *Input Type* choose *SSB/AM/PM* in the cyclic field.

g. For *Output Modulated Harmonic List*, press *Choose* to display the Choose Harmonic form.

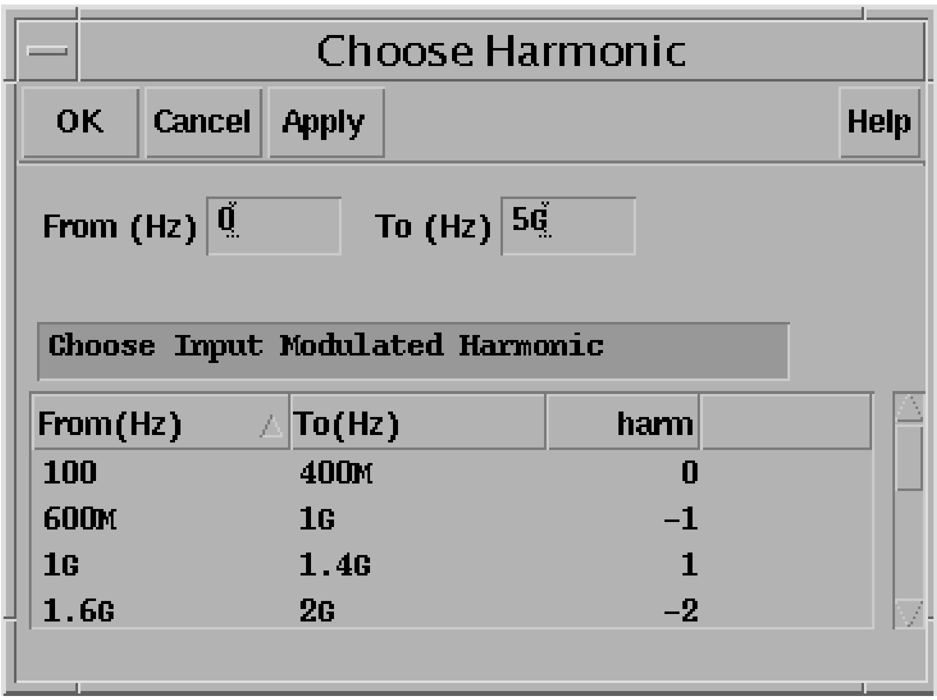


h. Scroll the list of harmonics and select harmonics with indexes 1, 2 and 3. Click to highlight the first harmonic. Press and hold down the *Control* key while you click to select harmonics 2 and 3. Then click OK.

1, 2, and 3 display in the *Output Modulated Harmonic List* field.

Note: You can also simply type the values, separated by spaces, in the *Output Modulated Harmonic List* field.

i. For *Input Modulated Harmonic*, press *Choose* to display the Choose Harmonic form.



j. Scroll the list of harmonics and select the harmonics with index of 1. Click to highlight the harmonic 1. Then click OK.

`1` displays in the *Input Modulated Harmonic* field.

Note: You can also simply type the value in the *Input Modulated Harmonic* field.

k. For *Modulated Input Source*, click *Select* and follow the prompt at the bottom of the Schematic window

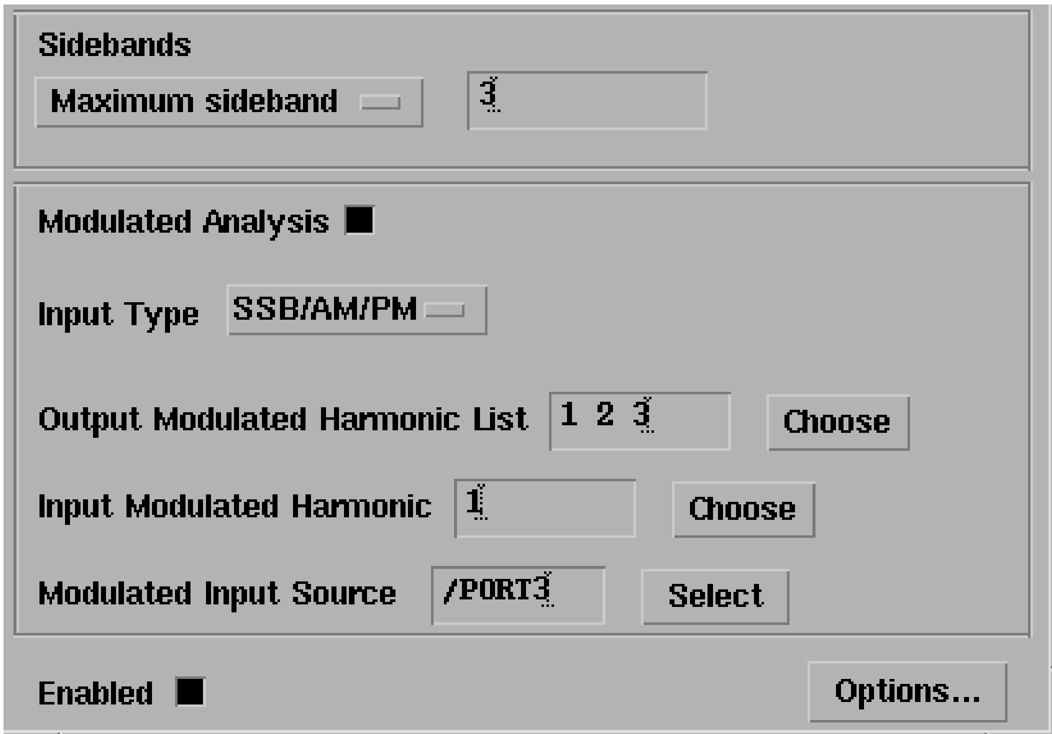
```
Select modulation sourcce instance...
```

and select the port on the left side of the schematic.

/PORT3 displays in the *Modulated Input Source* field.

l. Verify that *Enabled* is highlighted in the PAC Choosing Analysis form.

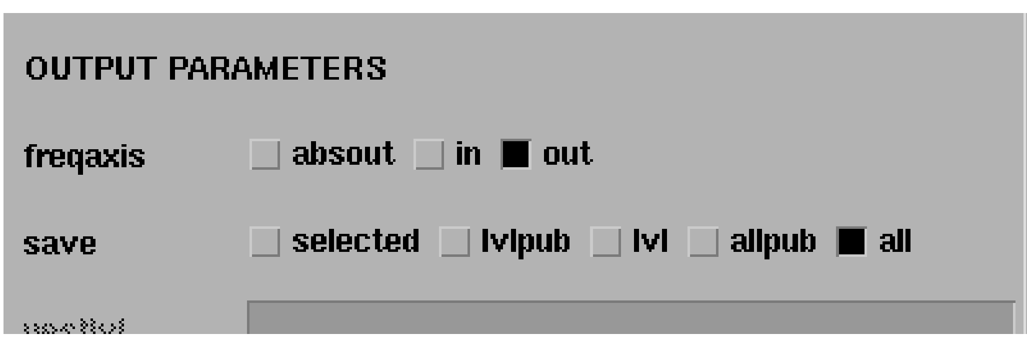
The bottom of the PAC analysis form looks like this.



m. Click the *Options* button to display the Options form for the PAC analysis.

n. In the *Output Parameters* section of the Options form, highlight, *out* for *freqaxis* and *all* for *save.* Then click *OK*.

The Output *Parameters* section of the Options form looks like the following.



Setting up the PXF Modulated Analysis

1. In the Choosing Analyses form, highlight *pxf*.

The form changes to display options for PXF analysis.

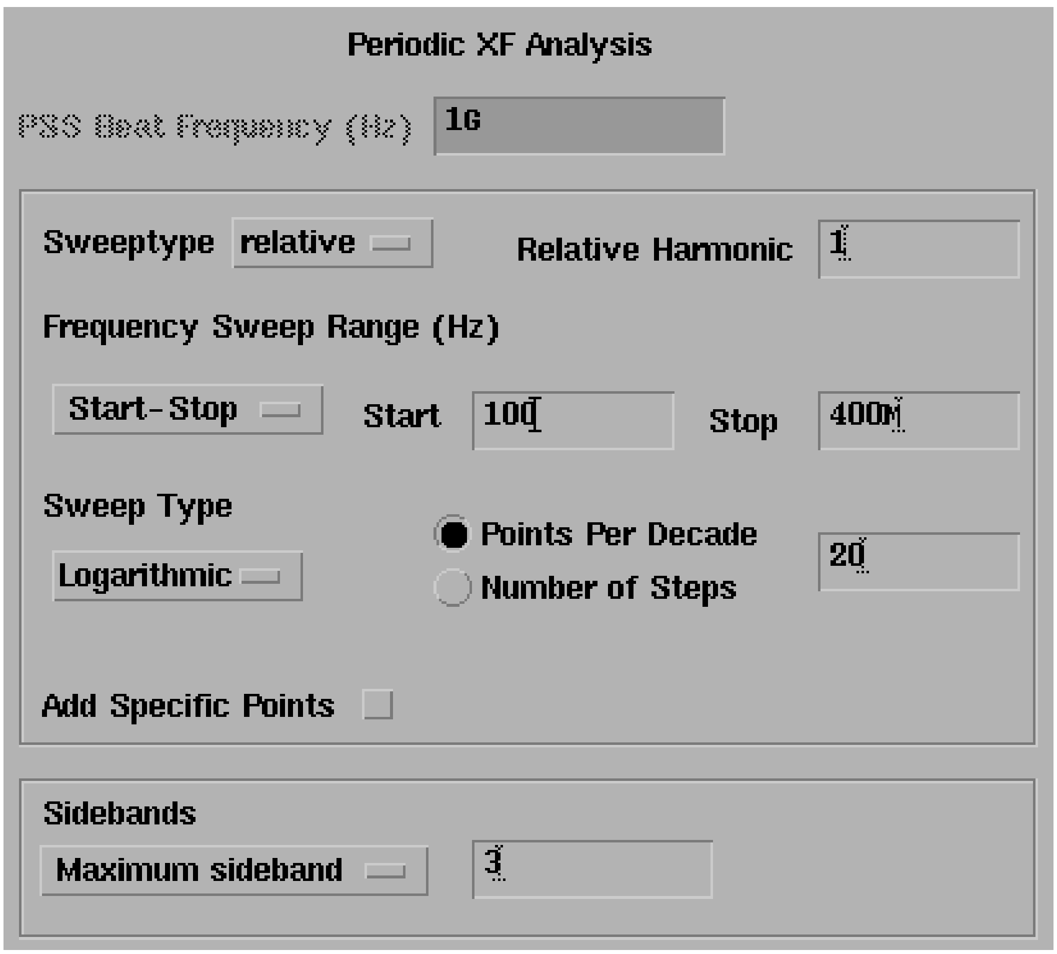
2. In the Choosing Analyses form, do the following and then click *Apply*:

a. For *Sweeptype*, select *Relative* in the cyclic field and type `1` in the *Relative Harmonic* field.

b. For *Frequency Sweep Range*, select *Start-Stop* in the cyclic field. Type `100` in the *Start* field and `400M` in the *Stop* field.

c. For *Sweep Type*, select *Logarithmic* in the cyclic field, highlight *Points Per Decade* and type `20` in the field.

The top of the PXF analysis form looks like this.



d. For *Sidebands*, select *Maximum sideband* in the cyclic field and type `3` in the field.

e. For *Output*, highlight *probe*. Click *Select* and follow the prompt at the bottom of the Schematic window

```
Select output probe instance...
```

and select the RF port on the right side of the schematic.

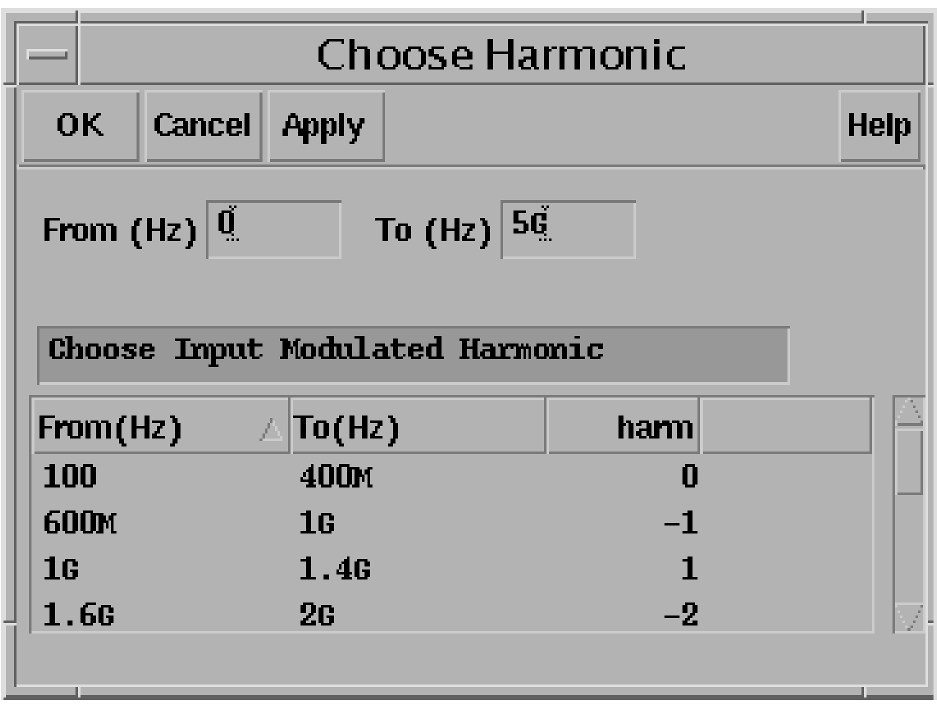
`/RF_OUT` displays in the *Output Probe Instance* field.

f. Verify that *Enabled* is highlighted in the PXF Choosing Analysis form.

g. Highlight *Modulated Analysis* to open the modulated analysis section of the PXF Choosing Analysis form.

h. For *Output Type* choose *SSB/AM/PM* in the cyclic field.

i. For *Input Modulated Harmonic List*, press *Choose* to display the Choose Harmonic form.

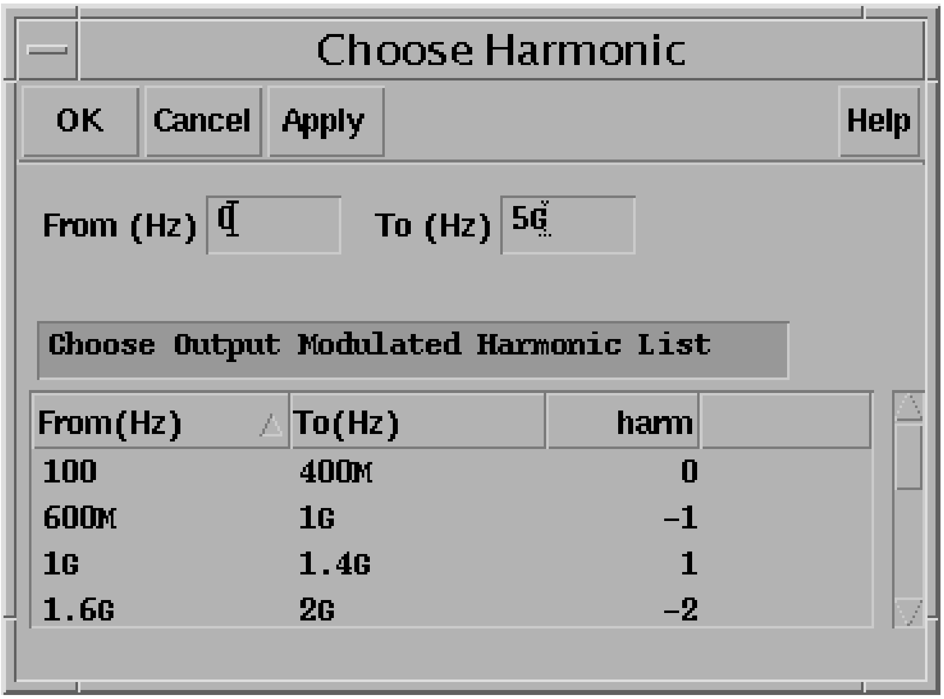


j. Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click OK.

`1` displays in the *Input Modulated Harmonic List* field.

Note: You can also simply type the value in the *Input Modulated Harmonic List* field.

k. For *Output Modulated Harmonic*, press *Choose* to display the Choose Harmonic form.

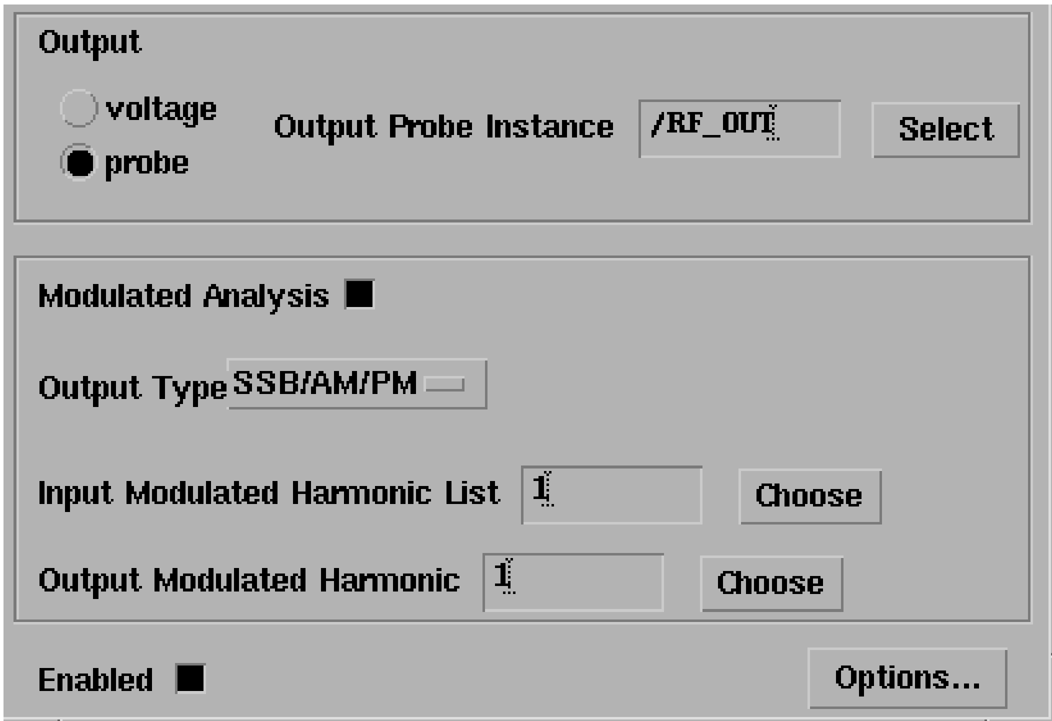


l. Scroll the list of harmonics and select the harmonic with index of 1. Click to highlight the harmonic 1. Then click OK.

1 displays in the *Output Modulated Harmonic* field.

Note: You can also simply type the value in the *Input Modulated Harmonic* field.

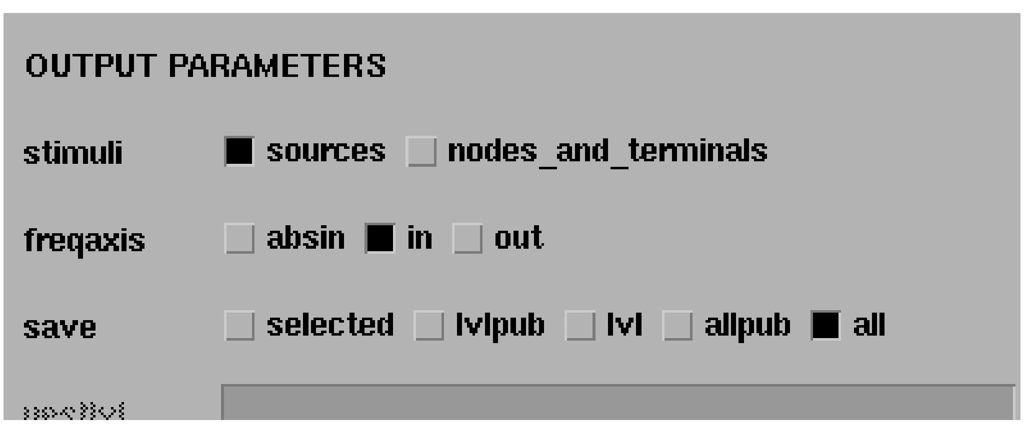
The bottom of the PXF analysis form looks like this.



m. Click the *Options* button to display the Options form for the PXF analysis.

n. In the *Output Parameters* section of the Options form, highlight, *sources* for *stimuli, in* for *freqaxis* and *all* for *save.* Then click *OK*.

The Output *Parameters* section of the Options form looks like the following.



3. Click OK in the Choosing Analyses form.

The Choosing Analyses form.closes.

Running the Simulations

1. To run the simulations, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting and Calculating PAC Modulated Results

➤ Choose *Results – Direct Plot – Main Form* in the Simulation window.

The Direct Plot form appears.

In the Direct Plot form, do the following:

1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pac modulated* for *Analysis*.
3. In the *Input* cyclic field, select *AM*.
4. In the *Modulated Input Harmonic* cyclic field, select *1G 1*. This is the only available choice.

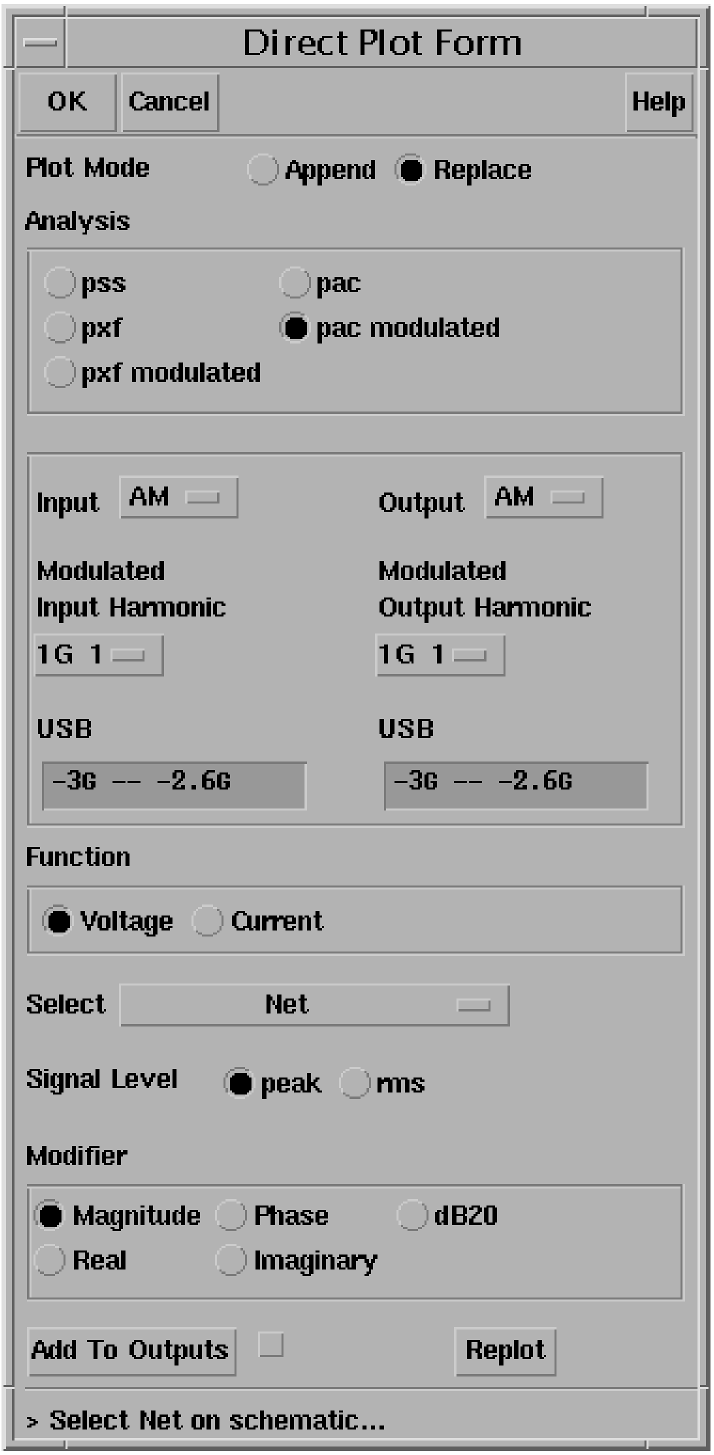
   The *USB* field displays `-3G -- -2.6G`.

5. In the *Output* cyclic field, select *AM*.
6. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `-3G -- -2.6G`.

7. Highlight *Voltage* for *Function*.
8. Highlight *peak* for *Signal Level*.
9. Highlight *dB20* for *Modifier*.

The Direct Plot form displays as follows.

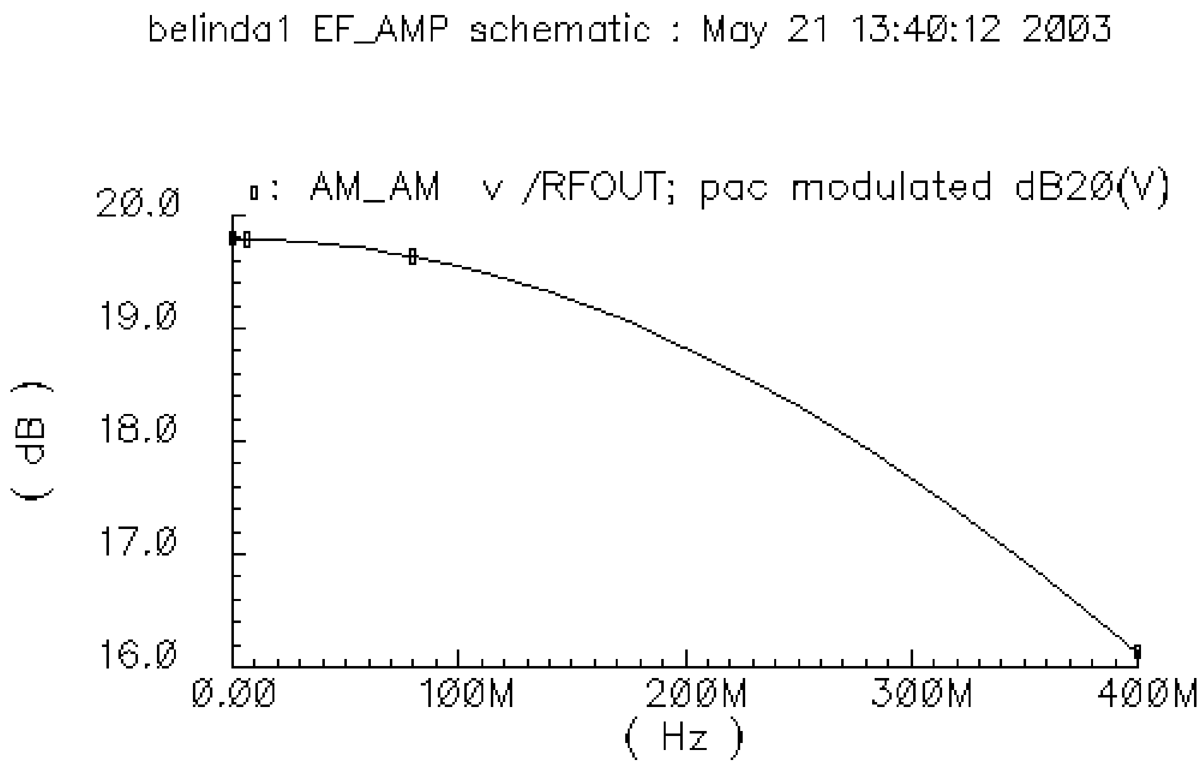


10. Select *Net* in the *Select* cyclic field. Follow the prompt at the bottom of the form

`Select Net on schematic`

Select the net labeled *RFOUT* on the right side of the schematic.

The waveform looks as follows.

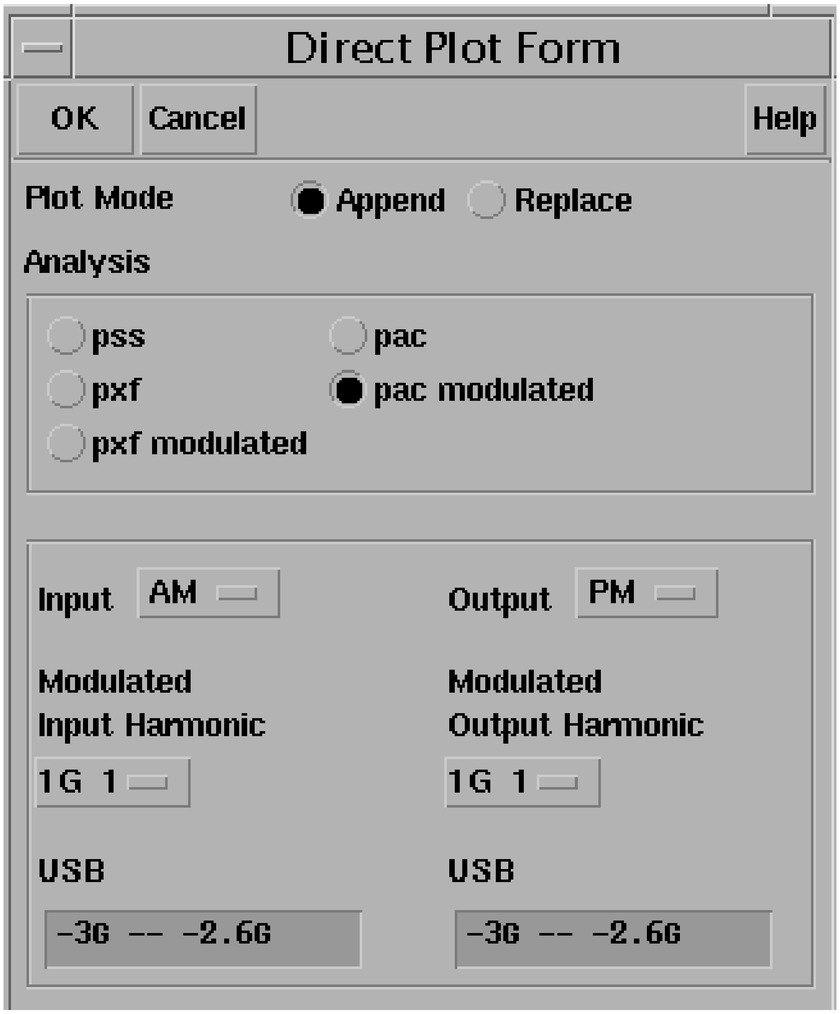


To add AM/PM to the plot, make the following changes in the Direct Plot form.

1. Change Plot Mode to Append.
2. Change the *Output* cyclic field to *PM*.
3. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

   The *USB* field displays `-3G -- -2.6G`.

The top of the Direct Plot form changes as follows.

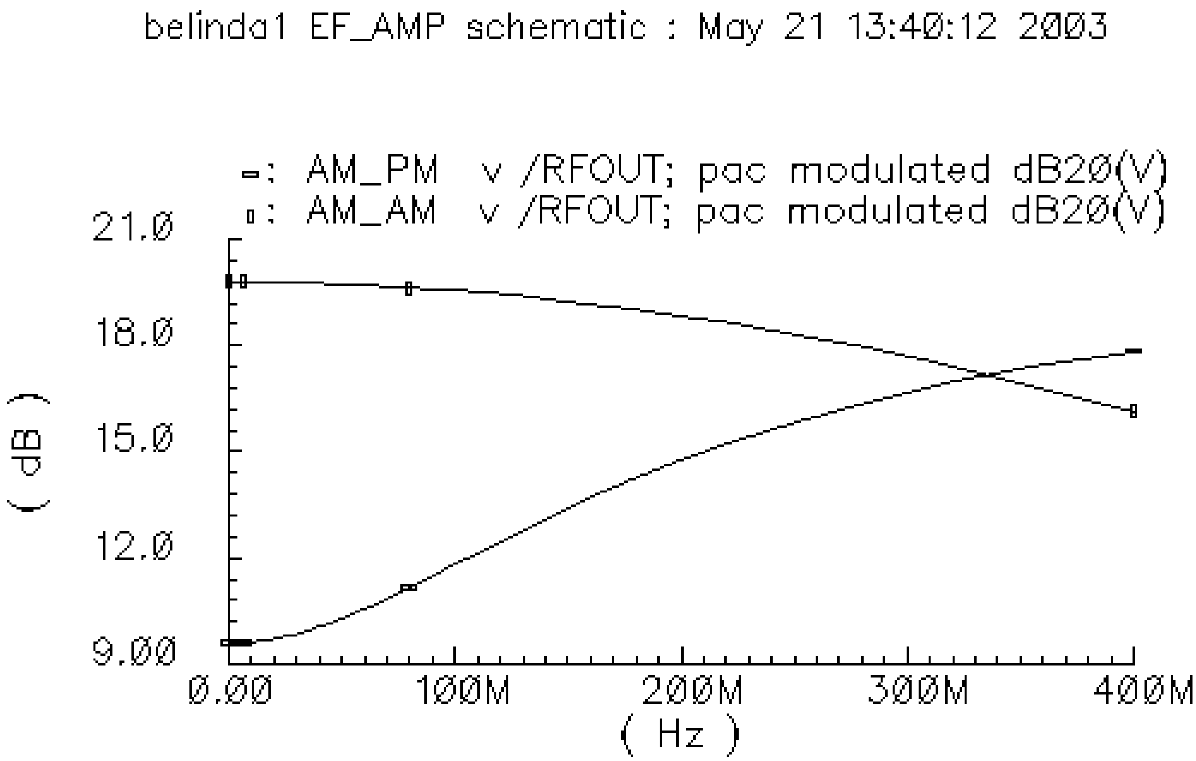


4. Follow the prompt at the bottom of the form

`Select Net on schematic`

Select the net labeled *RFOUT* again.

The waveform looks as follows.



Plotting and Calculating PXF Modulated Results

➤ If necessary, choose *Results – Direct Plot – Main Form* in the Simulation window.

The Direct Plot form appears.

In the Direct Plot form, do the following:

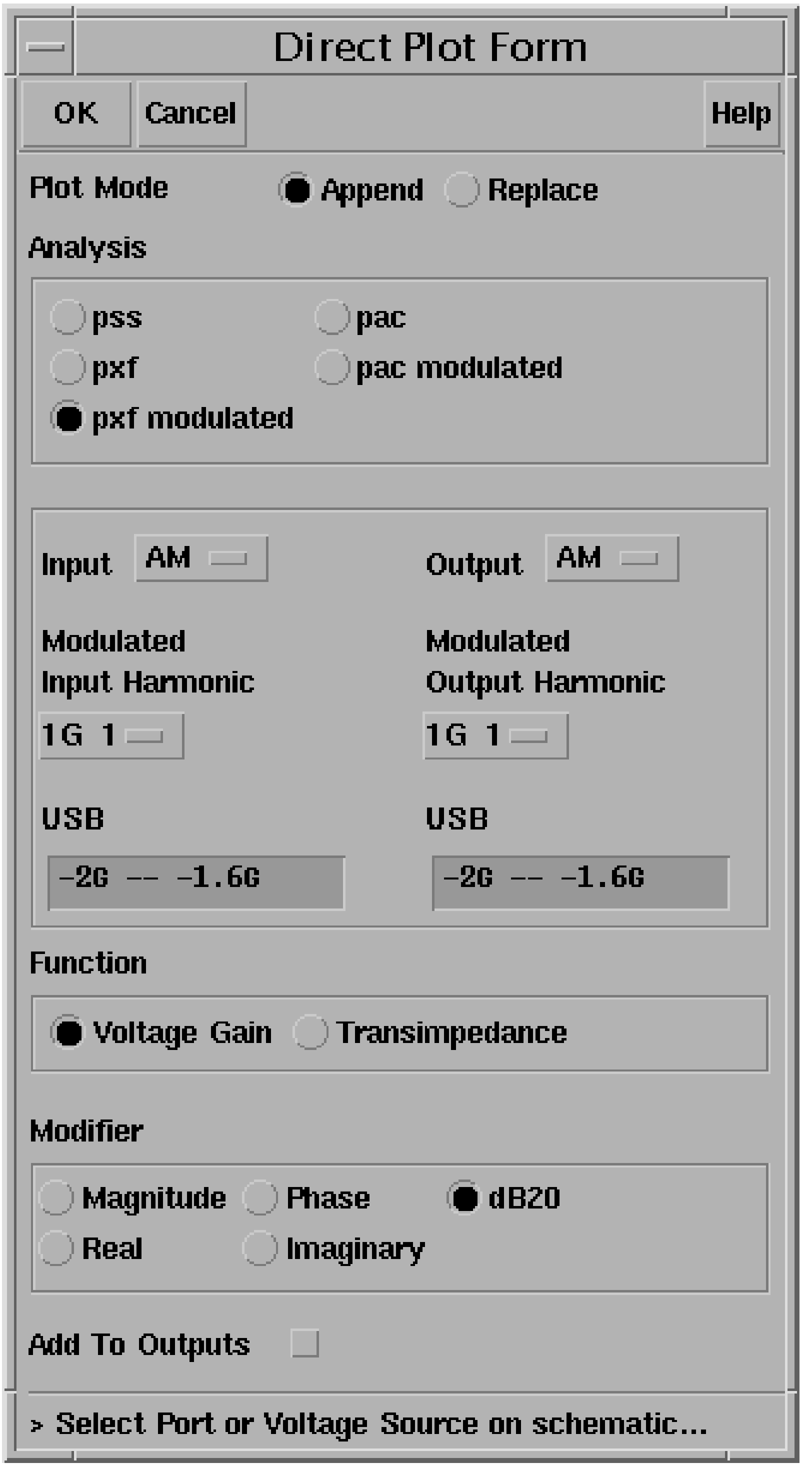
1. Highlight *Replace* for *Plot Mode*.
2. Highlight *pxf modulated* for *Analysis*.
3. In the *Input* cyclic field, select *AM*.
4. In the *Modulated Input Harmonic* cyclic field, select *1G 1*. This is the only available choice.

   The *USB* field displays `-2G -- -1.6G`.
5. In the *Output* cyclic field, select *AM*.
6. In the *Modulated Output Harmonic* cyclic field, select *1G 1*.

The *USB* field displays `-2G -- -1.6G`.

7. Highlight *Voltage Gain* for *Function*.
8. Highlight peak for Signal Level.
9. Highlight *dB20* for *Modifier*.

The Direct Plot form displays as follows.

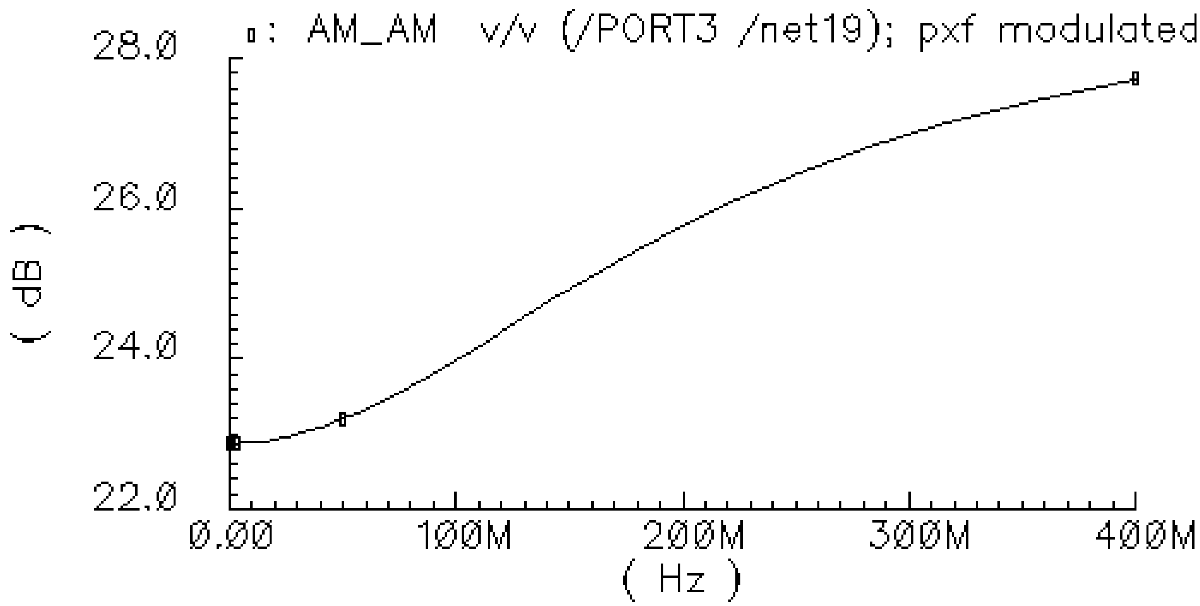


10. Follow the prompt at the bottom of the form

```
Select Port or voltage source on schematic
```

Select the input port labeled *PORT3* on the left side of the schematic.

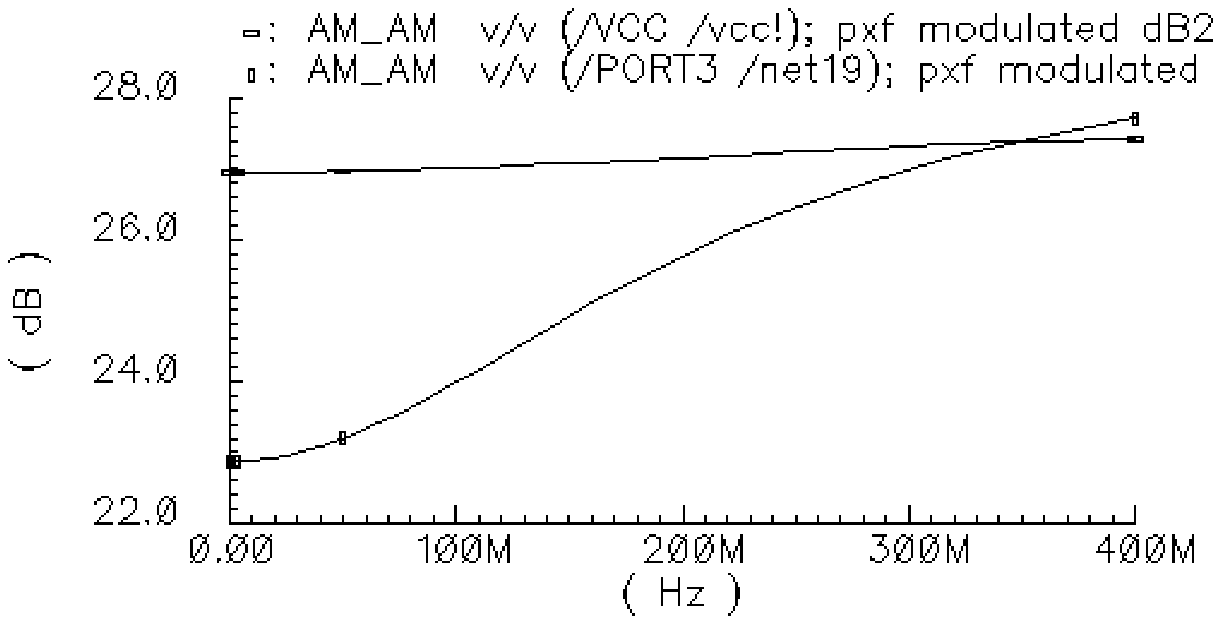
The waveform looks as follows.



To add the VCC voltage source to the plot,

1. Change Plot Mode to Append.
2. Select the voltage source labeled *VCC* in the lower center of the schematic.

The VCC voltage source is added to the plot.



This plot compares two different voltage sources from the schematic.

11

Modeling Spiral Inductors, Bonding Pads, and Transformers

The procedures for modeling spiral inductors, bonding pads and planar transformers are similar in many ways. The on-chip passive component modelers model spiral inductors, bonding pads, and planar transformers for RF applications. The modelers all have an extraction engine based on EM (Electromagnetics) techniques. The analog design environment calls this engine using a command interface.

See “Modeling Spiral Inductors” on page 843

See “Modeling Bonding Pads” on page 875

See “Modeling Planar Transformers” on page 892

Modeling Spiral Inductors

There are two use models for the Spiral Inductor modeler.

- One is to use the modeler in the schematic flow. You specify the input parameters for the modeler in the CDF form, and model extraction is executed at netlisting time.
- The other is using the RF sub-menu option to access the modeler from the Simulation window. You launch the modeler directly using the Tools menu in the Simulation window. The inductor model is extracted and written to the specific model directory, and the electric parameters of the components are calculated and displayed on the UI form. You can include the extracted model in Spectre/SpectreRF simulations, with multiple instances if necessary, and therefore avoid a long extraction time at netlisting.

The solver for the on-chip passive modeler employs a PEEC (Partial Element Equivalent Circuit) algorithm to generate macromodels for the spiral components. Electro-static and magneto-static EM solvers are called separately to extract the capacitive and inductive parameters of the structure.

The capacitive parameter extraction solver uses the BEM (Boundary Element Method), which computes the lossy substrate using a complex Green's function, and the solver is SVD (Singular Value Decomposition) accelerated.

The inductance solver is based on Fasthenry. It considers skin-depth effects, and models the substrate effect by treating the lossy layers as lossy conductor planes. The meshes for inductive and capacitive extractions are inter-correlated. There is one more segment for capacitive extraction than there is for the inductive mesh. There is also a half segment overlap between the two meshes. Because of the conductor and substrate loss, the extracted parameters are formulated as RLCG matrixes.

With the extracted parameters, a multi-PI equivalent circuit is created. Depending on the bandwidth, the modeler generates either a narrow-bandwidth or wide-bandwidth model.

- The narrow-bandwidth model uses the parasitic parameters extracted at the working frequency and builds a fully-coupled multi-PI equivalent circuit. Because the capacitive and inductive parameters are functions of frequency, this model is accurate only in the vicinity of the working frequency specified.
- The wide-bandwidth model extracts the capacitive and inductive parameters over a wide frequency range, and generates a model as an nport. In the nport, a ratio of rational polynomials represents the frequency dependent behavior. Unlike the narrow-bandwidth model, the wide-bandwidth model is accurate over the whole frequency range up to the specified maximum frequency of interest. However, the model extraction takes longer, and in some cases, the extracted model might not be passive and might cause simulation failure. If the simulation fails, the solver automatically switches to a narrow-bandwidth model.

The remainder of the spiral inductor section covers the following topics:

- The Process File Preparation section explains how to setup the process file and gives an example CMOS process file.
- The Spiral Inductor Simulation in the Schematic Flow section explains details about the spiral inductor symbol and CDF parameters.
- The Spiral Inductor Modeling Using the RF Sub-menu section tells you how to model a spiral inductor using the RF sub-menu and explains the electrical parameters.
- The Equivalent Circuit Models section explains the equivalent circuit models and discusses issues you need to be aware of.
- The Example section takes you step-by-step through a simulation that uses the spiral inductor modeling capability.

Process File Preparation

Before you use the passive modeler to model spiral inductors, bonding pads, or planar transformers, you must create a process file that contains the layer information for both the metal and the substrate.

You must hand edit the process file to conform exactly to the required format.

File Format

The process file is in the following format

```
processType type
numLayers # of layers
layerInfo 0
...
layerInfo # of layers -1
```

In the above format, the first line,

```
processType type
```

is currently used only for documentation.

The second line,

```
numLayers # of layers
```

specifies the number of layers, including both the dielectric and metal layers.

The lines that follow, are `N1` blocks of layer information; where each layer information block has a specific format you must follow.

In the process file, all separate structures are modeled so as to account for the effects of surrounding layers. For example, a two-metal layer structure immersed in the substrate is modeled as five different layers:

1. The dielectric layer underneath the bottom metal layer.
2. The bottom metal layer.
3. The dielectric layer between the two metal layers.
4. The top metal layer.
5. The layer above the top metal layer.

Layer Information Format

You can describe the following three types of layers:

- Ground Plane – The entire layer is a piece of metal.
- Dielectric Layer– The entire layer is a dielectric of uniform dielectric constant.
- Metal Layer – Part of the structure is metal surrounded by dielectric.

The formats for these three layer types are shown below:

Ground Plane Format

```
layer     0
    name            tab
    type            ground
    thickness       0.1e-6
    epsr            1
    rho             1.7857e-8
```

Dielectric Layer Format

```
layer     1
    name            substrate
    type            dielectric
    thickness       300e-6
    epsr            11.9
    rho             0.1
```

Metal Layer Format

```
layer   3
    name            metal1
    type            metal
    thickness       1e-6
    epsr            4
    rho             1e10
    rhom            3e-8
```

For the metal layer, *rho* is the resistivity of the dielectric surrounding the metal structure, and *rhom* is the resistivity of the metals. All the units must be in MKS. For example, express thickness is in meters and resistivity is in ohm meters.

Sample CMOS Process File

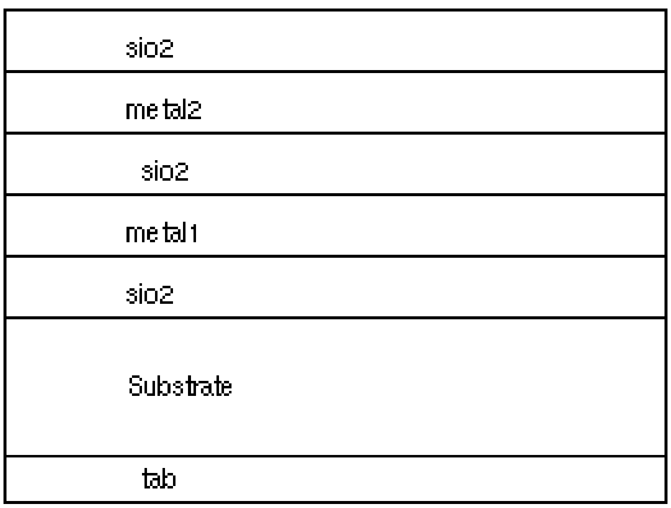
The following example shows a process file for the sample CMOS process shown in Figure 11-2 on page 848.

Figure 11-1 CMOS Process File

```
*Sample process file. All layers are stacked from bottom up
processType     CMOS
numLayers       7
layer     0
    name            tab
    type            ground
    thickness       0.1e-6
    epsr            1
    rho             1.7857e-8
layer     1
    name            substrate
    type            dielectric
    thickness       300e-6
    epsr            11.9
    rho             0.1
layer     2
    name            sio2
    type            dielectric
    thickness       1.5e-6
    epsr            4.0
    rho             1e10
layer     3
    name            metal1
    type            metal
    thickness       1e-6
    epsr            4
    rho             1e10
    rhom            3e-8
layer     4
    name            sio2
    type            dielectric
    thickness       1.5e-6
    epsr            4
    rho             1e10
layer     5
```

```
    name            metal2
    type            metal
    thickness       1e-6
    epsr            4
    rho             1e10
    rhom            3e-8
layer    6
    name            sio2
    type            dielectric
    thickness       43.5e-6
    epsr            4
    rho             1e10
```

Figure 11-2 Sample CMOS Process



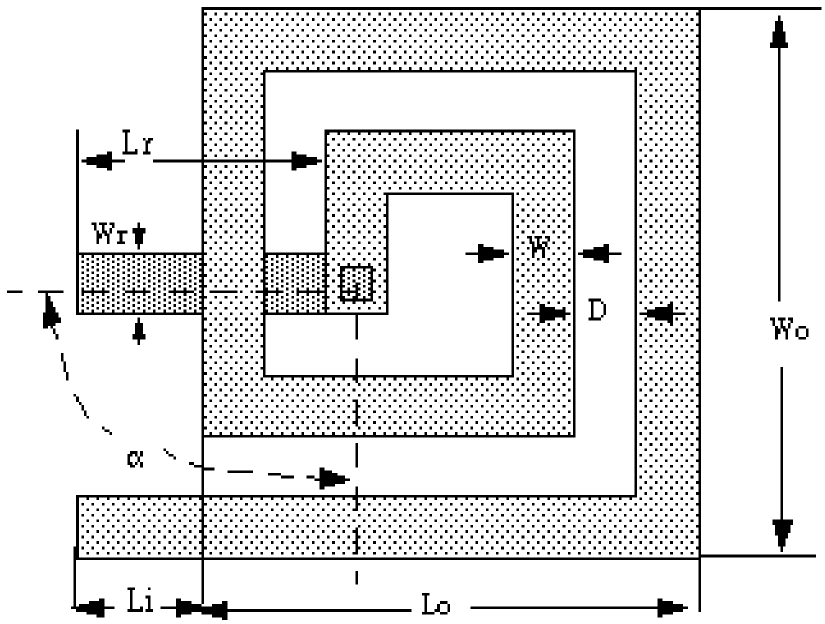
Spiral Inductor Simulation in the Schematic Flow

The spiral inductors, transformers, and bonding pads are stored in the *passiveLib* library.

The name of the spiral inductor component is *spiralInd*.

Figure 11-3 on page 849 shows the top view of a clockwise polarized 2-turn rectangular spiral inductor and displays the parameters used to describe it.

Figure 11-3 Top View of Rectangular Spiral Inductor



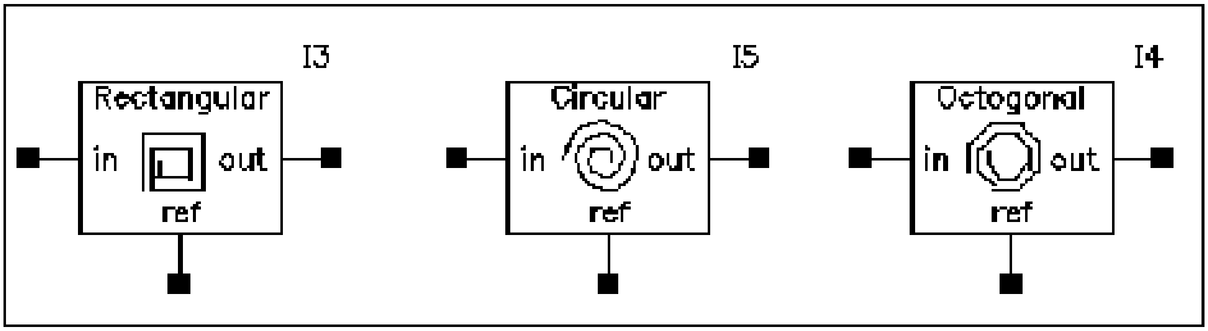
In Figure 11-3,

- Wo is the outer width
- Lo is the outer length
- Li is the input path length
- W is the metal width
- D is the turn spacing
- Lr is the return path length
- Wr is the return path width
- α is the angle between the return path and the innermost segment measured in degrees

Symbols and CDF Parameters for Spiral Inductors

The spiral inductor symbol is a 3-port block with the actual spiral configuration displayed. Figure 11-4 shows the symbols for rectangular, circular and octagonal spiral inductors.

Figure 11-4 Symbols for Spiral Inductors: Rectangular, Circular and Octagonal



For the symbols shown in Figure 11-4,

- The input terminal, *in*, represents the outer input terminal of the spiral inductor.
- The output terminal, *out*, represents the return path.
- The *ref* terminal represents the ground tab or the substrate bulk node.

➤ In the Schematic window, choose *Add—Instance* to display the Add Instance form.

In the Add Instance form, at the top

- ❑ `passiveLib` displays in the *Library* field
- ❑ `spiralInd` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field

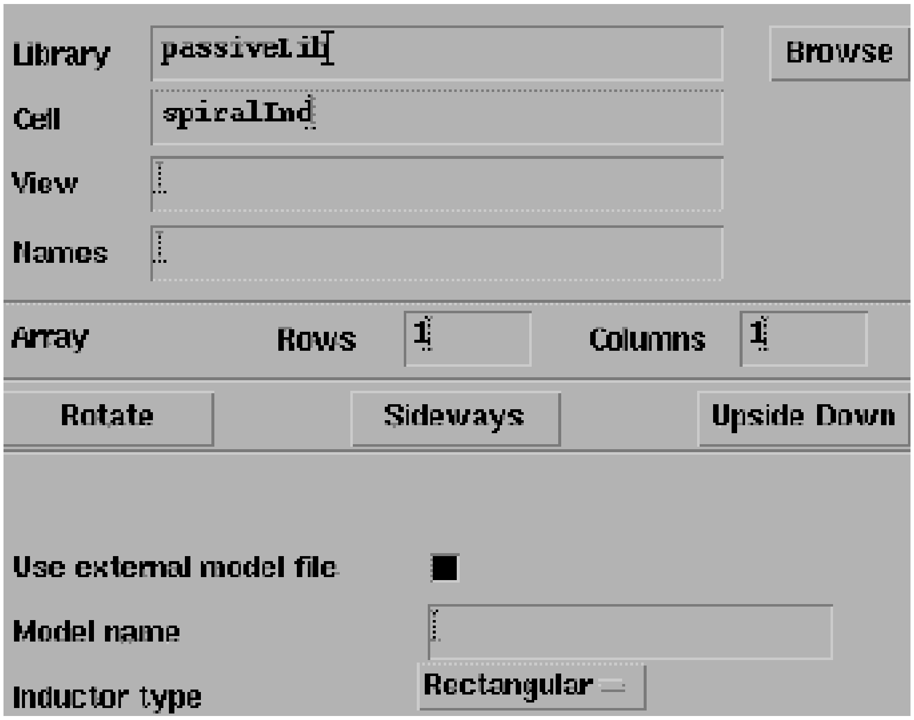
The CDF parameters for the spiral inductor element appear at the bottom of the form.

CDF Form Using an External Model File to Create a Spiral Inductor

- ***Use external model file***: This button lets you use an externally-generated spiral-inductor model file directly in a simulation. When you highlight the *Use external model file* button, the CDF parameters in the form change to look like the CDF form in Figure 11-5 on page 851 .
- ***Model name***: Enter a model file name in the field.

Note: The *Inductor type* parameter cyclic field changes the appearance of the symbol in the schematic but it does not affect the simulation results.

Figure 11-5 Use External Model File

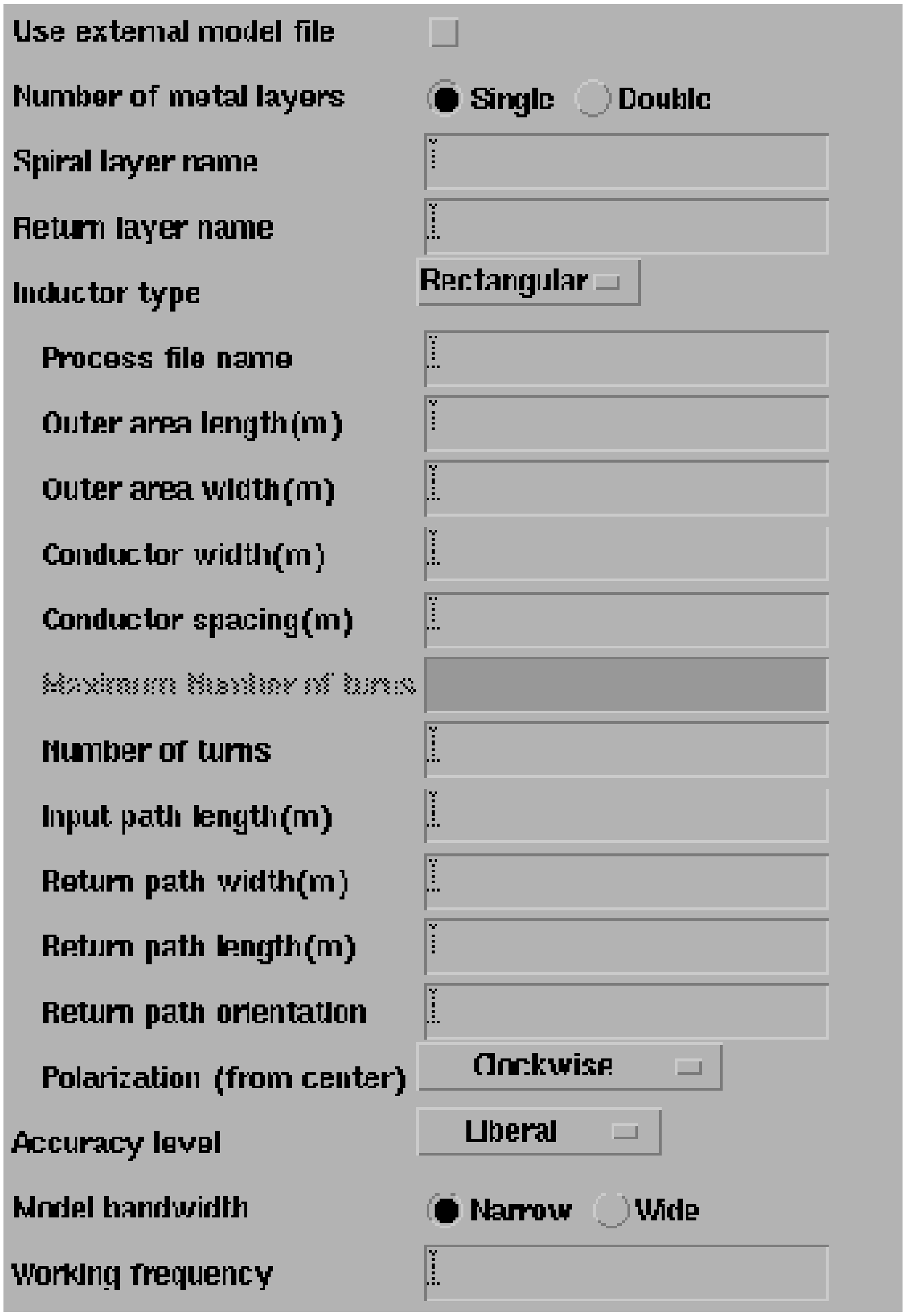


CDF Form for a Rectangular Spiral Inductor

The lower portion of the Add Instance form showing the CDF parameters for a single layer rectangular spiral inductor is shown in Figure 11-6 on page 852.

- ***Use external model file***: This button lets you use an externally-generated spiral-inductor model file directly in a simulation. When you highlight the *Use external model file* button, the CDF form changes to look like the CDF form in Figure 11-5. In this case, the *Inductor type* parameter only changes the appearance of the symbol in the schematic and does not affect the simulation results.
- ***Number of Metal Layers***: Possible choices are *Single* or *Double* for Single Spiral layer and Double Spiral layer inductors. For Double Layer inductors, the two Spiral Layers can have a *Connection type* of *Parallel* or *Series*.
- ***Spiral Layer Names***: The names of the layers the spirals sit in that you specify in the process file (*Spiral layer #1 name* and *Spiral layer #2 name*).
- ***Return layer name***: The name of the layer the return path sits in. For double-layer series inductors, this entry doesn't appear because the bottom spiral layer serves as the return path.

Figure 11-6 CDF for a Single Layer Rectangular Spiral Inductor



- ***Inductor Type***: The type of the spiral inductor. There are three inductor types: *Rectangular*, *Circular* and *Octagonal*. Changing this option also can change the appearance of the CDF.

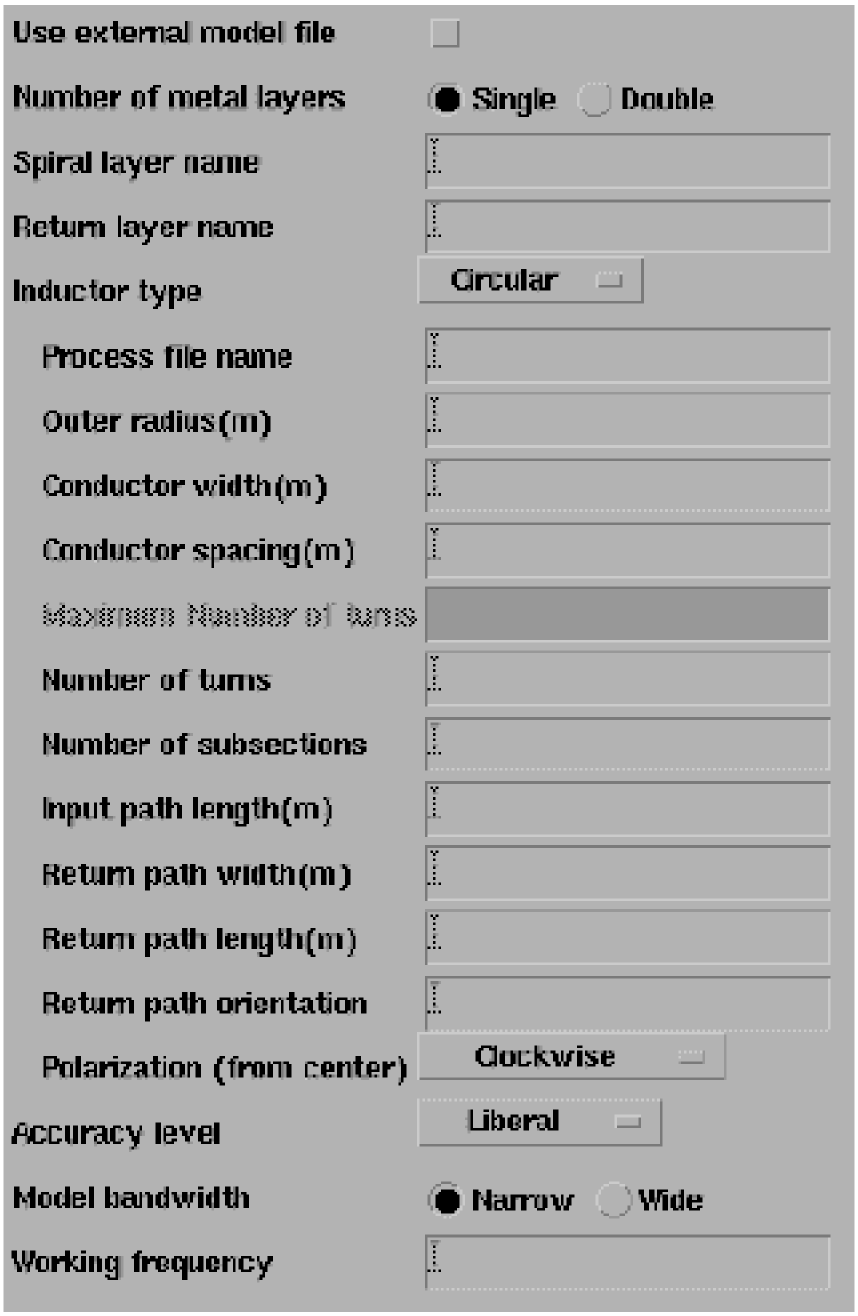
- ***Process file name***: The name of the process file. You must specify the full path.
- ***Outer area length***: For rectangular inductors, the length of the outer dimension of the spiral.

  For circular and octagonal inductors, specify the **Outer radius**. This is the distance from the center of the inductor to the outside edge of the spiral.
- ***Outer area width***: For rectangular inductors, the width of the outer dimension of the spiral.
- ***Conductor width***: The width of the spiral.
- ***Conductor spacing***: The gap between two neighboring turns of the spiral.
- ***Number of turns***: The number of spiral turns. The number can be fractional.
- ***Input path length***: The length of metal extending beyond the outermost segment of the spiral.
- ***Return path width***: The width of the return path
- ***Return path length***: The length of the return path
- ***Return path orientation***: The angle between the return path and the innermost segment of the spiral, measured in degrees.
- ***Polarization***: Possible choices are *Clockwise* or *Counterclockwise*, looking from the top down
- ***Accuracy level***: Determines the mesh density in capacitance and inductance extraction
- ***Model bandwidth***: Possible choices are *Narrow* and *Wide* Model bandwidth
- ***Working frequency***: The actual working frequency of the spiral. This value is used in the extraction of parasitics parameters.
- ***Maximum frequency***: The maximum frequency at which parasitic parameters are extracted for wide bandwidth models.

CDF Form for the Circular Spiral Inductor

With the rectangular spiral inductor (see Figure 11-6 on page 852), you specify the *Outer area length (m)* and *Outer area width (m)*. For a circular spiral (see Figure 11-7 on page 854), you specify the *Outer radius (m)* of the spiral and the Number *of subsections* in the polygon approximation of the spiral. The lower portion of the Add Instance form showing the CDF parameters for a single layer circular spiral inductor is shown in Figure 11-7 on page 854.

Figure 11-7 CDF Form for a Circular Spiral Inductor

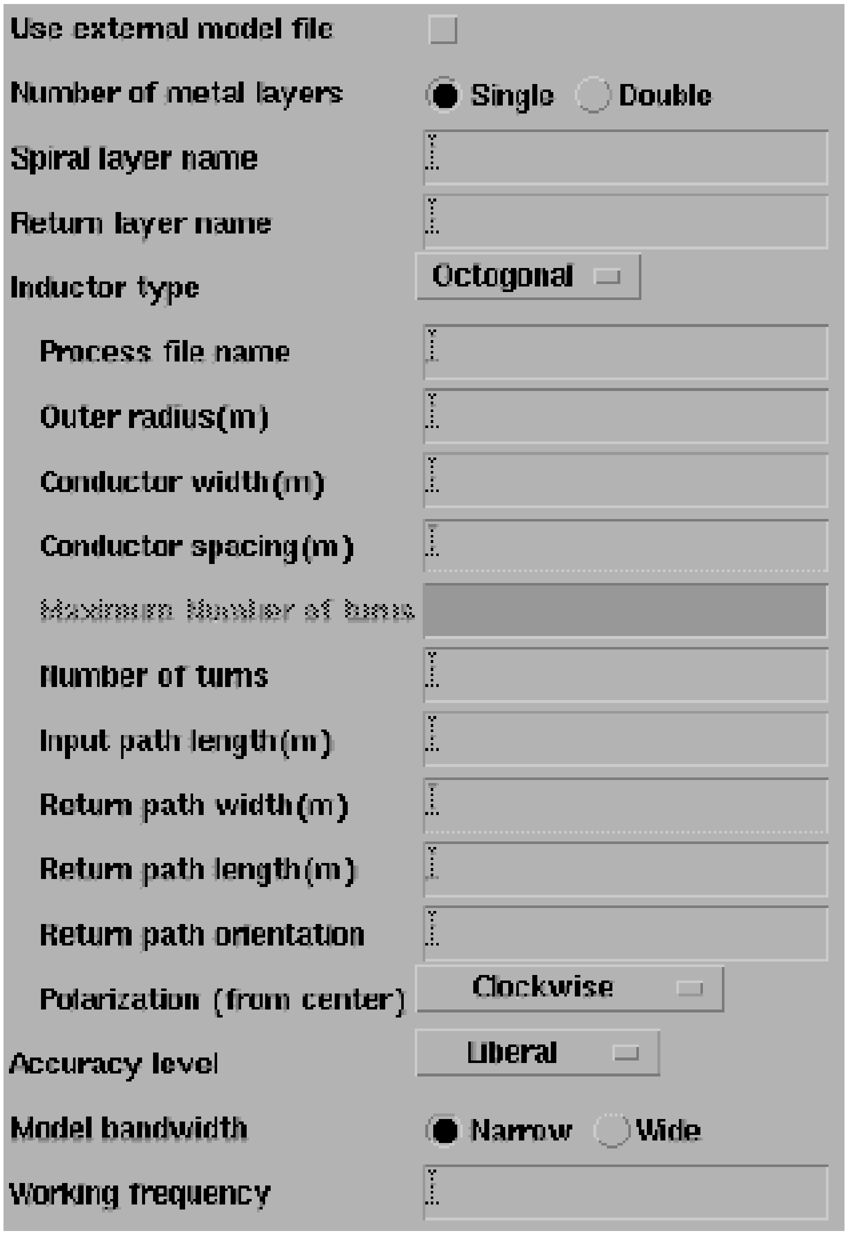


CDF Form for the Octagonal Spiral Inductor

An octagonal spiral inductor is a circular spiral inductor with eight subsections. Figure 11-8 shows the CDF file for an octagonal spiral inductor.

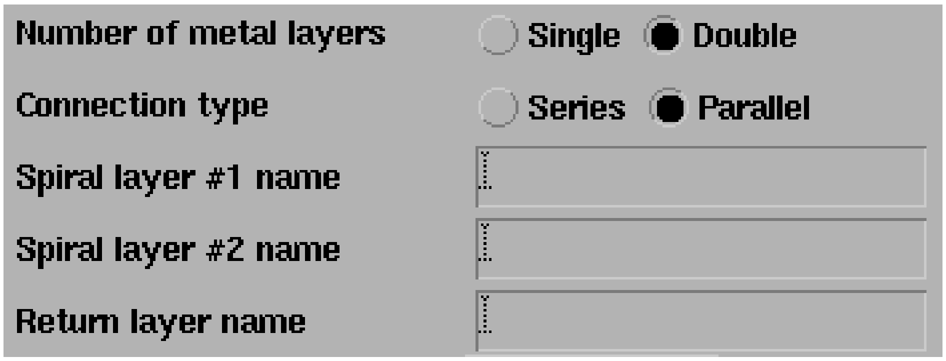
For an octagonal spiral inductor (see Figure 11-7 on page 854), you specify the *Outer radius (m)* of the spiral. The spiral inductor has 8 subsections by default. This replaces the *Outer area length (m)* and *Outer area width (m)* CDF parameters for the rectangular spiral inductor (see Figure 11-6 on page 852).

Figure 11-8 CDF Form for an Octagonal Spiral Inductor

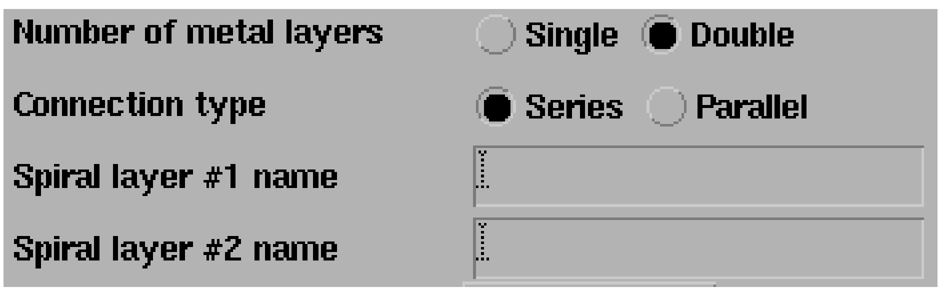


Double-Layered Spiral Inductors

When you highlight *Double* for *Number of metal layers*, the CDF form changes to add a new entry, *Connection type*. In addition, the *Spiral layer name* field is replaced by two fields: *Spiral layer #1 name* and *Spiral layer #2 name.* If you highlight *Parallel* for the *Connection type*, the *Return layer name* field remains.



If you highlight *Series* for the *Connection type*, the *Return layer name* field is removed and one of the spiral layers functions as the return path.



Process File and Parameter Update

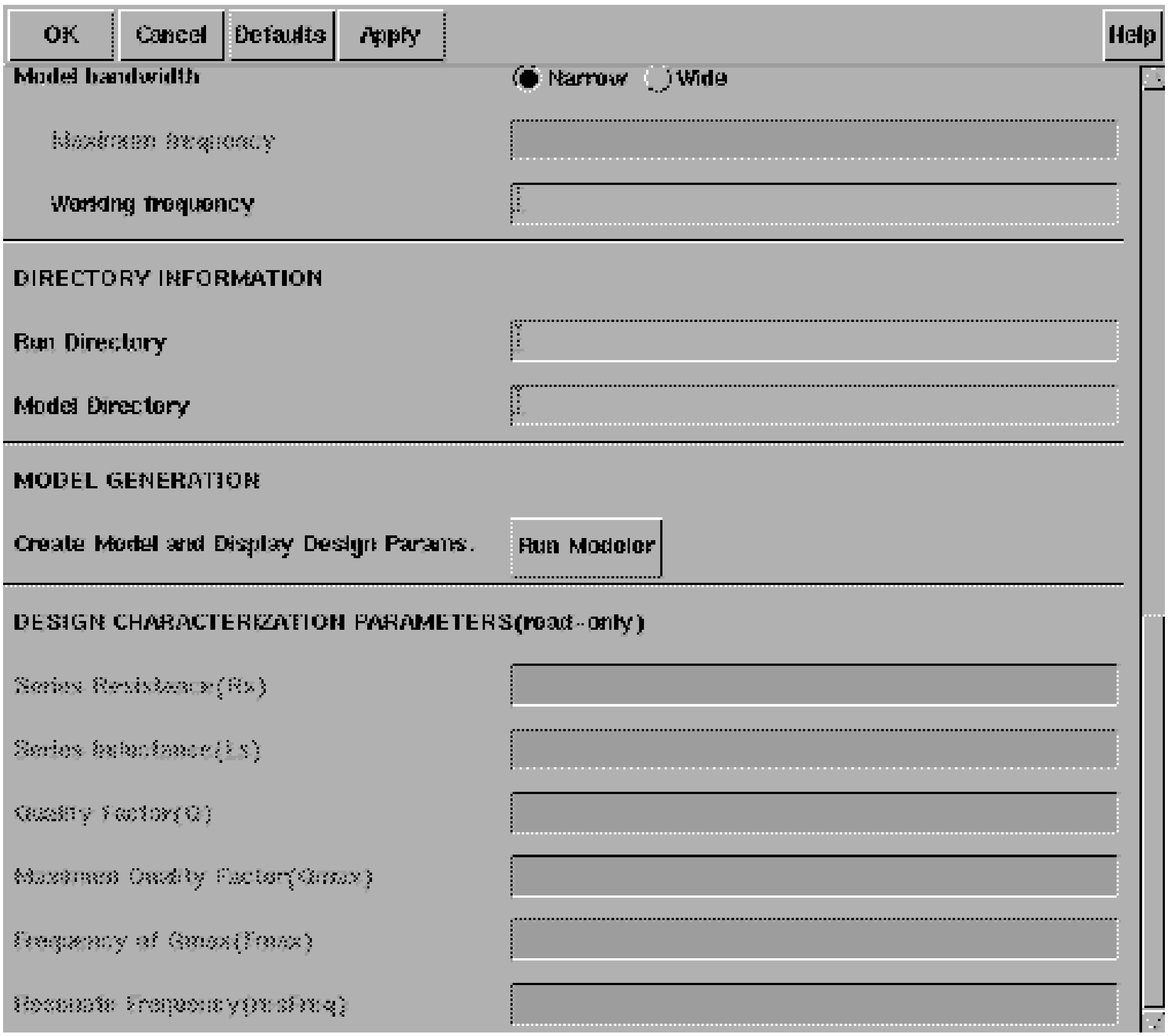
If you specify and apply new parameters, the model extraction is performed at netlisting time. If you do not change parameters, the previous model file is used in simulation and the model creation step omitted. If any parameters change, the inductor model is recreated. Because the process file name is the only reference, if you change the process file contents, the model is not updated unless you specify a new file name. Consequently, we recommend that you specify a new process file name for each new set of parameters.

Spiral Inductor Modeling under the RF Sub-Menu

You can launch the Spiral Inductor Modeler without going into the schematic by choosing *Tools—RF—Spiral Inductor Modeler* in the Simulation window. The Spiral Inductor Modeler UI accessed using the RF sub-menu is similar to the CDF form for spiral inductors.

The major difference is the addition of directory information and the design parameters at the bottom of the UI as shown in Figure 11-9.

Figure 11-9 Bottom of the Spiral Inductor Modeling GUI



Setting Up and Running the Simulation

After you specify the physical geometries of the spiral inductor such as length, width, spacing, and frequency information, you must specify the run directory where the auxiliary files created by the modeler engine are located. You must also specify the model directory where the extracted model file is saved. You launch the modeler engine by clicking on *Run Modeler*. After the extraction, the design characterization parameters are displayed at the bottom of the form. The bottom half of the UI is shown in Figure 11-9 on page 858.

List of Design Parameters

In Figure 11-9 on page 858,six parameters for the spiral inductor are displayed at the bottom of the GUI. The *Working frequency* (see Figure 11-8 on page 856) mentioned below is a CDF parameter specified in the UI. The six parameters in the Spiral Inductor GUI are

- ***Series Resistance*****:** The AC resistance of the spiral at the working frequency with the skin-depth effect included.
- ***Series Inductance*****:** The total inductance of the spiral inductor with the skin depth effect included.
- ***Quality Factor*****:** The Q-factor at the working frequency.
- ***Maximum Quality Factor*****:** The maximum quality factor of the inductor within the 10 MHz to 20 GHz frequency range.
- ***Frequency of Qmax*****:** The frequency at which *Qmax* occurs.
- ***Resonant Frequency:*** The frequency where *Q*=0. The search is from 10 MHz to 20 GHz. If no resonant frequency is found, a value of -1 is displayed.

These parameter calculations are based on the equivalent circuit model. Because of the difference between the wide-bandwidth and narrow-bandwidth models, the slight difference in the design parameters between the two models is to be expected.

Including the Inductor Model Files in the Simulation

After model extraction, a netlist file that describes the equivalent circuit model of the inductor is written to the model directory you specify. You can use this file directly with the *Use external model file* option on the spiral inductor CDF form. To do this, perform the following steps:

1. In the Simulation window of the analog design environment, choose *Setup —Simulation Files*.

   The Simulation Files Setup form appears.

2. In the Simulation Files Setup form, specify the full path to the model definition file.

Equivalent Circuit Models

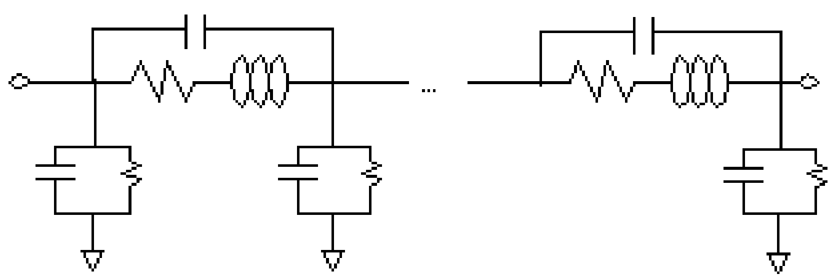
This section discusses the two types of equivalent circuit models:

- Narrow bandwidth models
- Wide bandwidth models

Narrow Bandwidth Model

This narrow bandwidth model, as shown in Figure 11-10, is constructed directly from the RLCG matrixes obtained at the working frequency. All the elements are fully coupled through capacitors and inductors.

Figure 11-10 Narrow-Bandwidth Model for a Spiral Inductor



Because the RLCG parameters are functions of frequency, the above model is valid only in the vicinity of the working frequency. It is therefore a narrow-bandwidth model. Expect large simulation error for frequency components outside of the bandwidth. Because this model only requires the extraction of RLCG parameters at a single frequency, the extraction is faster. The resulting model is also guaranteed to be passive and stable.

Equivalent Circuit Order

The number of PIs in the equivalent circuit depends on the number of turns in the spiral inductor. For single-spiral layer inductors and the double-spiral layer with parallel connection inductors, the number of PIs equals the number of turns. For double spiral layer with series connection inductors, the number of PIs equals twice the number of turns.

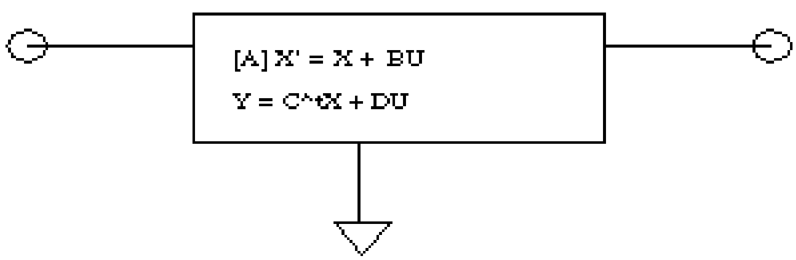
Working frequency selection and model accuracy

Because the narrow-bandwidth model is built using parameters extracted at the working frequency, the model is accurate only in the vicinity of the working frequency. The error increases as the frequency moves farther from the working frequency. Because the error is caused mostly by the large variation of the RLCG parameters with frequency, the higher the frequency, the narrower the bandwidth. If the skin depth or the substrate loss is not significant at the working frequency, the narrow-bandwidth model is rather accurate from DC up to the working frequency.

Wide Bandwidth Model

The wide-bandwidth model is designed to work over a wide frequency range. Unlike the narrow-bandwidth model, the RLCG parameters for the wide-bandwidth model are extracted over a frequency range. You specify the maximum frequency, and the minimum frequency is set to 10 MHz. Modeling the spiral inductor as a two-port system, permits the S-parameters of the two-port to be calculated at each frequency point. The resulting list of S-parameter matrixes are then sent to a rational fitting module to create a wide-bandwidth nport component. Figure 11-11 shows the mathematical representation of the wide bandwidth model for the inductor.

Figure 11-11 Spiral Inductor Wide-Bandwidth Model



Passivity of the Wide Bandwidth Model

The passivity of the wide-bandwidth models is tested over a frequency range by checking the energy conservation for each port at a set of sample frequencies. The default frequency range is from DC to 30GHz, which is normally wide enough for RF applications. Because the model is only guaranteed to work within this frequency range, it might not work for a simulation outside of the tested range. A mechanism is provided to change the test range.

Set maximum Frequency for Passivity Check

You can reset the maximum frequency for the passivity checking by setting the environment variable *maxFreqForPassivityTest.*

```
setenv maxFreqForPassivityTest 50e9
```

Automatic Model Switch

To avoid simulation failure caused by non-passivity of the wide-bandwidth model, the modeler automatically switches to a narrow-bandwidth model if the passivity test fails. You can prevent this automatic switching by setting an environment variable in the window where you start *icms*.

```
setenv pcmModelAutoSwitch OFF
```

If you turn off the switch, the wide-bandwidth model is created and a warning message issued. Spectre uses the model in simulation even though it is not passive.

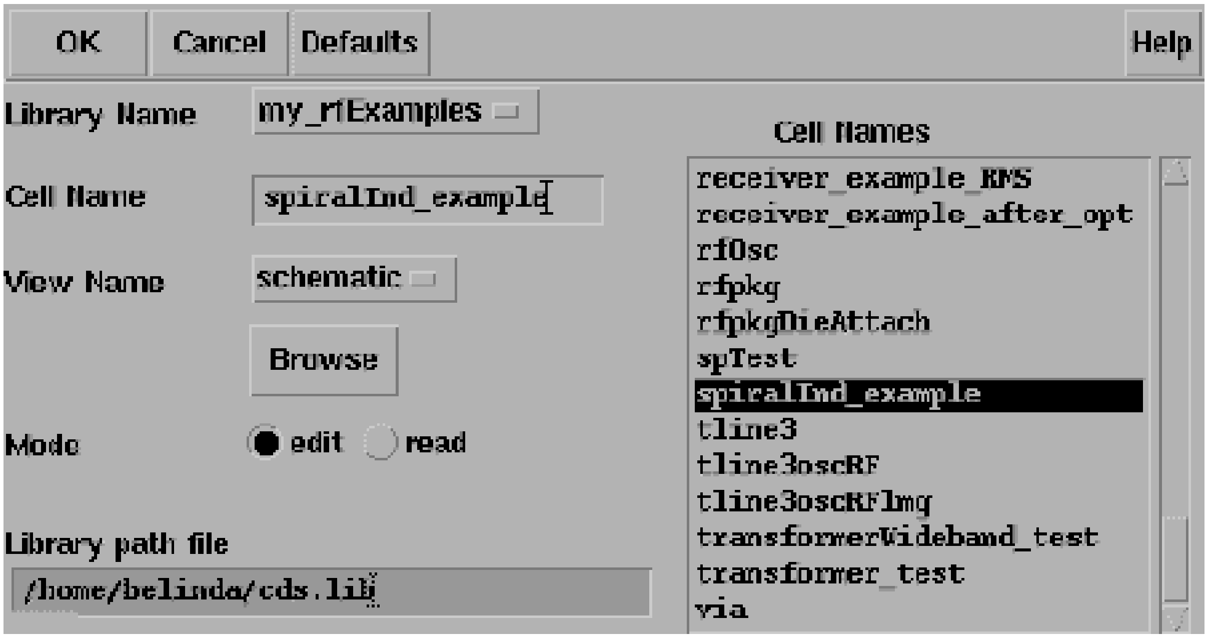
Accuracy Analysis

The *Accuracy level* entry on both the CDF and the RF sub-menu UI forms controls the mesh density for both the capacitance and inductance extraction. For capacitance, the number of subsections per width is three, five, or seven. These values correspond to the *Liberal*, *Moderate* and *Conservative* accuracy levels. Inductance calculation is similarly correlated with the accuracy level. Three, five, or seven filaments are used in the transverse direction to the current, corresponding to the three accuracy levels. The error is less than 3% at the *Liberal* level, and about 1% at the *Conservative* level.

Rectangular Spiral Inductor Example

1. In the Command Interpreter Window (CIW), choose *File – Open*.

The Open File form appears.



2. In the Open File form, do the following and then click *OK*:

a. Choose *my_rfExamples* for *Library Name*.

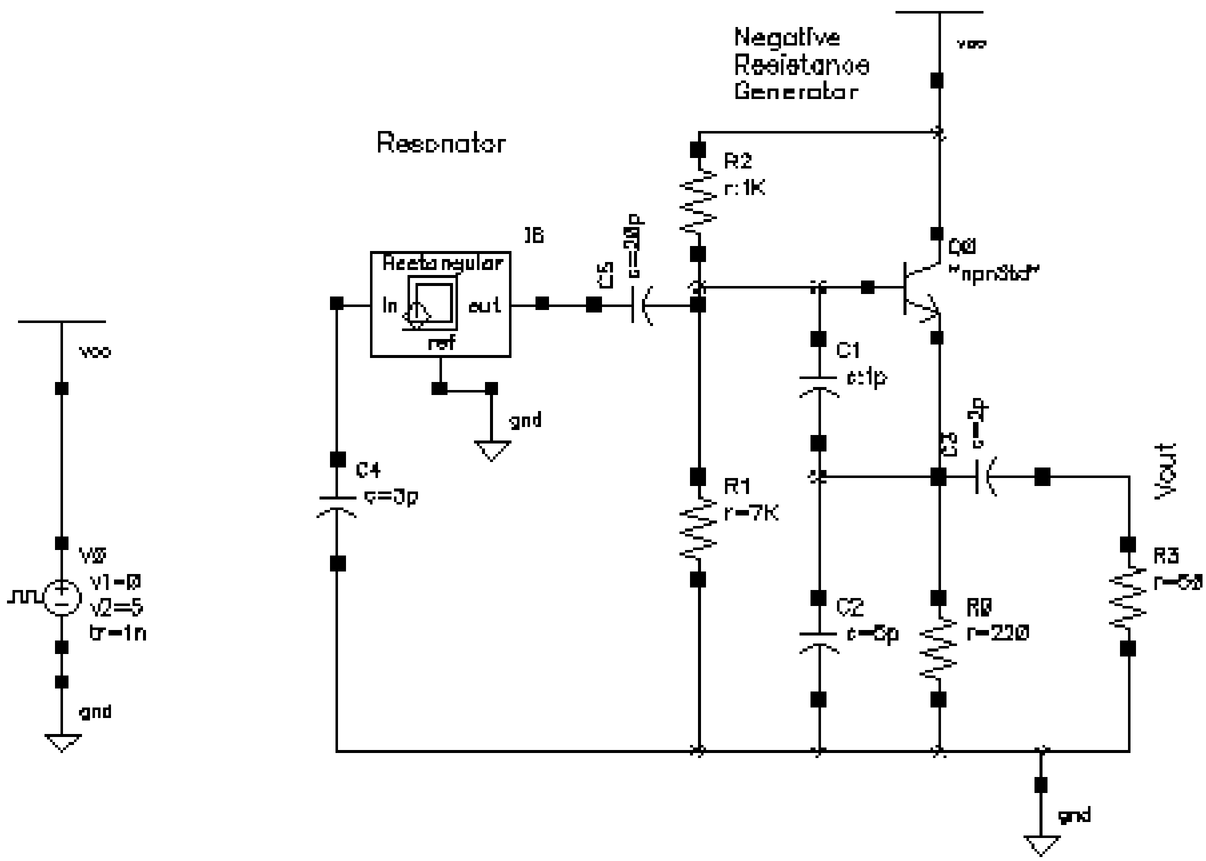
Select the editable copy of the *rfExamples* library that you created as described in Chapter 3, “Setting Up for the Examples.”

b. Highlight *spiralInd_example* for *Cell Name*.

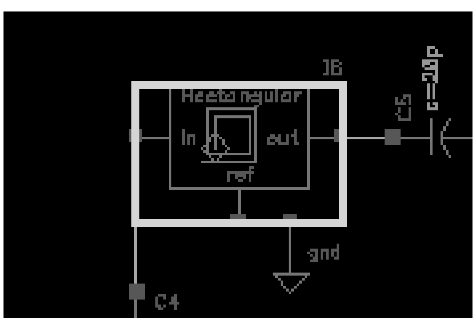
c. Choose *schematic* for *View Name*.

d. Click *OK*.

The *spiralInd_example* schematic appears.



3. In the Schematic window, select the Rectangular Spiral Inductor symbol and choose *Edit – Properties – Objects*.



The Edit Object Properties form appears.

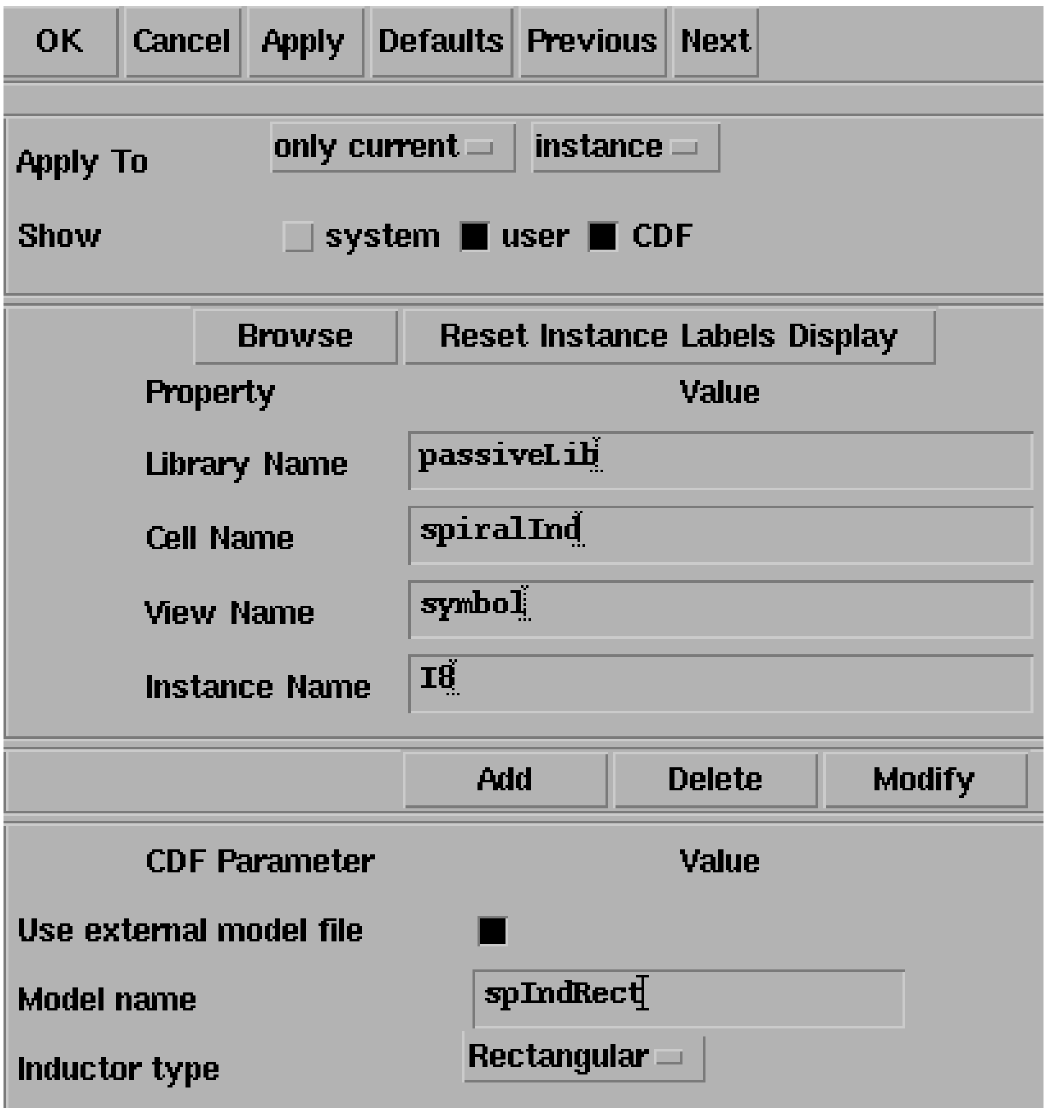
4. In the Edit Object Properties form, highlight the *Use external model file* button.

The form changes to let you specify the file name for the model you are about to extract.

Note: This example uses an external model file, but you can also run the example by specifying all the values in the Edit Object Properties form. To see this alternative method, look at the“LNA Example with Bonding Pads” on page 881. You can model both bondpads and spiral inductors using either method.

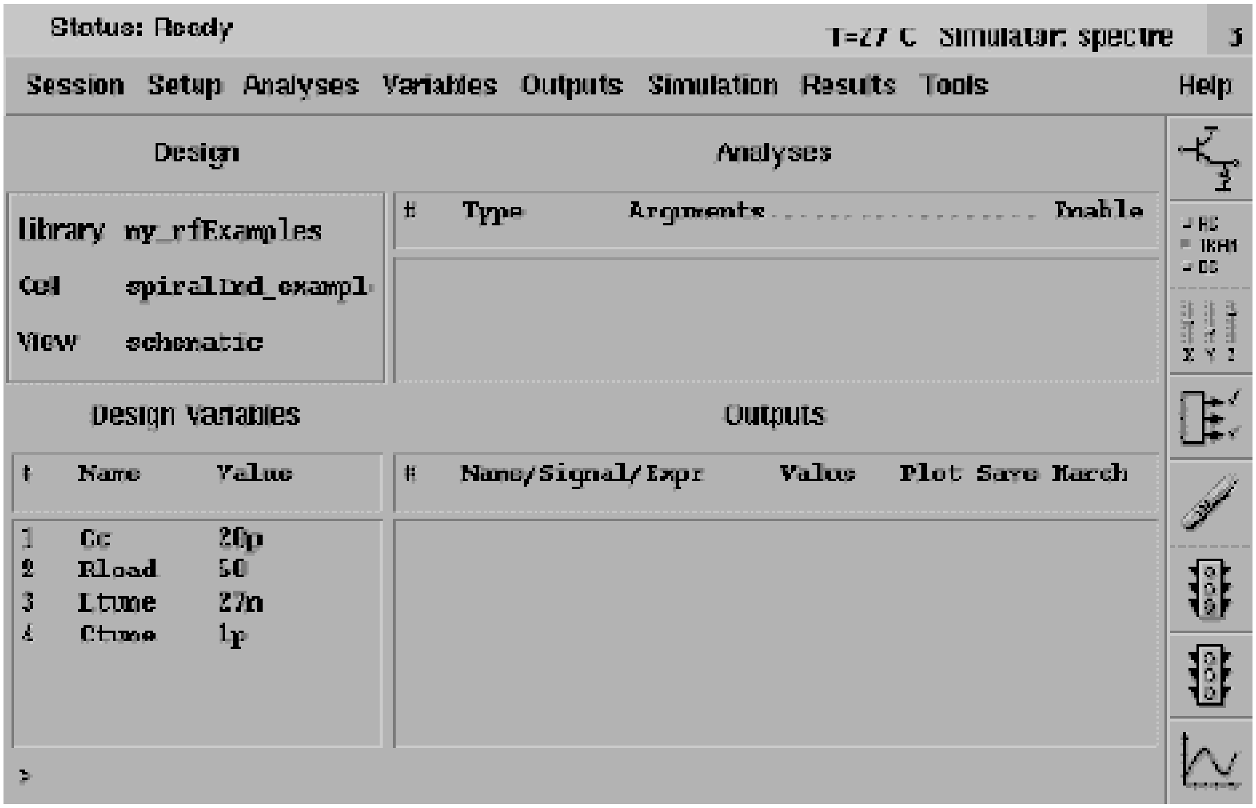
5. In the Edit Object Properties form, type `spIndRect` in the *Model name* field.
6. Click *OK*.

   The completed form looks like this.



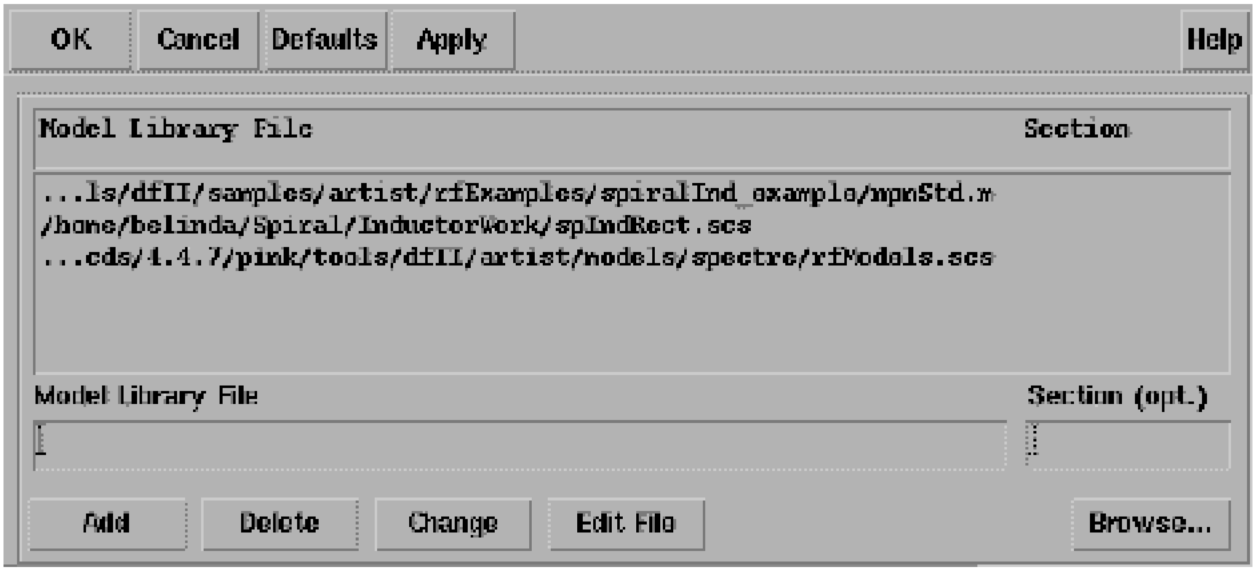
7. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation Window appears.



8. In the Simulation window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.



9. In the *Model Library File* field, type the full path to the following files. Click on *Add* after you type each path and file.

`$CDSHOME/tools/dfII/samples/artist/rfExamples/spiralInd_example/npnStd.m`

`/home/belinda/Spiral/InductorWork/spIndRect.scs`

`$CDSHOME/tools/dfII/samples/artist/models/spectre/rfModels.scs`

In the file specifications

- The `npnStd.m` file is in the *spiralInd_example* directory in the *rfExamples* library.
- The `spIndRect.scs` file path points to the local run directory where you want the extracted model file to be written.
- The `rfModels.scs` file is in the *spectre* models directory.

In the file specifications, `$CDSHOME` points to your software installation directory.

10. Click *OK*.

Opening the Spiral Inductor Modeler

1. In the Simulation window, choose *Tools—RF—Spiral Inductor Modeler*.

The Spiral Inductor Modeler GUI appears.

2. In the Spiral Inductor Modeler form, do the following:

a. Highlight *One* for *Number of metal layers*.

b. Type `metal1` for *Spiral layer #1 name*.

c. Type `metal2` for *Return layer name*.

d. Choose *Rectangular* in the *Inductor type* cyclic field.

e. In the *Process file name* field, type the path to the `cmos.data` file.

```
$CDSHOME/tools/dfII/samples/artist/rfExamples/spiralInd_example/
cmos.data
```

The `cmos.data` file is in the `spiralInd_example` directory in the `rfExamples` library.

a. Type `300u` for *Outer area length(m)*.

b. Type `300u` for *Outer area width(m)*.

c. Type `10u` for *Conductor width(m)*.

d. Type `7u` for *Conductor spacing(m)*.

e. Type `7` for *Number of turns*.

f. Type `13u` for *Return path width(m)*.

g. Type `110u` for *Return path length(m)*.

h. Type `-90` for *Return path orientation*.

i. Choose *Clockwise* from the *Polarization (from center)* cyclic field.

j. Choose *Liberal* from the *Accuracy level* cyclic field.

k. In the *Model Name* field, type the name that you choose for the model, `spIndRect`.

l. Highlight *Narrow* for *Model bandwidth*.

m. Type `1.5G` for *Working frequency*.

n. In the *Run Directory* field, type the full path to the run directory.

```
/home/belinda/Spiral/InductorWork
```

o. In the *Model Directory* field, type the full path to the model you are creating.

`/home/belinda/Spiral/InductorWork`

The completed form looks like the following two illustrations:

Figure 11-12 Top part of the Spiral Inductor Modeler Form

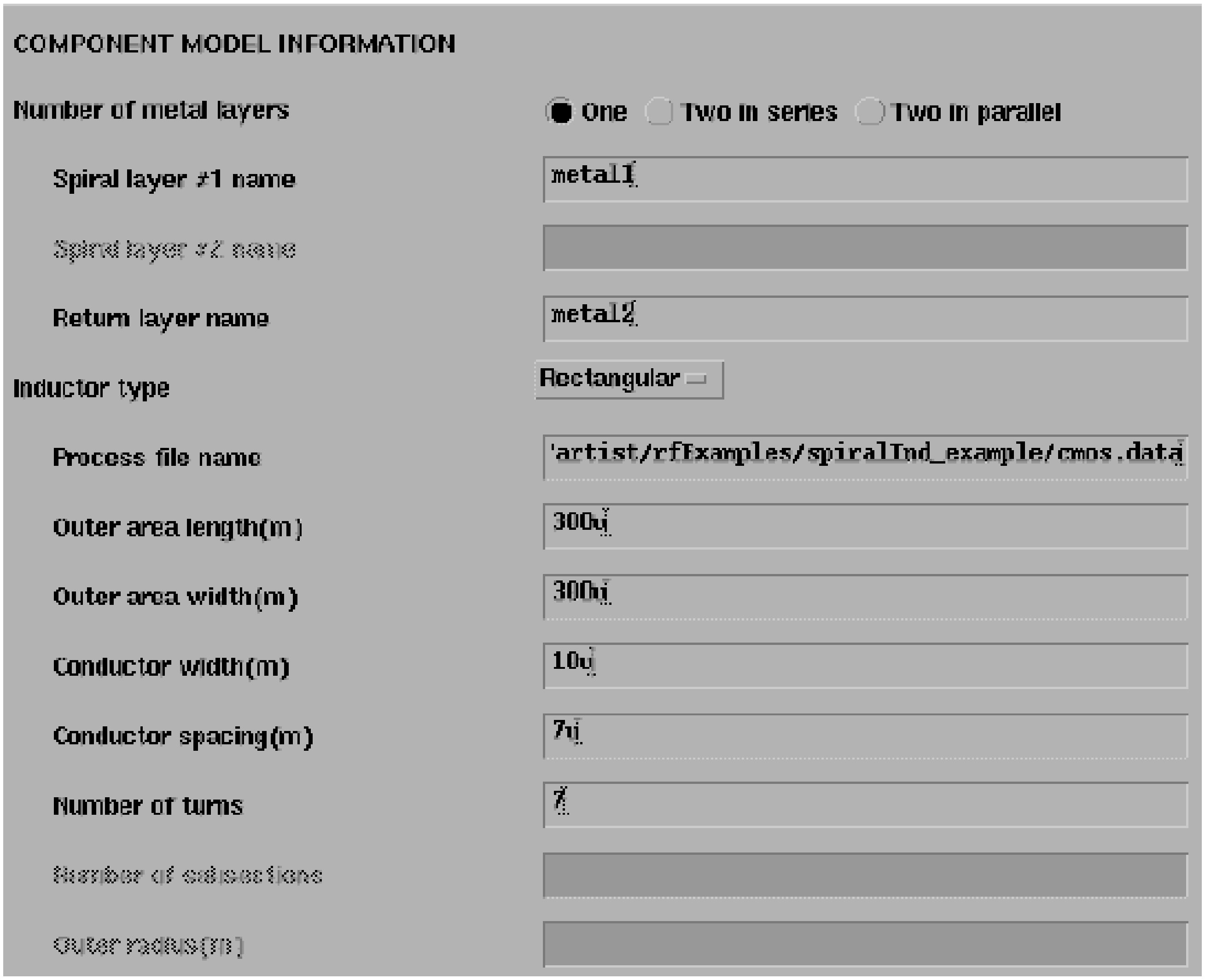


Figure 11-13 Middle Part of the Spiral Inductor Model Form

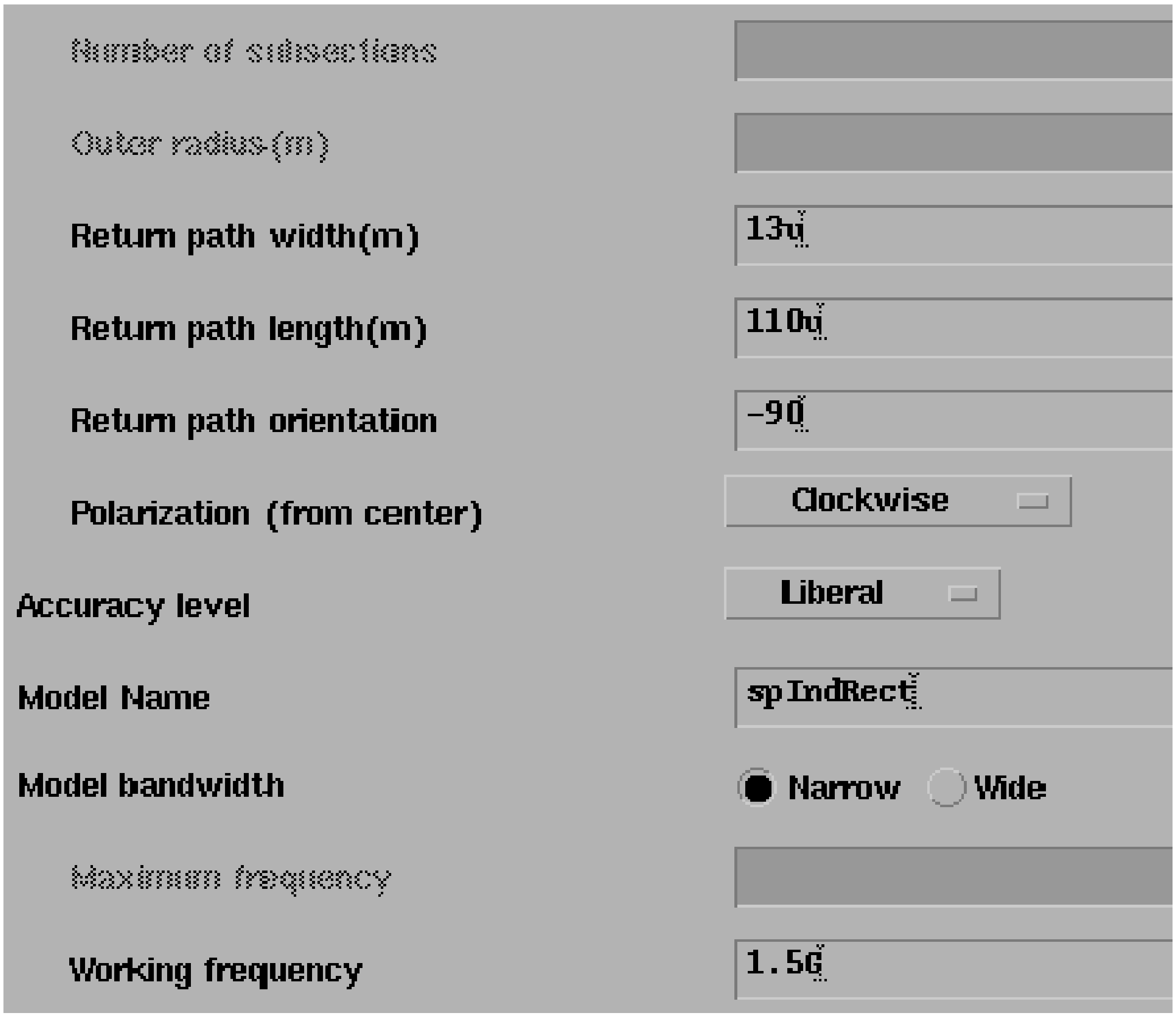
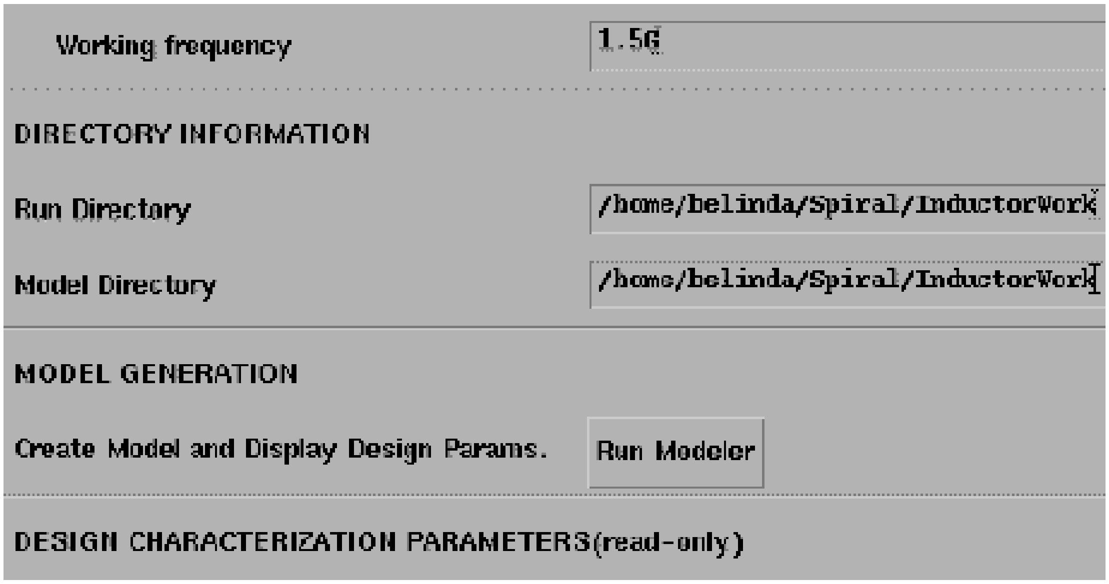


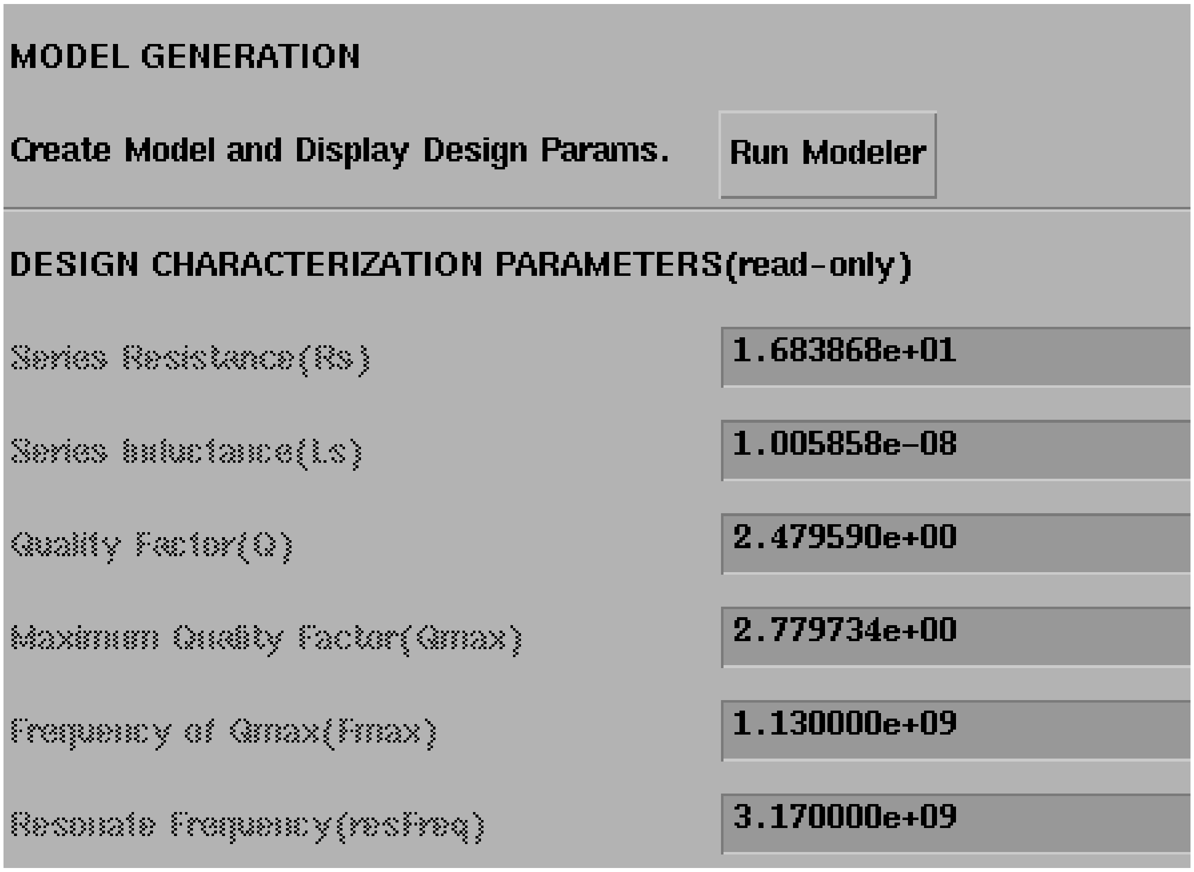
Figure 11-14 Bottom of the Spiral Inductor Model Form



3. Click *Run Modeler* next to *Create Model and Display Design Params*.

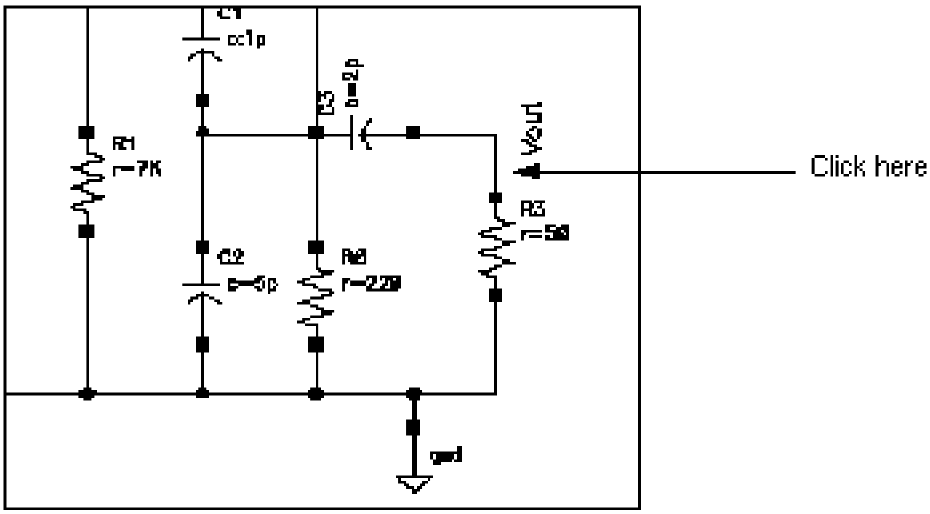
   The model is created. When the process is completed, the calculated parameters are displayed in the read-only fields at the bottom of the Spiral Inductor Modeler form.

Figure 11-15 Spiral Inductor Parameters



4. In the Simulation window, choose *Outputs – To Be Plotted – Select On Schematic*.

5. In the Schematic window, click on the *Vout* node.



6. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.

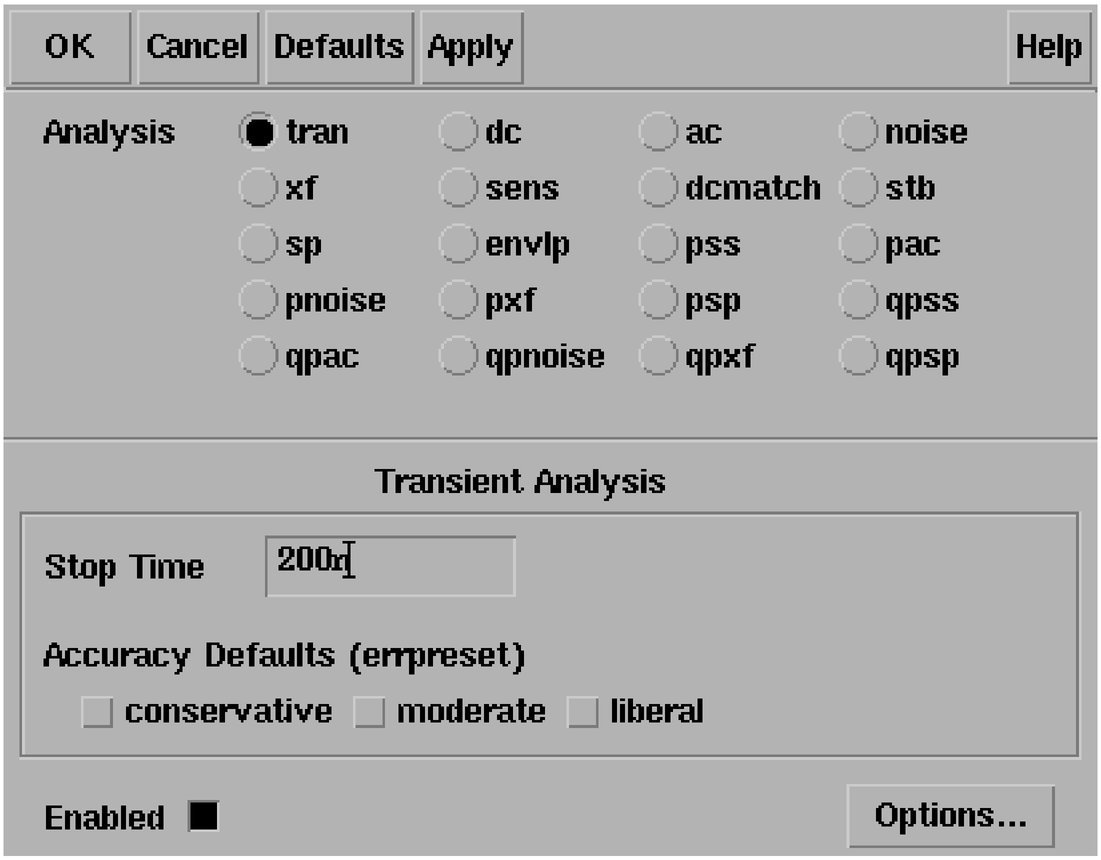
7. In the Choosing Analyses form, do the following.

a. Highlight *tran* for *Analysis*.

b. Type `200n` for *Stop Time*.

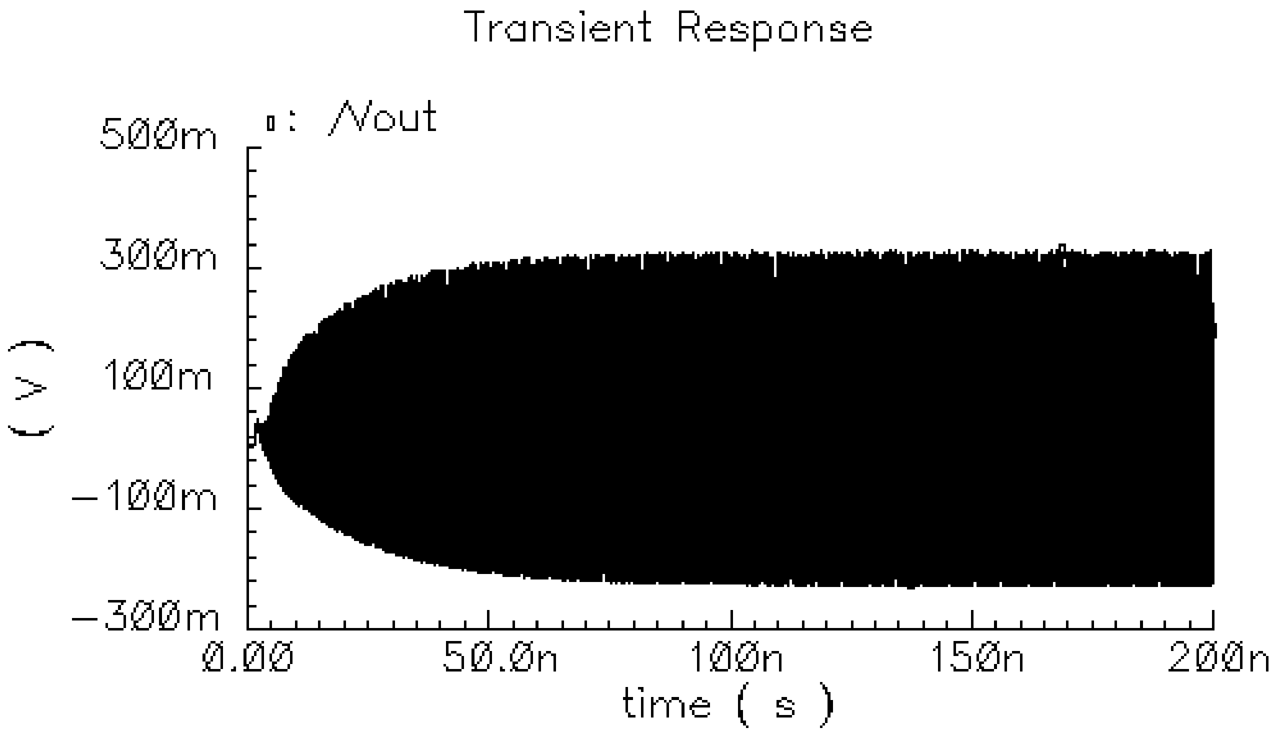
c. Highlight *Enabled*.

d. Click *OK*.



8. In the Simulation window, choose *Simulation – Netlist and Run*.

The simulation runs. When the simulation completes, the Waveform window appears with the plot for the example.



Modeling Bonding Pads

On-chip passive components, especially those situated on lossy substrate and operating at high frequencies, need careful modeling. Bonding pads are the interface between the chip and outside world, and their behavior significantly affects circuit performance. The core algorithms used in modeling bonding pads are also used in the capacitance extraction for spiral inductor modeling.

Capacitive coupling, appearing as capacitance and resistance caused by displacement current, normally is modeled in high-frequency or high-speed applications because of its coupling characteristics. This displacement current is different from inductive current that is characterized by inductance. Accurate extraction of these coupling effects, however, requires solving Poisson's equation and becomes computation-intensive. For lossy substrate, when capacitive coupling and resistive coupling are of the same order of magnitude, the frequency-dependent effects require additional extraction efforts. This bonding pad model uses the complex multi layered Green's function [2] to capture capacitive couplings and loss information. It uses IES3[1] to accelerate its computation and further uses a novel technique to fit the frequency-dependent data. Because of this technique, you can perform PSS analysis as well as all the conventional circuit analyses.

The on-chip passive component modeler simulates spiral inductors, planar transformers, and bonding pads for RF applications. The modeler has an extraction engine based on EM (Electromagnetics) techniques. The analog design environment calls this engine using a command interface.

Use the Bonding Pad modeler at schematic level. As long as you know physical parameters, you can use the modeler through CDF with LNAs, mixers, and receivers.

Based on the limited pad placement provided by the Bond Pad tool, it is more suited for use in a Top Down design flow and for block level design. These are cases involving a limited number of pads and the options for bond pad placement are limited. Or the Bonding Pad modeler might be useful when you are designing a block for quick what-if analysis as you optimize the location of block bond pads.

You should use SCA for final device verification since it can handle any number pads with any relative location, i.e, SCA understands two dimensional pad placement.

There are two use models for the Bonding Pad modeler.

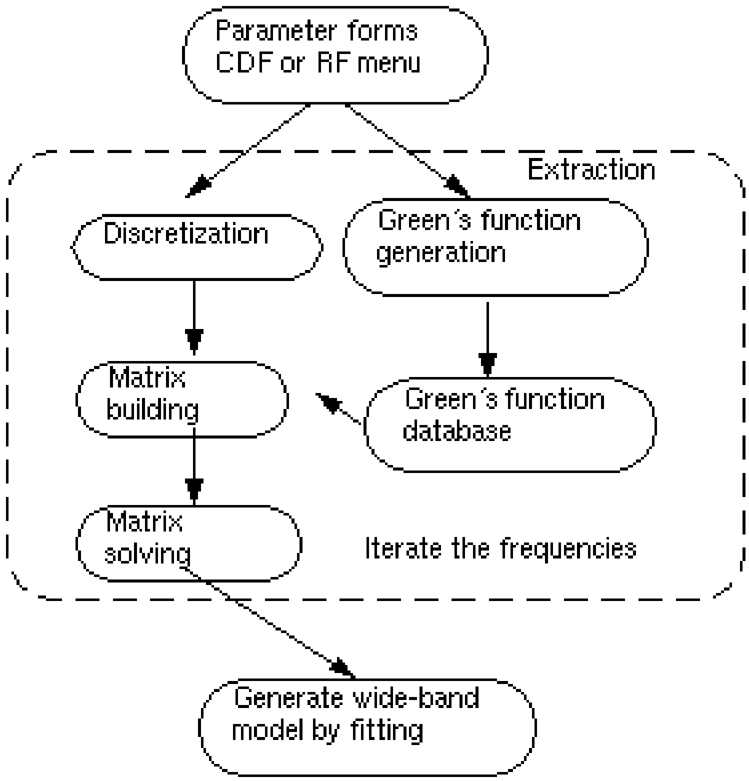
- One is to use the modeler in the schematic flow. You place a bonding pad symbol in your schematic design. You then change the CDF parameters that specify the input parameters for the modeler in the Edit Properties Objects form. You extract the model at netlisting time. If you netlist a circuit before you simulate, the extraction tool is automatically invoked and added or changed parameters are inserted into the netlist.
- The other use model is to launch the bonding pad modeler using the Tools menu in the Simulation window. The bonding pad model is extracted from the info you enter in the Bonding Pad Modeler GUI and written to the specific model directory. The electric parameters of the components are calculated and displayed on the Bonding Pad modeler GUI form.

  You can include the extracted model in Spectre/SpectreRF simulations; you can use multiple copies of the extracted model if necessary, thereby avoiding long extraction times at netlisting. This approach generates only the internal model called a reduced order model or ROM that is stored in a file. You can use this ROM file in CDF form to avoid unnecessarily generating the model during netlisting. The model is only generated during netlisting procedure when the model is new or the parameters are changed.

The current fitting algorithm still has not completely solved the passivity problem. Rarely, a non-passive model might cause simulation failure because of strong positive feedback within a circuit (such as an oscillator). When this happens, you can use a narrow-band model that is accurate at the specified working frequency.

The data flow for using and modeling bonding pads is shown in Figure 11-16.

Figure 11-16 Data Flow for Bondpad Modeling



The remainder of the bonding pad section covers the following topics:

- The Process Files section indicates that you need the same process file used for spiral inductor modeling and refers you back to that section.
- The CDF Parameter Sets section explains details about the bonding pad symbol and its CDF parameters.

Process Files

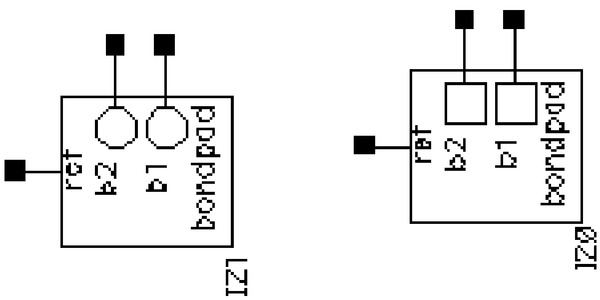
Initially, you must create a process file to describe the vertical profile of the process. These parameters will be used in extraction. The file format is the same as that used for spiral inductors. For more information about creating the file, see “Process File Preparation” on page 845.

Symbols and CDF Parameters for Bonding Pads

The spiral inductors, transformers, and bondpads are stored in the *passiveLib* library. The name of the bonding pad component is *bondpad*. The bonding pad symbol is a 3-port block. You can specify data in either the CDF form from the schematic or using the *Tools—RF—Bond Pad Modeler* GUI in the Simulation window. The bonding pad procedures are similar to those for spiral inductors.

Figure 11-17 shows the symbols for rectangular and octagonal bonding pads.

Figure 11-17 Symbols for Rectangular and Octagonal Bonding Pads



➤ In the Schematic window, choose *Add—Instance* to display the Add Instance form.

In the Add Instance form, at the top

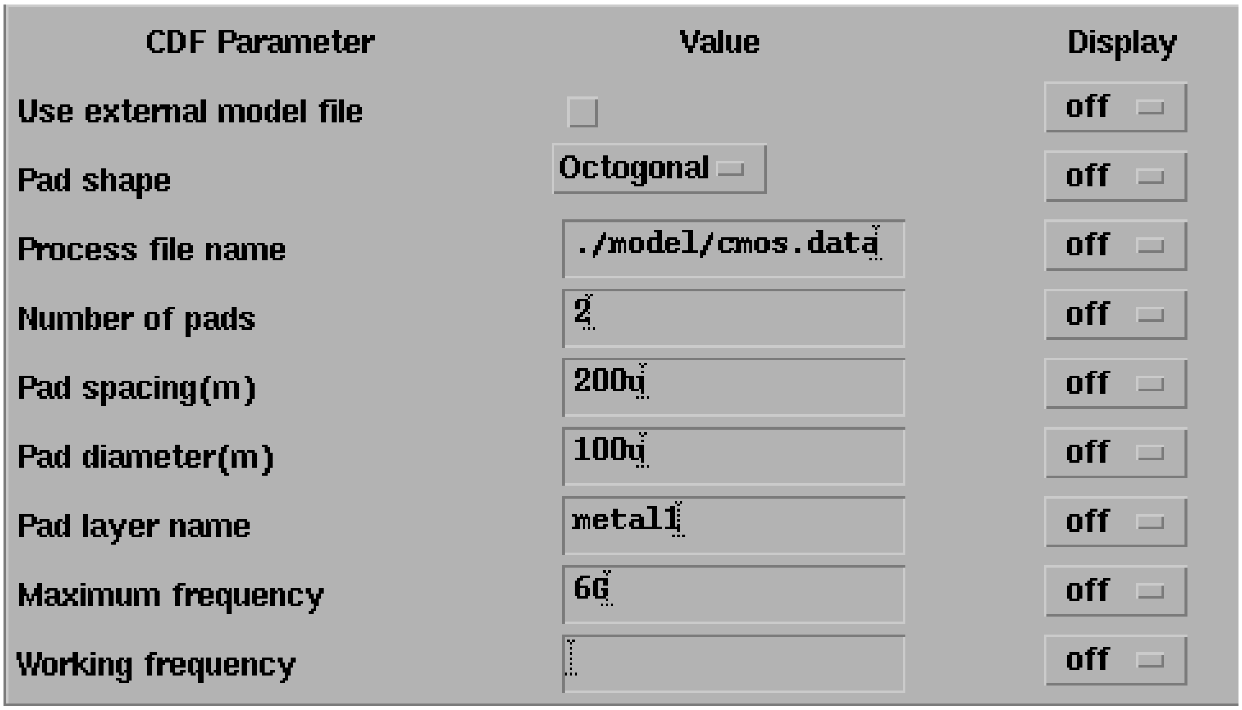
- `passiveLib` displays in the *Library* field
- `bondpad` displays in the *Cell* field
- `symbol` displays in the *View* field

The CDF parameters for the bonding pad element appear at the bottom of the form.

Parameter Sets

Figure 11-18 on page 879 shows the CDF input form for bonding pads.

Figure 11-18 Component Description Format (CDF) for Bonding Pad Models



The parameters you enter are described below:

- **Use external model file** lets you use externally-generated bondpad model files directly in simulations
- **Pad shape** can be square or octagonal. When this cyclic button is changed, the symbol shape changes accordingly.
- **Process file name** can be either an absolute file path or a relative path to the current directory.
- **Number of pads** lets you specify the number of pads.
- **Pad spacing** is the distance between the centers of two adjacent pads. Remember that this is not the spacing between the adjacent edges.
- **Pad diameter** is the diameter size of the single bonding pad. Remember that it is not the size of the edges.
- **Pad layer name** must have its corresponding layer name defined in the process file.
- **Maximum Frequency and Working frequency** – There are two types of models available: wide-band and narrow-band. When you specify the maximum frequency, a

wide-band model accurate up to the maximum frequency is generated. The default maximum frequency is 10 GHz.

When you supply the working frequency and also omit the maximum frequency, the modeler generates a narrow-band model that is accurate at the working frequency.

Modeling Issues

This section explains modeling issues related to the internal workings of the modeler and the generated models.

Wide-Band and Narrow-Band Models

Models are generated either as wide-band or narrow-band models. In most cases, you use the wide-band model because it is more accurate for time-domain simulation.

The wide-band model is generated after fitting the charge-voltage relationships over a frequency range from DC to the maximum frequency. The model is created as a ROM (reduced order model) file. The netlisting code generates an nport, whose *romdatfile* is set to the ROM file to represent bonding-pads.

The narrow-band model only extracts the model exactly at the working frequency. An RC network is derived from that extraction, and it is hoped that this model is accurate for a reasonable frequency range around the working frequency. This RC network is also represented internally as an nport.

Passivity Issues and Use of the Narrow-Band Model

The wide-band model is fitted to a sample of frequencies. For typical RF applications with operating frequency of 1-2GHz, a maximum frequency of 10GHz covers sufficient frequency content and the model is generally very satisfactory. However, the fitting algorithm might produce an unstable model at frequencies higher than the maximum frequency. This is true because the model is extrapolating at these frequencies. This possible nonpassivity has no impact on S-parameter small-signal analysis. It also does not harm simulation for most RF circuits in transient and PSS analyses. It might, however, cause simulation to diverge when it is within a feedback loop of an unstable circuit, such as an oscillator.

You can use the narrow-band model if the wide-band model does not converge. Mathematically, the narrow-band model is an RC network that is guaranteed to be stable and passive.

Note: Currently, the bonding pad model does not model noise effects.

LNA Example with Bonding Pads

RF circuits operating at high frequencies are more susceptible to bonding pad effects. Using the process file in “Process File Preparation” on page 845, the series resistance of the bonding pad at 2GHz is about 450 Ohm and the series capacitance is 0.133p. These values are very similar to those in the literature[3].

This example shows the effects of the bonding pads in the following LNA circuit. The octagonal pad diameter is 100u and the spacing is 200u. It uses the process file described in “Process File Preparation” on page 845. A two-pad model is used, and the coupling effects are included.

The example runs an S-parameter analysis without the bonding pad and plots the results. The bondpad is then added to the schematic, the simulation is rerun, and the results are compared.

Setting Up and Running the S-Parameter Simulation

1. In the CIW, choose *File – Open*.

   The Open File form appears.

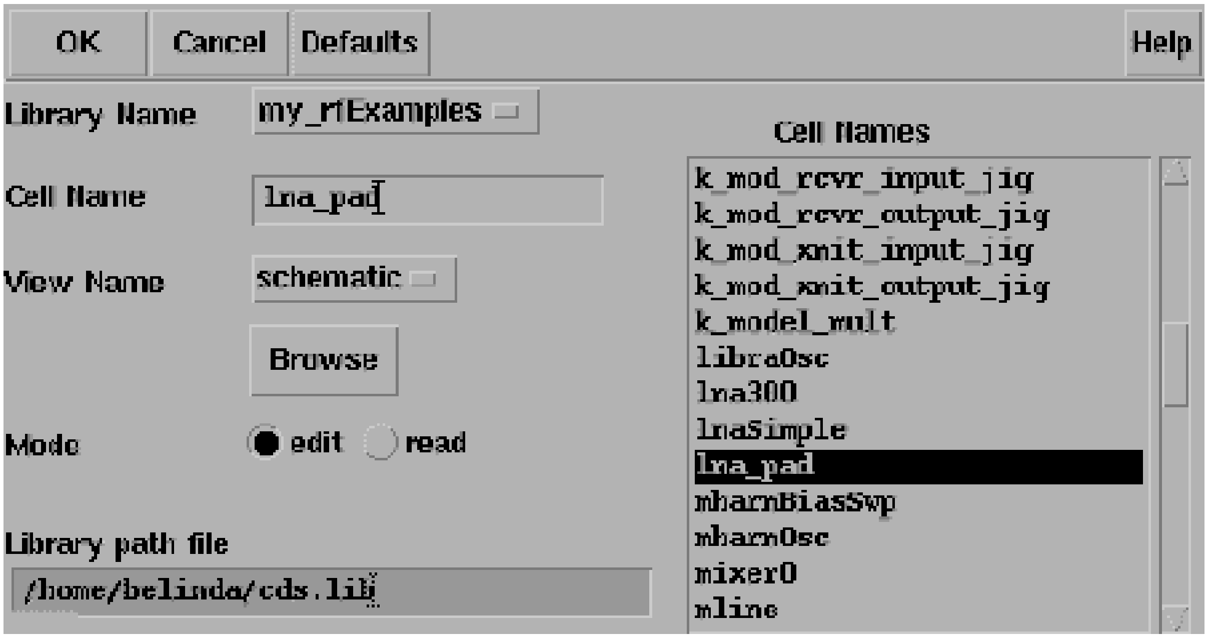
2. In the Open File form, do the following and click *OK*:

   a. Choose *my_rfExamples* for *Library Name*.

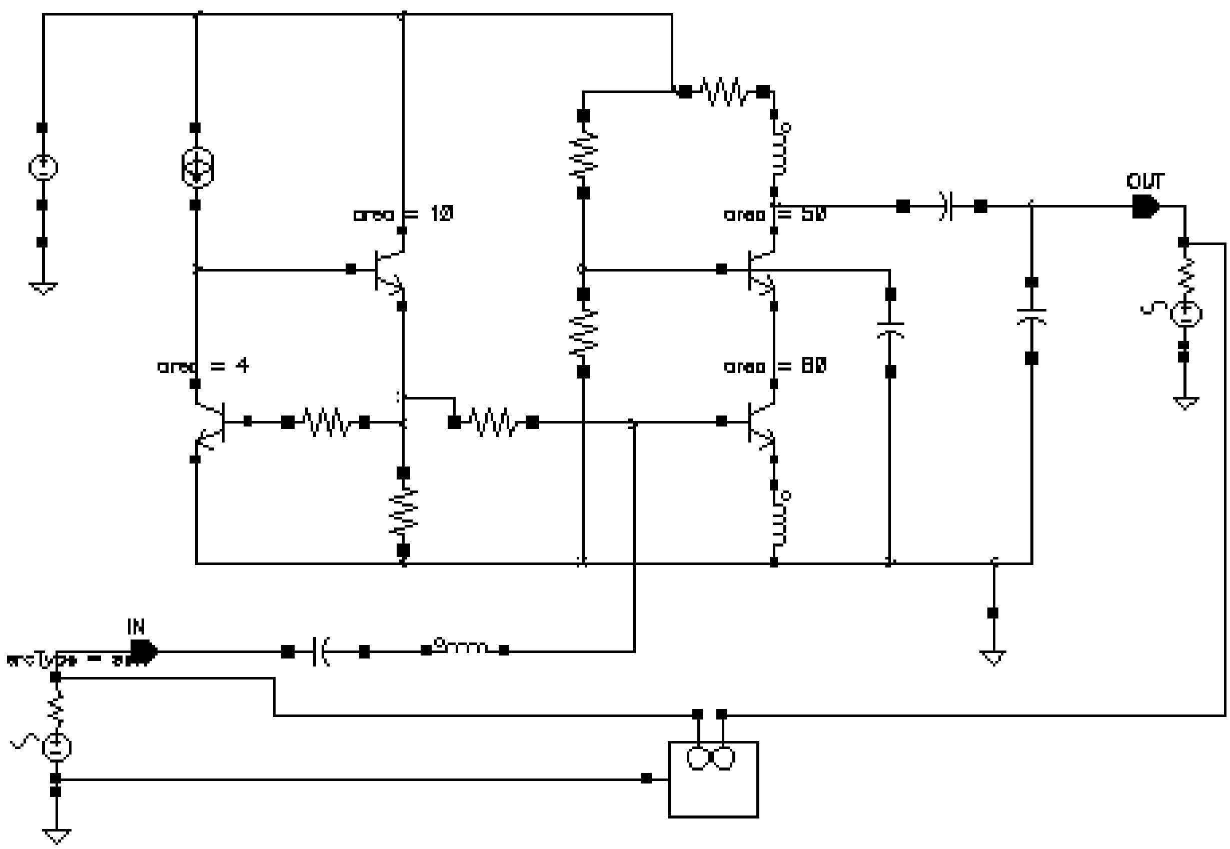
   b. Highlight *lna-pad* for *Cell Name*.

   c. Choose *schematic* for *View Name*.

d. Click OK.


OK
Cancel
Defaults
Help
Library Name
my_rfExamples
Cell Names
Cell Name
lna_pad
View Name
schematic
Browse
Mode
edit
read
Library path file
/home/belinda/cds.lib
k_mod_rcvr_input_jig
k_mod_rcvr_output_jig
k_mod_xmit_input_jig
k_mod_xmit_output_jig
k_model_mult
libraOsc
lna300
lnaSimple
lna_pad
mharmBiasSwp
mharmOsc
mixer0
mline

The schematic appears



3. In the Schematic window, choose *Tools – Analog Environment*.

   The Simulation window appears.

4. In the Simulation window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

5. In the Model Library Setup form, do the following:

   a. In the *Model Library File* field, type the complete path to the *npnStd.scs* file in your software installation hierarchy.

   b. Click on *Add*.

The completed form looks like this.

c. Click *OK*.

6. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.

7. In the Choosing Analyses form, do the following:

   a. Highlight *sp* for *Analysis*.

   b. Highlight *Frequency* for *Sweep Variable*.

   c. Highlight *Start – Stop* for *Sweep Range*.

   d. Type `100M` for *Start* and `5G` for *Stop*.

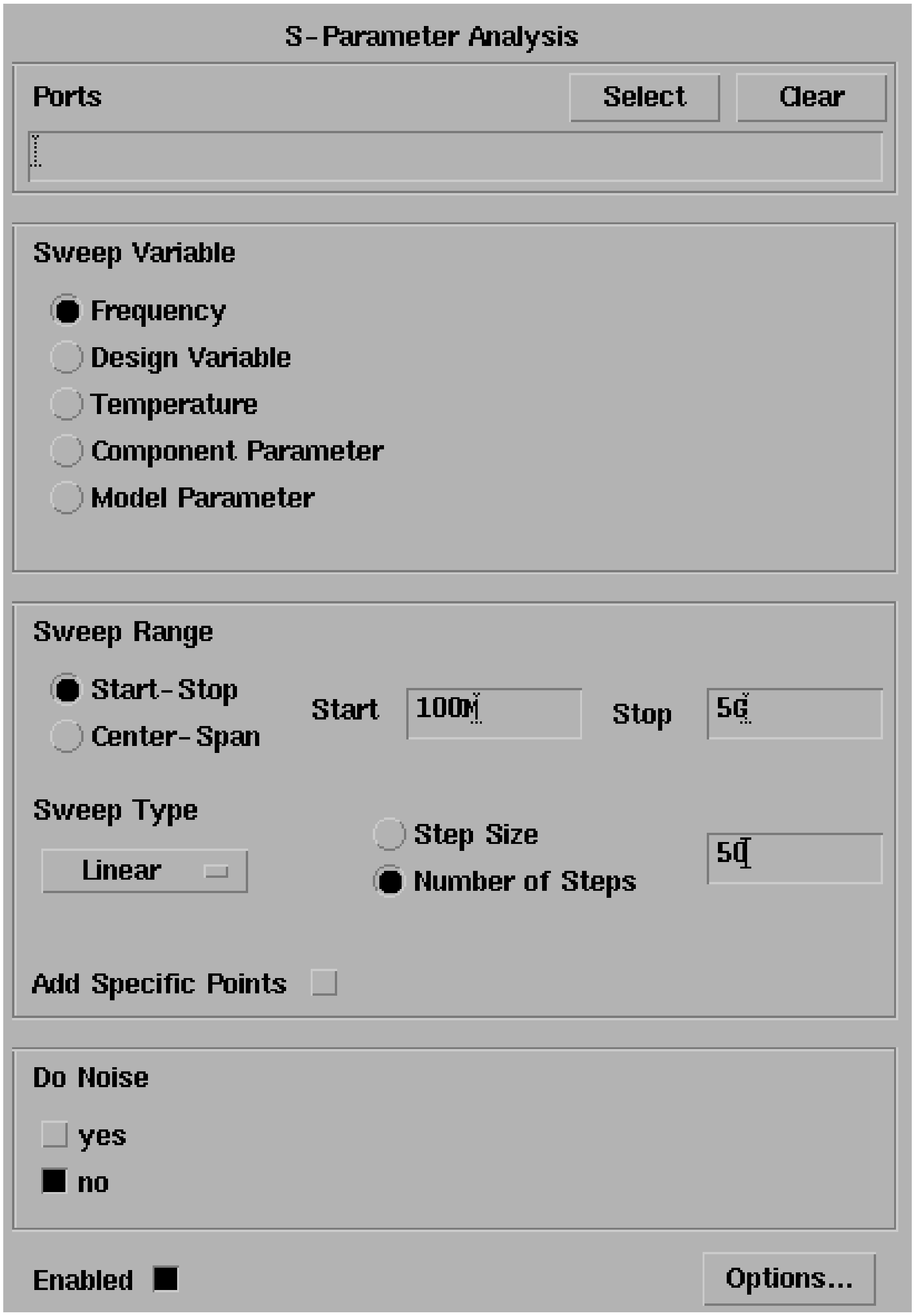
   e. Choose *Linear* for *Sweep Type*.

   f. Highlight *Number of Steps* and type `50` in the adjacent field.

   g. Highlight *no* for *Do Noise*.

   h. Highlight Enabled.

The completed form looks like the this.



8. Highlight *OK*.

9. In the Simulation window, choose *Simulation – Netlist and Run*.

Be sure to check the CIW for a message that the simulation completed successfully.

Plotting the Results

1. In the Simulation window, choose *Results – Direct Plot – Main Form.*

The S-Parameter Results form appears.

2. In the S-Parameter Results form, do the following:

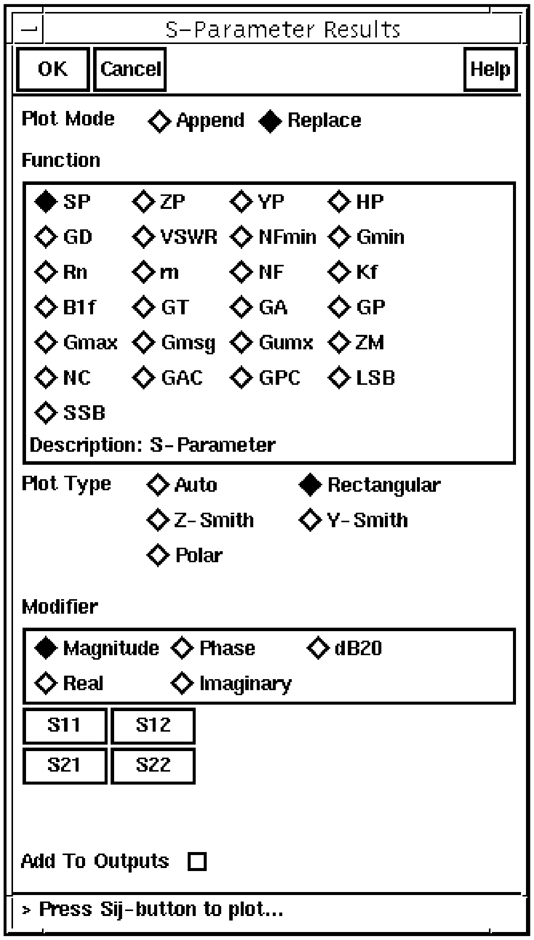
a. Highlight *Replace* for *Plot Mode*.

b. Highlight *SP* for Function.

c. Highlight *Rectangular* for *Plot Type*.

d. Highlight *Magnitude* for *Modifier*.

The completed form looks like this.



a. Click on *S21*.

The plot appears in the Waveform window. Leave all the windows up on your screen so you can resimulate and append the simulation results with the bondpad to the current plot.

Setting Up and Simulating with the Bondpad

1. In the Schematic window, choose *Add – Instance*.

   The Add Instance form appears.

2. In the Add Instance form, click on *Browse*.

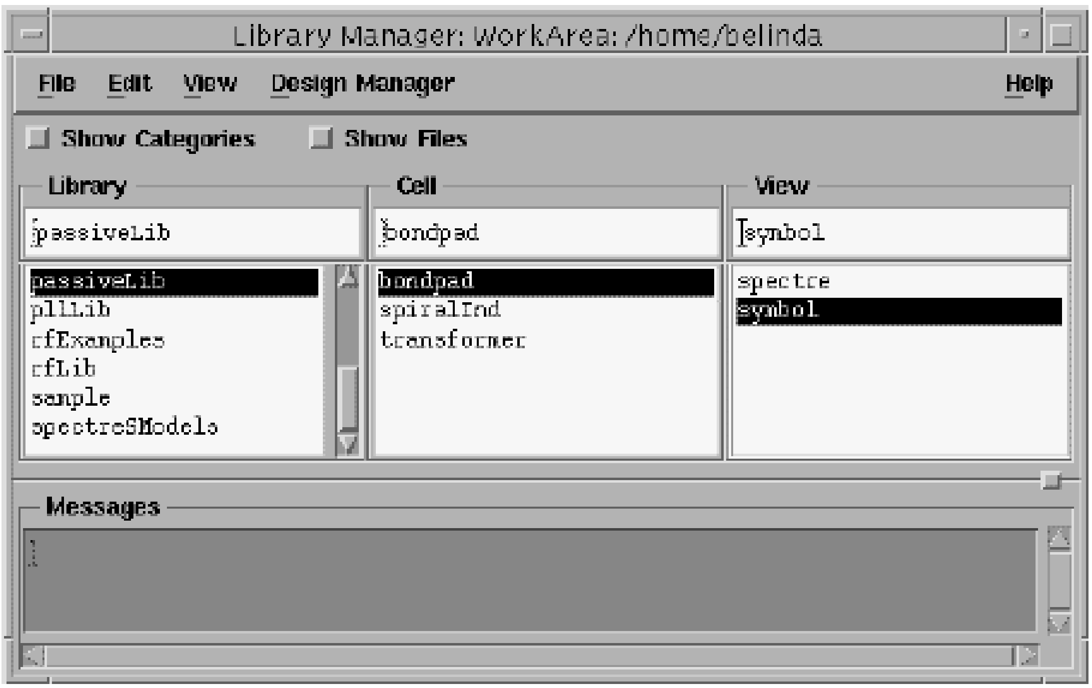
   The Library Browser – Add Instance form appears.

3. In the Library Browser – Add Instance form, do the following:

a. Highlight *passiveLib* for *Library*.

b. Highlight *bondpad* for *Cell*.

c. Highlight *symbol* for *View*



The Add Instance form changes to let you specify values for the bondpad.

4. In the Add Instance form, do the following:

a. Choose *Octagonal* for *Pad shape*.

b. Type the complete path to the process file in *Process file name*.

c. Type 2 for *Number of pads*.

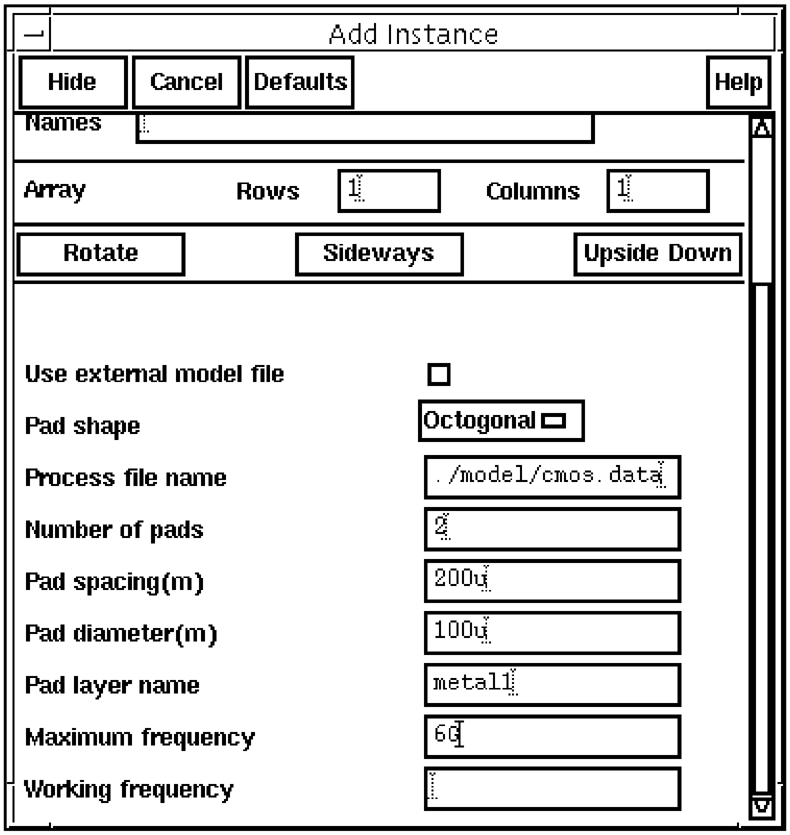
d. Type `200u` for *Pad spacing(m)*.

e. Type `100u` for *Pad diameter(m)*.

f. Type `metal1` for *Pad layer name*.

g. Type `6G` for *Maximum frequency*.

The completed form looks like this.



An instance of the bondpad is now attached to the cursor if you place the cursor in the Schematic window.

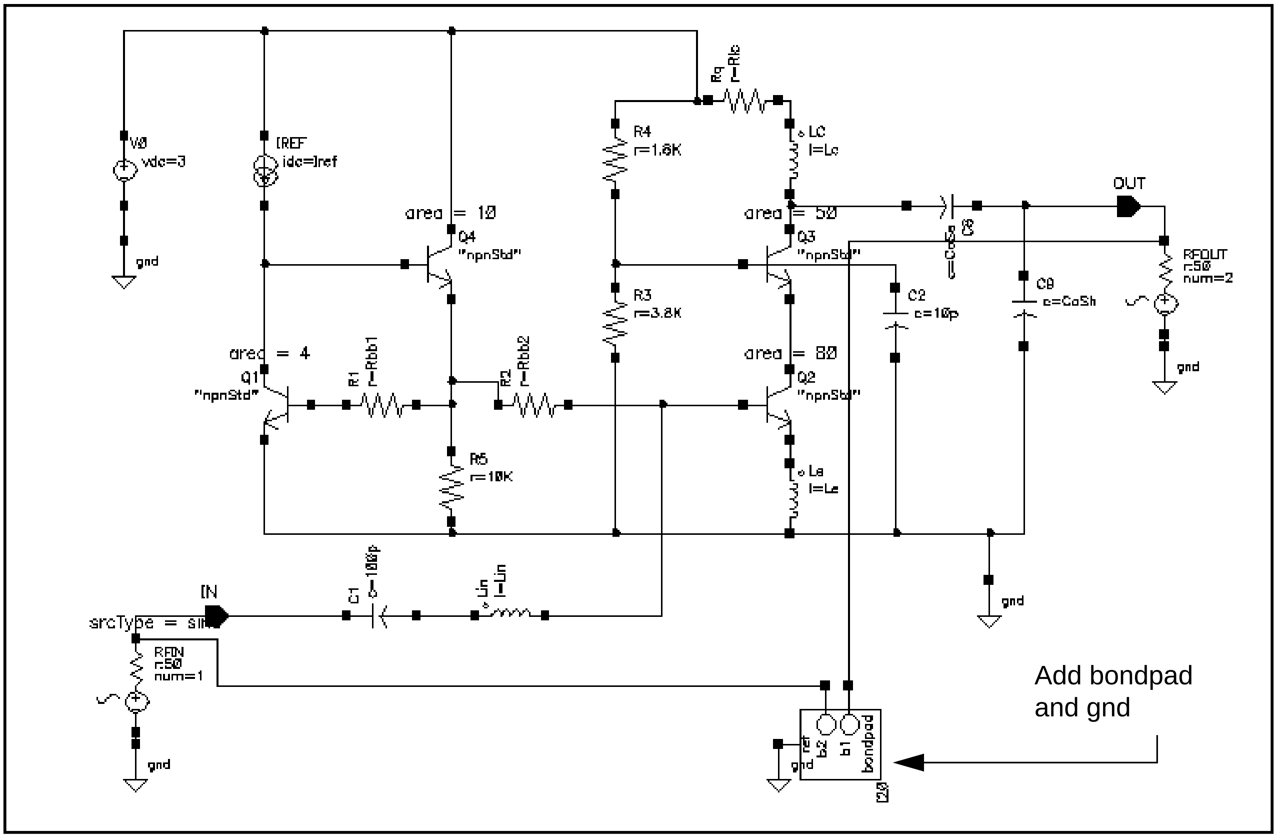
5. In the Schematic window, do the following:

a. Place an instance of the bondpad in the Schematic window.

b. Make a copy of one of the existing ground symbols in the schematic.

Note: If you need assistance with techniques for editing a schematic, see the Virtuoso® Schematic Composer User Guide.

6. In the Schematic window, add wires to connect the components so the final schematic looks like this.



7. In the Schematic window, choose *Design – Check and Save.*

8. In the Simulation window, choose *Simulation – Netlist and Run*.

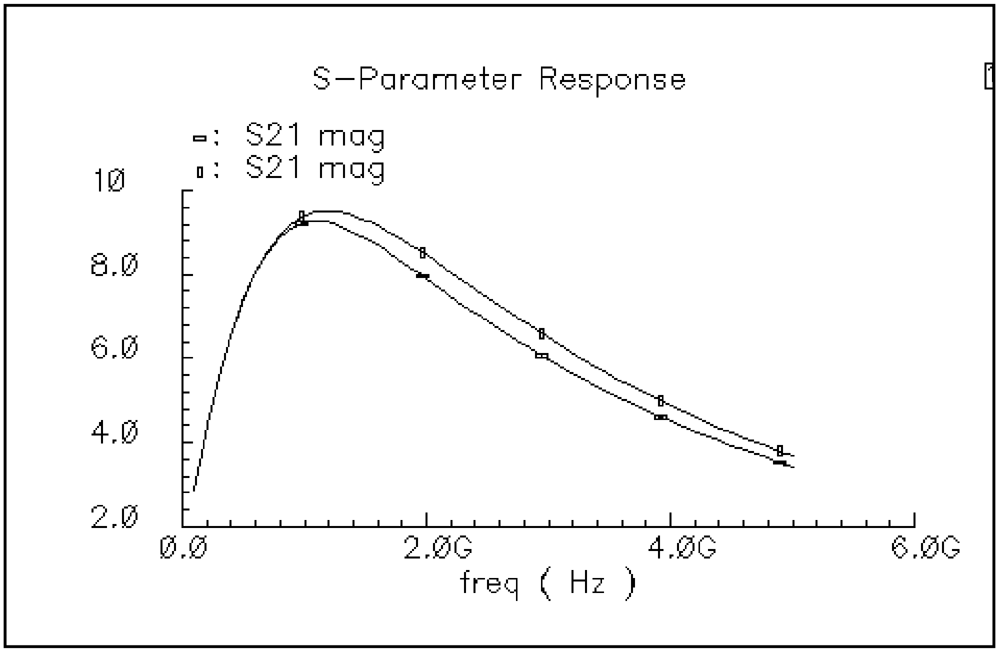
   The simulation is rerun with the bondpad you added to the schematic.

9. In the S-Parameters Results form, do the following:

   a. Highlight *Append* for *Plot Mode*.

   b. Click on *S21*.

The new plot is appended to the Waveform window and the window now looks like this.



The waveform plot shows the comparative results. At 2GHz, the difference of gains is about 8% and the effects are more prominent when frequency is high. Also, the peak gain, tuned by the LC matching at the input, is shifted higher from 1GHz because of the capacitive effects of the input pad.

References for the Bondpad Section

[1] Jinsong Zhao, Wayne Dai, Sharad Kapur, David Long, “Efficient Three-Dimensional Extraction Based on Static and Full-Wave Layered Green’s Functions”, 35th Design Automation Conference.

[2] Sharad Kapur, David Long, “IES3: A Fast Integral Equation Solver for Efficient 3-Dimensional Extractions”, 37th International Conference on Computer Aided Design.

[3] Bonding pad models for silicon VLSI technologies and their effects on the noise figure of RF NPNs, Natalio Camilleri etc, pp. 225-228, IEEE 1994 Microwave and Millimeter-wave Monolithic Circuits Symposium.

Modeling Planar Transformers

The on-chip passive component modelers simulate spiral inductors, rectangular spiral planar transformers, and bonding pads for RF applications. The modelers have an extraction engine based on EM (Electromagnetics) techniques. The analog design environment calls this engine using a command interface.

There are two use models for the planar transformer modeler.

- One is to use the transformer modeler in the schematic flow. You place a transformer symbol in your schematic design. You then change the CDF parameters that specify the input parameters for the transformer in the Edit Properties form. You extract the model at netlisting time. If you netlist a circuit before you simulate, the extraction tool is automatically invoked and added or changed parameters are inserted into the netlist.

- The other use model is to launch the transformer modeler using the *Tools* menu in the Simulation window. The transformer model is extracted from the info you enter in the Transformer modeler GUI and written to the specific model directory. The electric parameters of the components are calculated and displayed on the Transformer modeler GUI form.

  You can include the extracted model in Spectre/SpectreRF simulations; you can use multiple copies of the extracted model if necessary, thereby avoiding long extraction times at netlisting. This approach generates only the internal model called a reduced order model or ROM that is stored in a file. You can use this ROM file in CDF form to avoid unnecessarily generating the model during netlisting. The model is only generated during netlisting procedure when the model is new or the parameters are changed.

The current fitting algorithm still has not completely solved the passivity problem. Rarely, a non-passive model might cause simulation failure because of strong positive feedback within a circuit (such as an oscillator). When this happens, you can use a narrow-band model that is accurate at the specified working frequency.

Process File

Initially, you must create a process file to describe the vertical profile of the process. These parameters will be used in extraction. The file format is the same as that used for spiral inductors. For more information about creating the file, see “Process File Preparation” on page 845.

Symbols and CDF Parameters for Planar Transformers

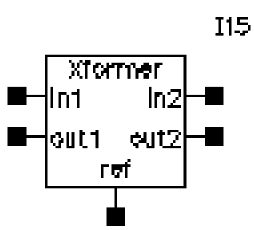
The spiral inductors, bonding pads and planar transformers are stored in the *passiveLib* library. The name of the planar transformer component is *transformer*.

You can specify data for the transformer component in either the CDF form from the Schematic window or using the *Tools—RF—Transformer Modeler* menu in the Simulation window. For more information about the procedure using the RF menu, see "Spiral Inductor Modeling under the RF Sub-Menu" on page 857. The transformer procedures are similar to those for spiral inductors.

Figure 11-19 shows the symbol for a transformer. The transformer symbol is a 5-port block. Since this release is restricted to the rectangular spiral planar transformer which consists of two identical spiral inductors interwound together, the procedures for it's modeling, mainly the CDF form and the RF Sub-Menu GUI, are similar to that of the Spiral Inductor. Only information different from that for the spiral inductor will be emphasized in this section.

- *In1* and *Out1* are for the primary winding
- *In2* and *Out2* are for the secondary winding
- *ref* for the reference

Figure 11-19 Symbol for the Transformer



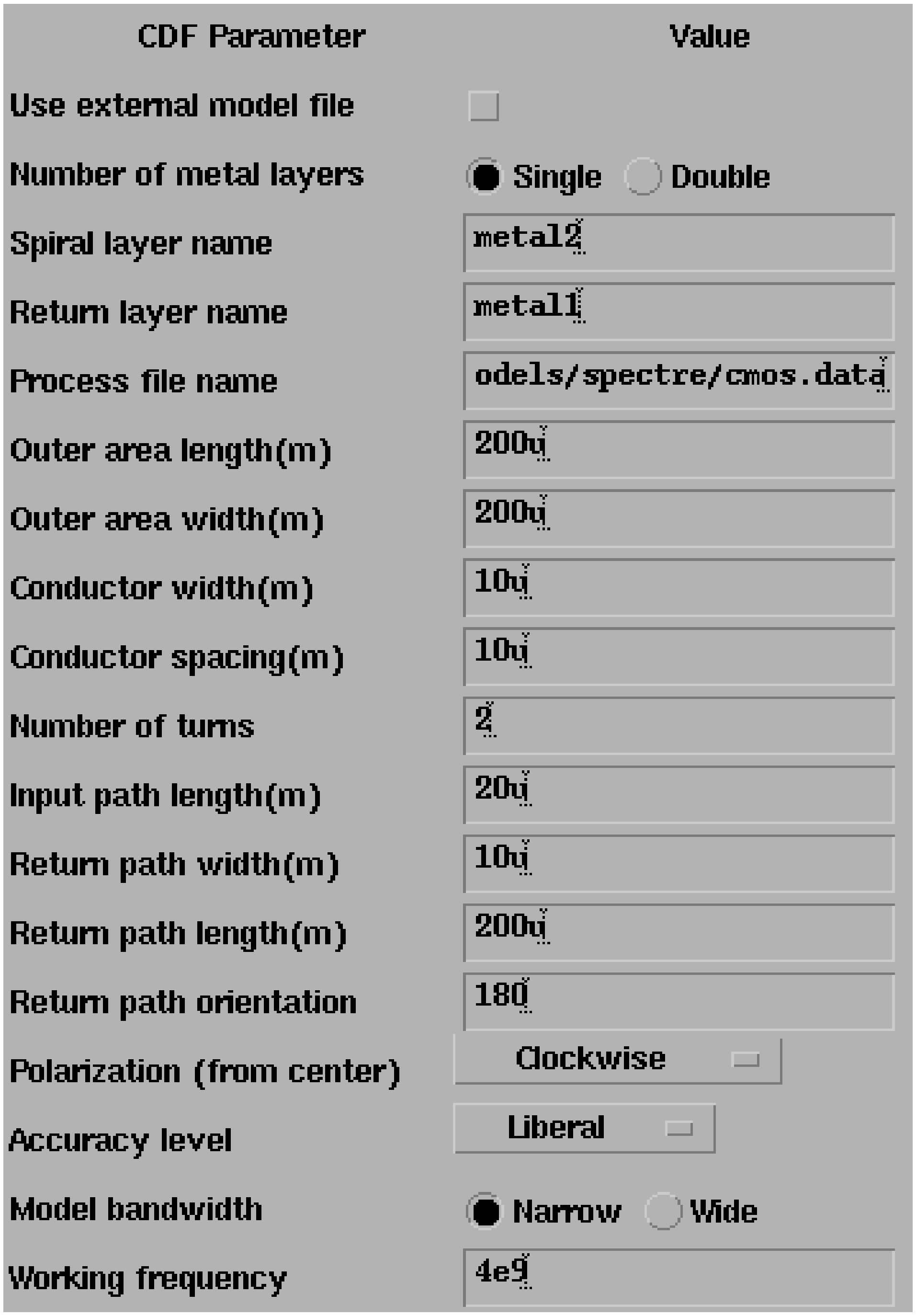
➤ In the Schematic window, choose *Add—Instance* to display the Add Instance form.

In the Add Instance form, at the top

- `passiveLib` displays in the *Library* field
- `transformer` displays in the *Cell* field
- `symbol` displays in the *View* field

The CDF parameters for the transformer element appear at the bottom of the form as shown in Figure 11-20. Notice that there is no shape option on the CDF form as the Transformer is always rectangular.

Figure 11-20 CDF for Transformer Models

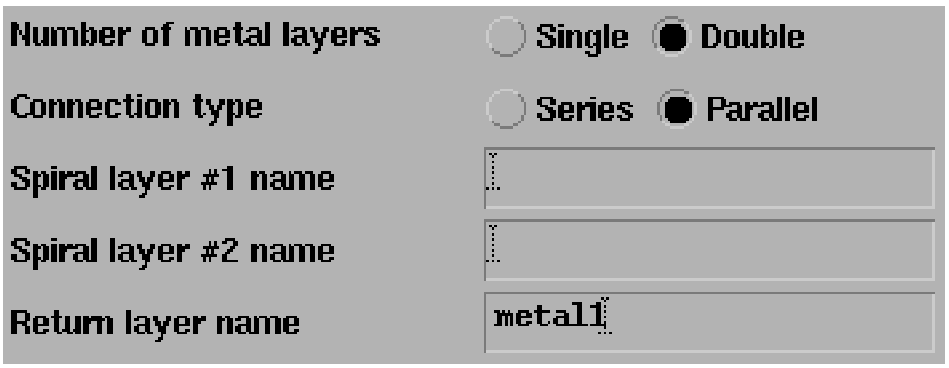


- ***Use external model file***: This button lets you use an externally-generated planar transformer model file directly in a simulation.

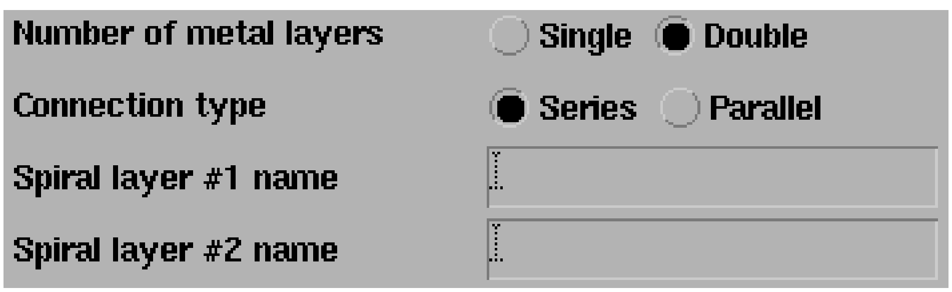
- ***Number of Metal Layers***: Possible choices are *Single* or *Double* for Single Spiral layer and Double Spiral layer transformers. For Double Layer transformers, the two Spiral Layers can have a *Connection type* of *Parallel* or *Series*.
- ***Spiral Layer Names***: The names of the layers the spirals sit in that you specify in the process file (*Spiral layer #1 name* and *Spiral layer #2 name*).
- ***Return layer name***: The name of the layer the return path sits in. For double-layer series inductors, this entry doesn't appear because the bottom spiral layer serves as the return path.
- ***Process file name***: The name of the process file. You must specify the full path.
- ***Outer area length (m)***: The length of the outer dimension of the planar transformer.
- ***Outer area width***: The width of the outer area for the planar transformer.
- ***Conductor width***: The width of the planar transformer.
- ***Conductor spacing***: The gap between two neighboring turns.
- ***Number of turns***: The number of spiral turns. The number can be fractional.
- ***Input path length***: The length of metal extending beyond the outermost segment of the spiral.
- ***Return path width***: The width of the return path.
- ***Return path length***: The length of the return path.
- ***Return path orientation***: The angle between the return path and the innermost segment of the spiral, measured in degrees.
- ***Polarization (from center)***: Possible choices are *Clockwise* or *Counterclockwise*, looking from the top down.
- ***Accuracy level***: Determines the mesh density in capacitance and inductance extraction.
- ***Model bandwidth***: Possible choices are *Narrow* bandwidth model and *Wide* bandwidth model.
- ***Working frequency***: The actual working frequency of the spiral used in the extraction of parasitics parameters.
- ***Maximum frequency***: The maximum frequency at which parasitic parameters are extracted for wide bandwidth models.

Double-Layered Spiral Inductors

When you highlight *Double* for *Number of metal layers*, the CDF form changes to add a new entry, *Connection type*. In addition, the *Spiral layer name* field is replaced by two fields: *Spiral layer #1 name* and *Spiral layer #2 name.* If you highlight *Parallel* for the *Connection type*, the *Return layer name* field remains.



If you highlight *Series* for the *Connection type*, the *Return layer name* field is removed and one of the spiral layers functions as the return path.



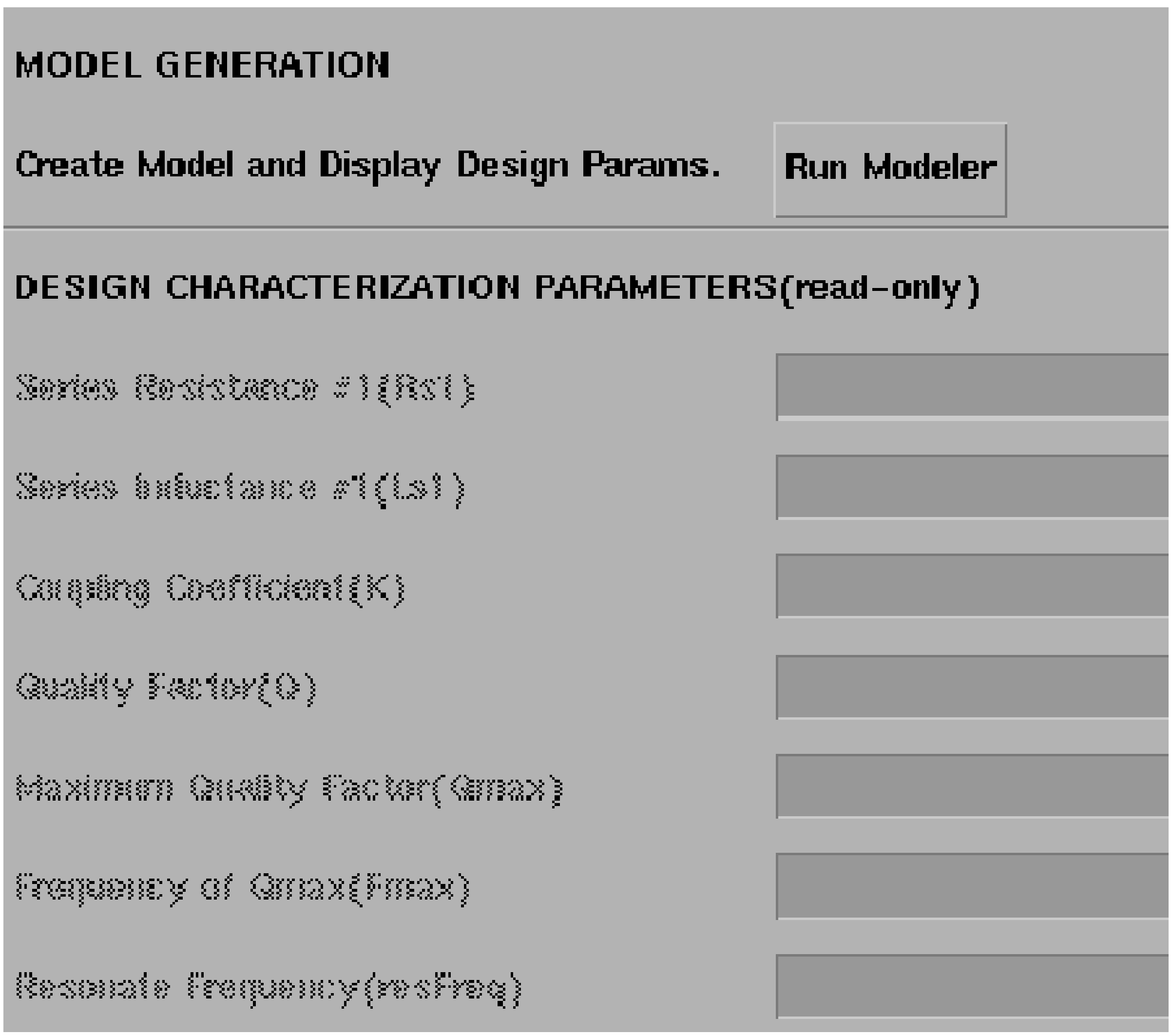
Process File and Parameter Update

If you specify and apply new parameters, the model extraction is performed at netlisting time. If you do not change parameters, the previous model file is used in simulation and the model creation step omitted. If any parameters change, the inductor model is recreated. Because the process file name is the only reference, if you change the process file contents, the model is not updated unless you specify a new file name. Consequently, we recommend that you specify a new process file name for each new set of parameters.

Planar Transformer Modeling Under the RF Sub-Menu

You can launch the Planar Transformer Modeler without going into the schematic by choosing *Tools—RF—Planar Transformer Modeler* in the Simulation window. The Planar Transformer Modeler GUI accessed using the RF sub-menu is similar to the CDF form for transformers. The major difference is the addition of directory information and the design characterization parameters at the bottom of the GUI as shown in Figure 11-21.

Figure 11-21 Bottom of the Planar Transformer Modeling GUI



Setting Up and Running a Simulation

After you specify the physical geometries of the planar transformer such as length, width, spacing, and frequency information, you must specify the run directory where the auxiliary files created by the modeler engine are located. You must also specify the model directory where the extracted model file is saved. You launch the modeler engine by clicking on *Run Modeler*. After the extraction, the design characterization parameters are displayed at the bottom of the form. The bottom half of the UI is shown in Figure 11-21 on page 897.

List of Design Parameters

In Figure 11-21 on page 897, seven parameters for the planar transformer are displayed at the bottom of the GUI. Although the transformer has two windings, the design parameters are calculated with one winding grounded at the two terminals. Since the two windings are identical, only the series resistance and inductance of one winding are displayed. All the parameters displayed, except the coupling coefficient, are that of a loaded spiral inductor. the Coupling coefficient is defined in the following list.

- ***Series Resistance #1 (Rs1)*****:** The AC resistance of the spiral at the working frequency with the skin-depth effect included.
- ***Series Inductance #1 (Ls1)*****:** The total inductance of the spiral inductor with the skin depth effect included.
- ***Coupling Coefficient (K)*****:** The ratio of the mutual inductance between the two windings divided by the self inductance of each winding (K= M/L).
- ***Quality Factor (Q)*****:** The Q-factor at the working frequency
- ***Maximum Quality Factor (Qmax)*****:** The maximum quality factor of the inductor within the 10 MHz to 20 GHz frequency range.
- ***Frequency of Qmax (Fmax)*****:** The frequency at which *Qmax* occurs
- ***Resonant Frequency resFreq)*****:** The frequency where *Q*=0. The search is from 10 MHz to 20 GHz. If no resonant frequency is found, a value of -1 is displayed.

These parameter calculations are based on the equivalent circuit model. Because of the difference between the wide-bandwidth and narrow-bandwidth models, the slight difference in the design parameters between the two models is to be expected.

Including the Inductor Model Files in the Simulation

After model extraction, a netlist file that describes the equivalent circuit model of the inductor is written to the model directory you specify. You can use this file directly with the *Use external model file* option on the transformer CDF form. To do this, perform the following steps:

1. In the Simulation window of the analog design environment, choose *Setup —Simulation Files*.

   The Simulation Files Setup form appears.

2. In the Simulation Files Setup form, specify the full path to the model definition file.

Equivalent Circuit Models

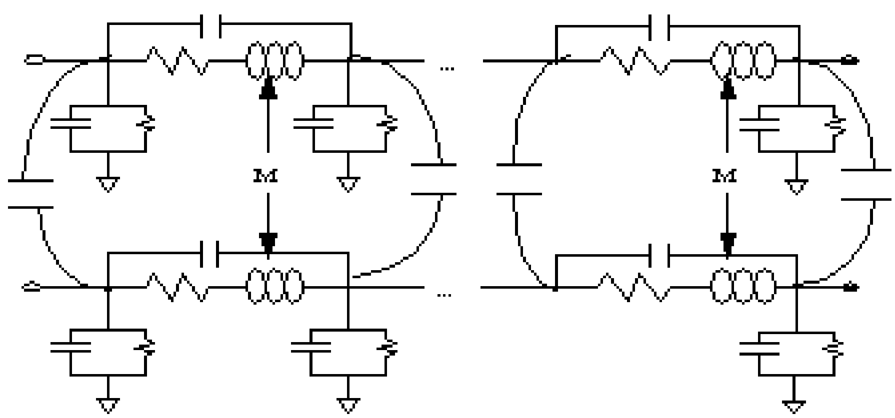
This section discusses the two types of equivalent circuit models:

- Narrow bandwidth models
- Wide bandwidth models

Narrow Bandwidth Model

This narrow bandwidth model, as shown in Figure 11-22, is constructed directly from the RLCG matrixes obtained at the working frequency. All the elements are fully coupled through capacitors and mutual inductance.

Figure 11-22 Narrow-Bandwidth Model for a Planar Transformer



Because the RLCG parameters are functions of frequency, the above model is valid only in the vicinity of the working frequency. It is therefore a narrow-bandwidth model. Expect large simulation error for frequency components outside of the bandwidth. Because this model only requires the extraction of RLCG parameters at a single frequency, the extraction is faster. The resulting model is also guaranteed to be passive and stable.

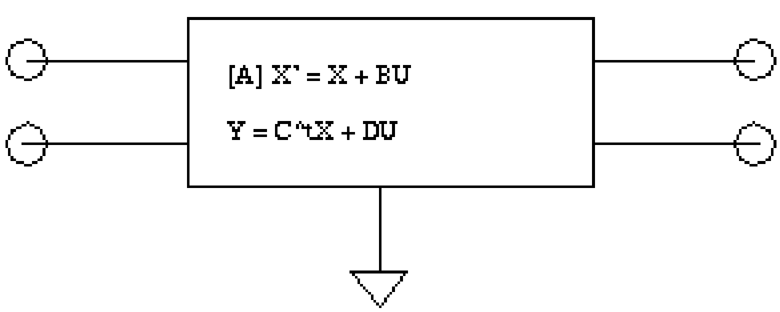
Working frequency selection and model accuracy

Because the narrow-bandwidth model is built using parameters extracted at the working frequency, the model is accurate only in the vicinity of the working frequency. The error increases as the frequency moves farther from the working frequency. Because the error is caused mostly by the large variation of the RLCG parameters with frequency, the higher the frequency, the narrower the bandwidth. If the skin depth or the substrate loss is not significant at the working frequency, the narrow-bandwidth model is rather accurate from DC up to the working frequency.

Wide Bandwidth Model

The wide-bandwidth model is designed to work over a wide frequency range. Unlike the narrow-bandwidth model, the RLCG parameters for the wide-bandwidth model are extracted over a frequency range. You specify the maximum frequency, and the minimum frequency is set to 10 MHz. Modeling the planar transformer as a two-port system, permits the S-parameters of the two-port to be calculated at each frequency point. The resulting list of S-parameter matrixes are then sent to a rational fitting module to create a wide-bandwidth nport component. Figure 11-23 shows the mathematical representation of the wide bandwidth model for the transformer.

Figure 11-23 Planar Transformer Wide-Bandwidth Model



Passivity of the Wide Bandwidth Model

The passivity of the wide-bandwidth models is tested over a frequency range by checking the energy conservation for each port at a set of sample frequencies. The default frequency range is from DC to 30GHz, which is normally wide enough for RF applications. Because the model is only guaranteed to work within this frequency range, it might not work for a simulation outside of the tested range. A mechanism is provided to change the test range.

Set maximum Frequency for Passivity Check

You can reset the maximum frequency for the passivity checking by setting the environment variable *maxFreqForPassivityTest.*

```
setenv maxFreqForPassivityTest 50e9
```

Automatic Model Switch

To avoid simulation failure caused by non-passivity of the wide-bandwidth model, the modeler automatically switches to a narrow-bandwidth model if the passivity test fails. You can prevent this automatic switching by setting an environment variable in the window where you start *icms*.

```
setenv pcmModelAutoSwitch OFF
```

If you turn off the switch, the wide-bandwidth model is created and a warning message issued. Spectre uses the model in simulation even though it is not passive.

Accuracy Analysis

The *Accuracy level* entry on both the CDF and the RF sub-menu UI forms controls the mesh density for both the capacitance and inductance extraction. For capacitance, the number of subsections per width is three, five, or seven. These values correspond to the *Liberal*, *Moderate* and *Conservative* accuracy levels. Inductance calculation is similarly correlated with the accuracy level. Three, five, or seven filaments are used in the transverse direction to the current, corresponding to the three accuracy levels. The error is less than 3% at the *Liberal* level, and about 1% at the *Conservative* level.

12

Methods for Top-Down RF System Design

Methods for Top Down RF System Design

This chapter describes a methodology for designing analog RF subsystems that fit into larger DSP systems. In particular, this chapter describes how to use a canonical set of top-down behavioral baseband models for exploring RF architectures in the analog design environment. These models come form the following categories in *rfLib*

- Category *top_dwnBB* contains models of common RF function blocks.
  - The default view of each model is the baseband view (called *veriloga*).
  - All models in this category also have a differential passband view (called *veriloga_PB*).

  The only exceptions are the BB_loss and VGA_BB models. The BB_loss model is meant only for baseband analysis. The VGA_BB is an engineering release of a variable gain amplifier. It is not documented or supported in this release and is subject to change in future releases.

- Category *top_dwnPB* contains single-ended passband versions of the baseband models.

- Category *measurement* contains the instrumentation block and baseband signal generator models used to make RF measurements. These elements are not part of an RF architecture. They simply facilitate RF measurements and diagnostics.

- Category *testbenches* contains the test circuits used in this chapter to define the model specifications in the *rfLib*. Where possible, the models are specified in terms of standard RF measurements. The most precise way to describe a measurement is with a test circuit, set up instructions, and sample measurements. The circuits in the testbenches category serve that purpose

See Appendix D, “The RF Library” for more information about the *rfLib* and detailed descriptions of the models it contains.

These models provide RF designers with a fast method to map RF system specifications into detailed RF designs. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs of individual RF system components.

Baseband models are behavioral models and all behavioral models sacrifice some accuracy for increased simulation speed. Such sacrifices are usually acceptable in architectural studies because many implementation-dependent details do not affect high level decisions. The modeling approach taken in top-down design is to simulate only those effects that drive the decisions at hand.

Baseband modeling in no way replaces passband modeling. Some effects missed by equivalent baseband models can affect high level decisions. However, the application of baseband models early and passband models later minimizes the number of slow simulations needed at the lower levels of design abstraction. Baseband models help you to quickly weed out designs that would surely fail tests simulated with passband models.

The success of a modeling approach to top-down design hinges on knowing how the models fit into the design flow and knowing exactly what each modeling parameter means. This chapter has two goals:

- To describe the top-down design flow, from a modeling perspective, for baseband modeling
- To define, as clearly and concisely as possible, the parameters that specify the models

Top-Down Design of RF Systems

Ideally, the digital signal processing, or DSP, team specifies an RF subsystem that fits snugly into the DSP system. A *snug fit* means that

- The specified RF subsystem does exactly what it needs to do at the lowest possible cost
- A functional specification exists that describes requirements for the RF subsystem

In a top down design flow like the one shown in Figure Figure 12-1 on page 906, the DSP team writes a functional specification for an RF subsystem that has not yet been designed. The functional specification describes what the RF subsystem should and should not do without describing how to build the RF subsystem.

The functional specification supplied by the DSP team describes the RF subsystem at the highest possible level of abstraction. At this point behavioral models can be specified rather than measured. This early in the design cycle, the functional specification might well be incomplete or inconsistent. A good top-down design flow can detect problems, such as omissions and inconsistencies in the design, early in the design cycle when they are easier and less expensive to fix. Problems detected later in the design cycle can be much more costly and very difficult to resolve.

Using the functional specification supplied by the DSP team and the behavioral baseband models from *rfLib*, the RF system designers can easily explore RF architectures in the analog design environment. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. By switching to the passband views of the baseband models, the RF design team maps DSP measurements to RF measurements. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs for individual system components.

Using the functional specification and exploring and testing with the baseband and passband models, the RF team can efficiently create a detailed design specification that fully describes the RF subsystem. The design specification can include detailed instructions for building the RF subsystem. At this stage of the design cycle, everything that is known about the design is described at the lowest level of abstraction.

You can now extract behavioral models of a detailed design from simulated measurements. The problem remains that detailed designs usually do not exist until the project is complete. To jump directly to a detailed design implies that the design flow is bottom up. Bottom up flows are important in many projects, but not in all.

DSP and RF designers sometimes have trouble communicating through specifications because the two groups deal with different metrics. For example, DSP designers deal with *bit error rates* and *error vector magnitude statistics* whereas RF designers deal with *intercept points* and *noise figures*.

The new models described here are designed to help RF system designers in two ways.

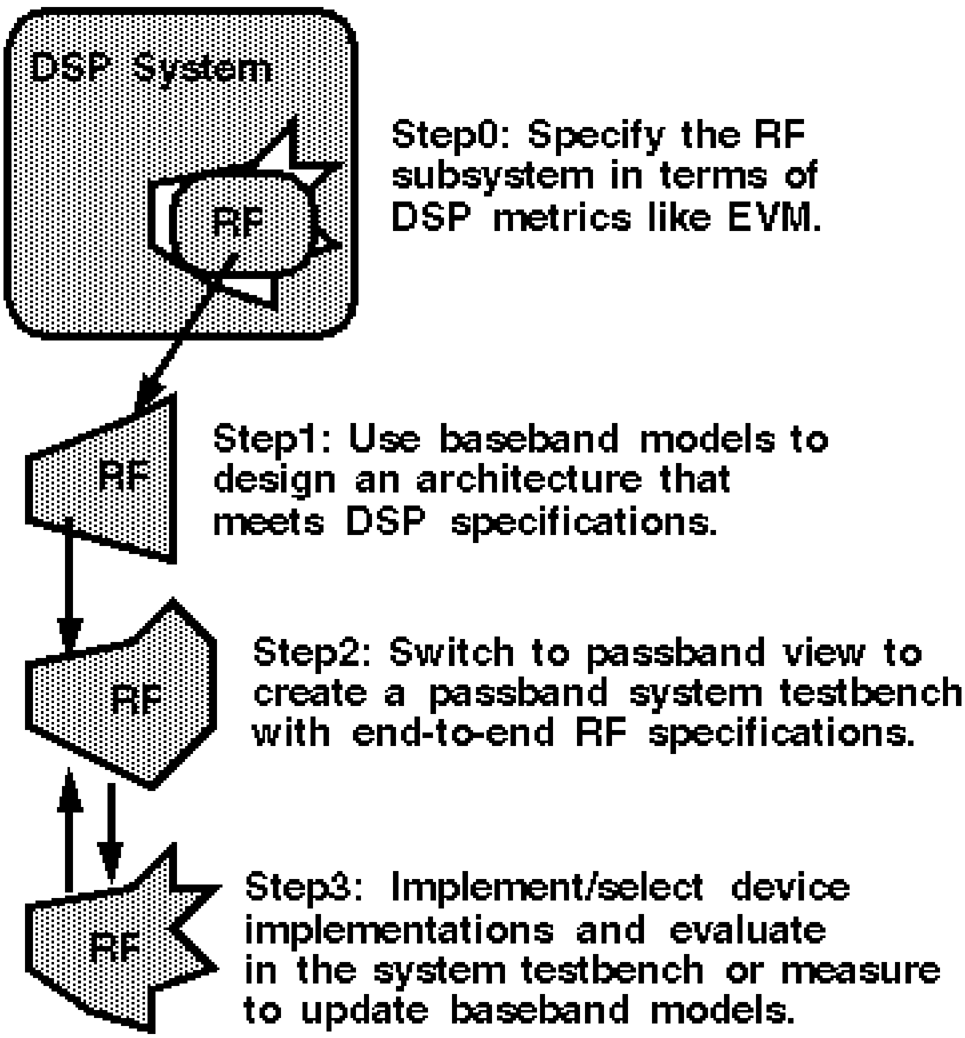
- First, the baseband models enable RF system designers to quickly explore the RF architectural space, as specified by the DSP metrics, while letting the RF engineers specify the RF system components with RF metrics. The circuit implementations of the RF system components are easier to design and test when the components are specified with standard RF metrics.

- Second, the baseband models can be switched quickly to a passband views where the RF system model can generate end-to-end RF metrics. With end-to-end metrics, the new view can quickly simulate how the detailed design of a particular RF system component affects end-to-end performance.

Use Model for Top Down Design

The following steps outline the RF design process with focus on the early phases of the design as illustrated in Figure Figure 12-1 on page 906.

Figure 12-1 The Top Down Design Flow and Use Model



Specify the RF Subsystem in Terms of DSP Metrics

Before you begin the RF subsystem top-down design flow, the DSP design team should completely specify the RF subsystem in terms of DSP metrics. This preliminary step distinguishes the end of the DSP design flow from the beginning of the RF top down design flow and formally hands-off the RF subsystem design specifications to the RF design team.

Explore Candidate Architectures with Baseband Models

The first step in top down RF design is to select a candidate RF architecture. An RF architecture is a set of interconnected RF function blocks that, taken together, describe how a receiver or transmitter operates. You specify each function block in terms of standard RF metrics such as IP3, gain, bandwidth, and noise figure.

The models you use early in the design cycle as you explore candidate RF architectures must run fast. Each simulations can span hundreds of symbols and each symbol can easily span thousands of RF carrier cycles. The space defined by the function block specifications in each candidate architecture is far too vast to explore with slow, highly precise models. Models used for architectural exploration must quickly reduce the design space down to a size that can be explored with more precision.

The most efficient models for architectural exploration suppress the RF (IF) carrier and are called *baseband* models. In contrast, the *passband* models (introduced in the next step) do not suppress carriers.

You can use the Circuit Optimizer during architectural exploration to help balance the function block specifications for a candidate architecture. For example, you can use the Circuit Optimizer to minimize RMS EVM while ensuring that other measurements stay within acceptable limits.

When you have determined the nominal specifications for each function block, you must put tolerances around them. In the analog world *specifications without tolerances are meaningless*. The tolerance space is usually explored with some mix of experience, feasibility, a variety of analyses, and outright arbitrary decisions.

There are several ways that you can use the baseband models to test candidate tolerances as well as to determine some tolerances analytically.

- One way to test a candidate set of tolerances is to run a Monte Carlo analysis on the metric of interest, like RMS, EVM, or signal-to-noise ratio (SNR).
- Another approach is to use the Circuit Optimizer *in reverse*, as a de-optimizer, to determine worst case performance.
- Yet another approach is to compute each tolerance separately from a parametric plot. When you have determined all but 2 or 3 tolerances, you can use a multidimensional parametric analysis to map out the performance space and easily identify the remaining tolerances.

Switch to Passband Models and Create an RF System Testbench

The second step is to create a passband view of the system model.

The passband system model performs two functions:

- Confirms that the filters perform as expected.
- Creates an end-to-end testbench that you can use to design the individual function blocks.

For computational efficiency during system passband testing, at any one time, model one or two selected function blocks at the device level. Model all other blocks in the system behaviorally using passband models.

It is not practical to use the passband view to assess DSP metrics because Spectre analysis would take too long. Also, you cannot use SpectreRF to reduce run times because the instrumentation blocks have internal state variables and SpectreRF does not allow state variables inside behavioral models. However, you can apply SpectreRF to the passband behavioral view to derive end-to-end RF specifications, if you do it before adding any device-level models to the testbench.

Derive the tolerances by performing the same Monte Carlo analysis or Circuit Optimizer analysis you used to test the function block tolerances in the first step, but this time replace the DSP metrics with end-to-end RF metrics. Once you know how far the end-to-end RF metrics can vary, you can insert a device-level model of a function block into the testbench to see how close it drives the system toward violating a derived end-to-end RF specification.

Implement the Function Blocks with Active and Passive Devices

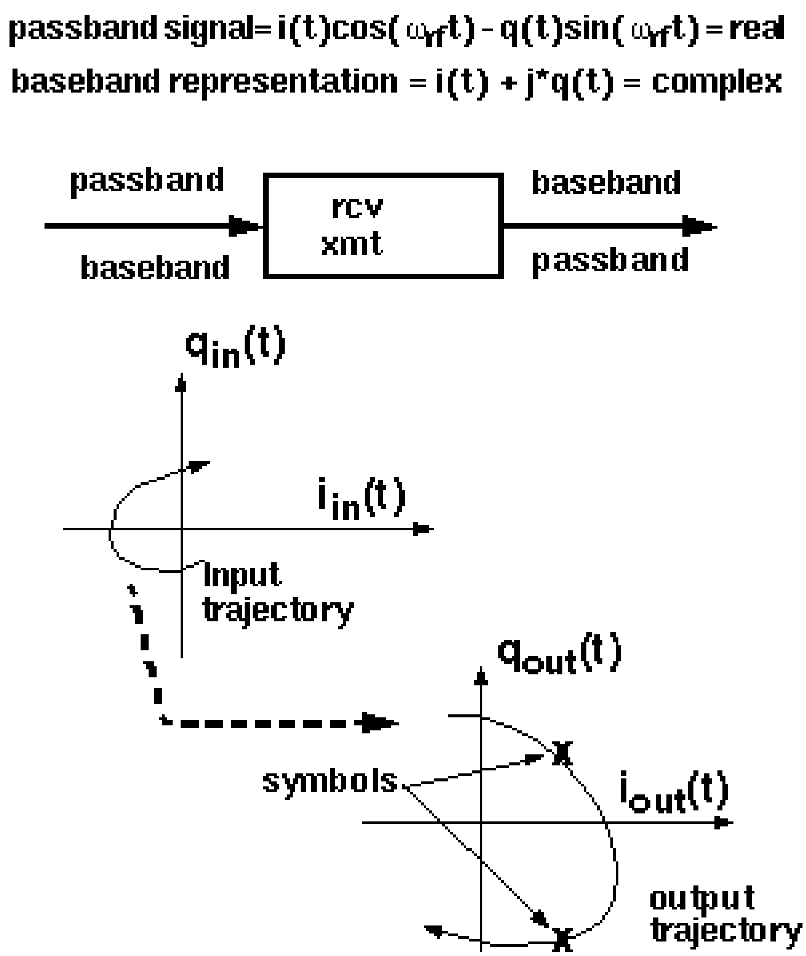
The last step in the top down design process is to implement the function blocks with device models. Since the function blocks are specified in terms of standard RF metrics, you can easily measure the modeling parameters to make sure they fall within the specified tolerances. You can also insert the measured parameters back into the baseband model of the system to check the DSP metrics, or insert the device-level model directly into the passband testbench to check the derived end-to-end RF specifications.

Baseband Modeling

A baseband model for an RF function block simulates what happens to the baseband representation of a signal as it passes through the block. A baseband model maps input baseband signal trajectories into output baseband signal trajectories. If you sample a baseband signal periodically in time and plot the samples in the complex plane, the resulting scatter plot shows the symbol constellation.

Figure 12-2 on page 909 mathematically defines a baseband representation of a passband signal. The *i* and *q* signals are the real and imaginary parts of a complex signal that rides on the two phases of an RF carrier.

Figure 12-2 Baseband Representation of a Passband Signal



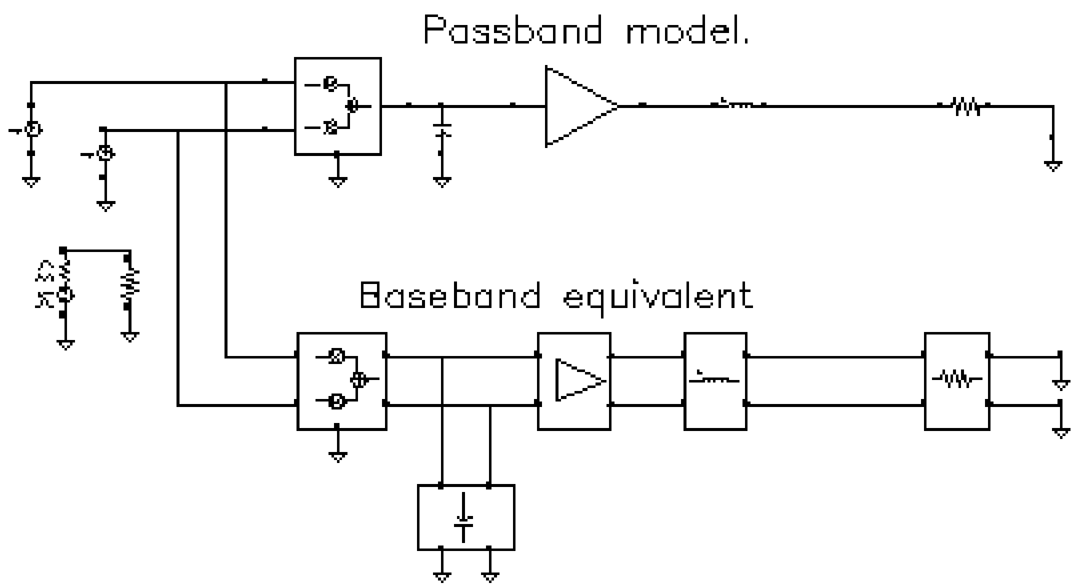
Baseband models simulate only what happens to the carrier fundamental. Consequently, they only account for non-linearities with odd symmetry. Non-linearities with even symmetry produce no output at the carrier fundamental; they affect the carrier fundamental only when cascaded. For example, a second order non-linearity in one block can create a DC offset at it’s output. Upon passing through a subsequent block with another second order non-linearity, the DC offset can mix with the carrier to affect the output carrier fundamental. You should model cascaded blocks producing unfiltered even harmonics as a single baseband model rather than as separate baseband models cascaded together. The non-linearities that most often dominate performance have odd symmetry.

Example Comparing Baseband and Passband Models

The example in this section walks you through an Envelope analysis that illustrates the relationship between baseband and passband models. Following the simulation, you plot the baseband equivalent output signals as computed by the baseband and passband circuits.

The *BB_test_bench* schematic shown in Figure 12-3 on page 910 illustrates the difference between passband and baseband modeling. This circuit is located in the *rfExamples* library.

Figure 12-3 The BB_test_bench Schematic



The *BB_test_bench* circuit shows a passband circuit (across the top of the schematic) and its baseband equivalent circuit (across the bottom of the schematic). The same baseband signals drive both circuits but only the passband circuit mixes the baseband signals up to RF. The power amplifier is not matched to either input or output impedances and both impedances are reactive.

Before you start, perform the setup procedures described in Chapter 3.

Opening the Baseband Test Bench Circuit

1. In the CIW, choose *File – Open*.

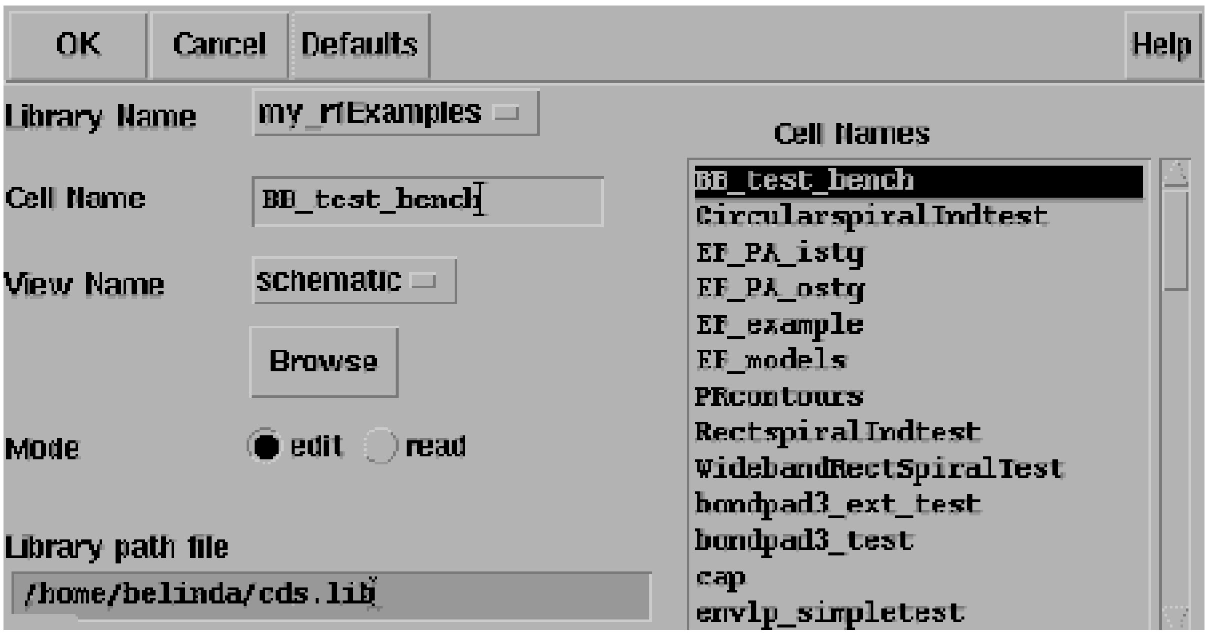
The Open File form appears.

2. In the Open File form,

a. Choose *rfExamples* in the *Library Name* cyclic field. (Choose the editable copy of *rfExamples* you created as described in Chapter 3.)

b. Choose *BB_test_bench* in the *Cell Names* list box. Note that the *View Name* cyclic field displays *Schematic*.

The completed Open File form appears like the one below.

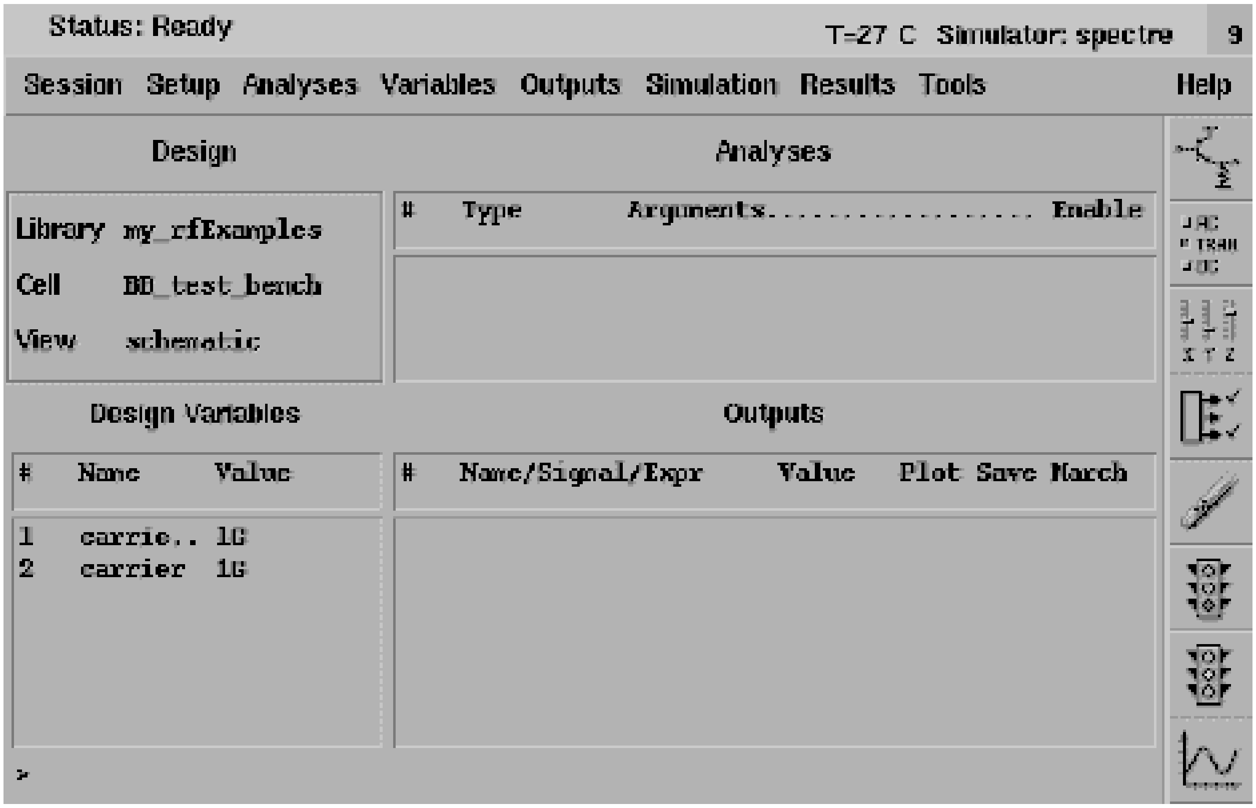


3. Click *OK*.

The Schematic window for the *BB_test_bench* appears.

4. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window opens.



You can also use *Tools – Analog Environment – Simulation* in the CIW to open the Simulation window without opening the design. You can open the design later by choosing *Setup – Design* in the Simulation window and choosing the *BB_test_bench* in the Choosing Design form.

Choosing Simulator Options

1. Choose *Setup – Simulator/Directory/Host* in the Simulation window.

The Simulator/Directory/Host form appears.

2. In the Simulator/Directory/Host form, specify the following:

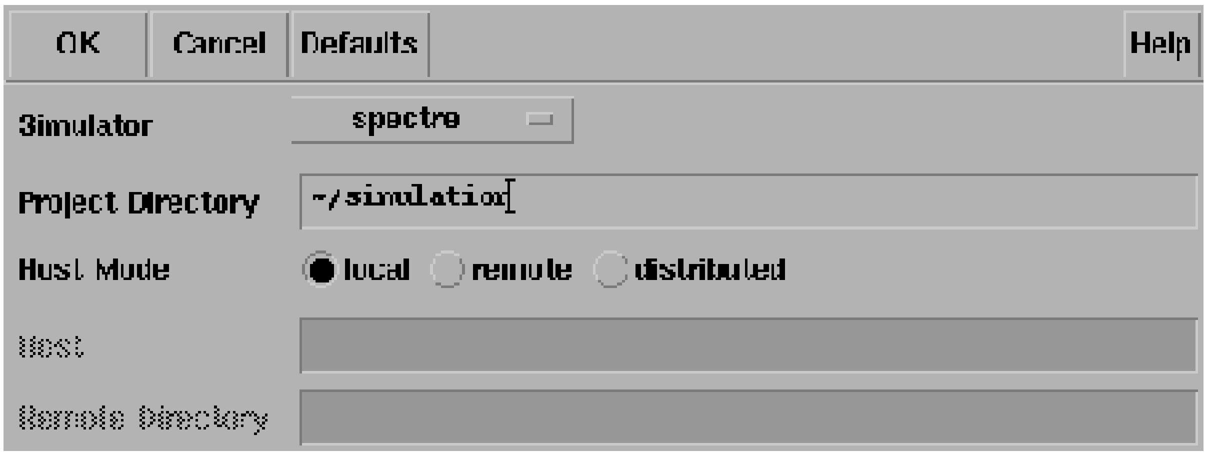
a. Choose *spectre* for the *Simulator*.

b. Type the name of the project directory, if necessary.

c. Highlight the *local* or the *remote* button to specify the *Host Mode*.

For remote simulation, type the name of the host machine and the remote directory in the appropriate fields.

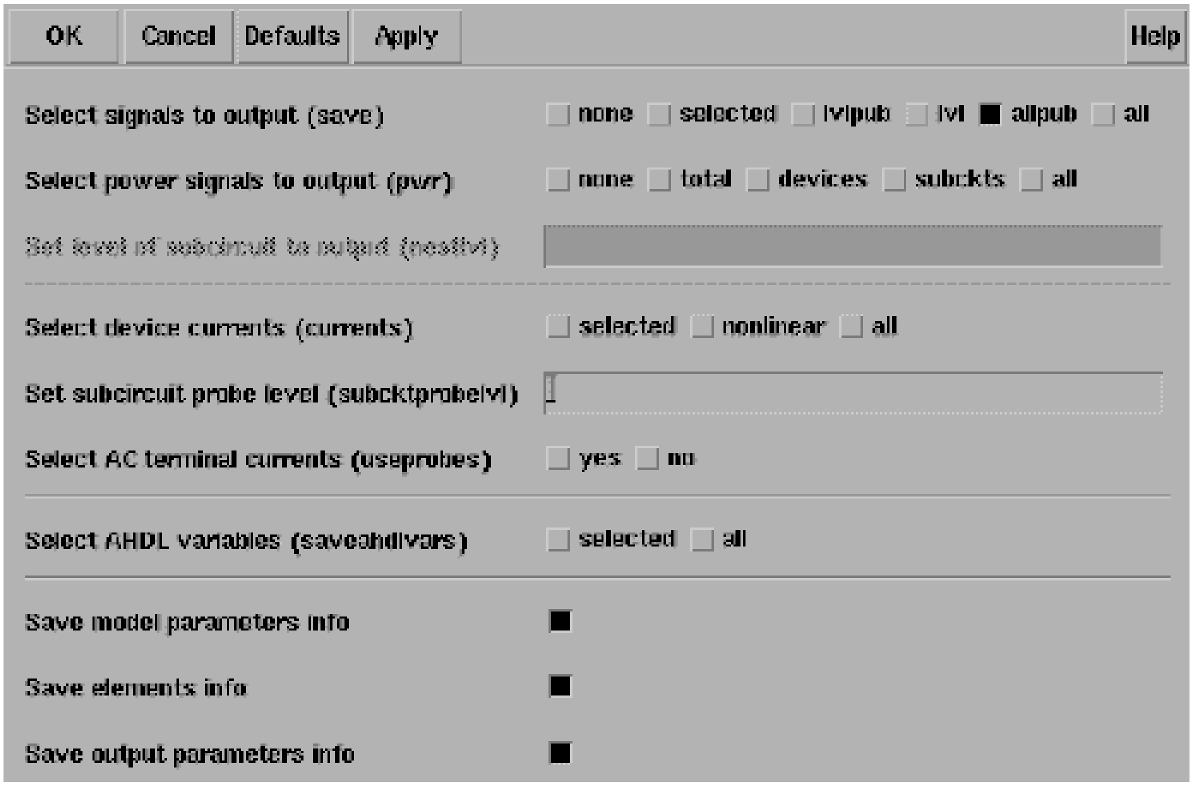
The completed form appears like the one below.



3. In the Simulator/Directory/Host form, click *OK*.
4. In the Simulation window, choose *Outputs – Save All.*

   The Save Options form appears.

5. In the *Select signals to output (save)* section, be sure *allpub* is highlighted.



6. In the Save Options form, click *OK*.

Setting Up Model Libraries

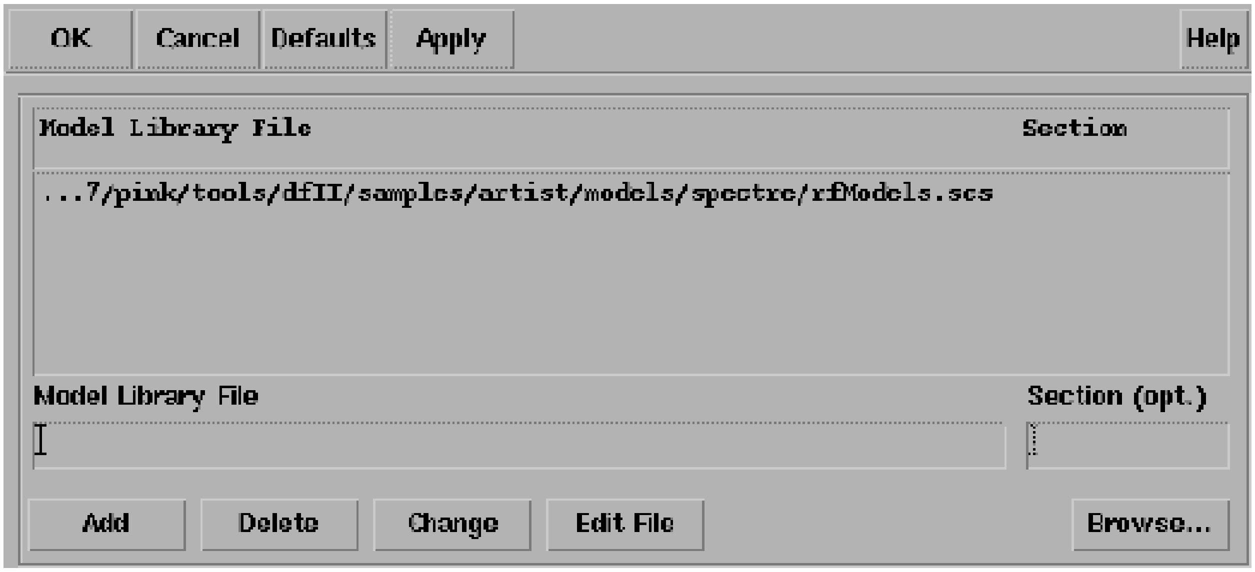
1. In the Simulation window, choose *Setup – Model Libraries*.

   The Model Library Setup form appears.

2. In the *Model Library File* field, type the full path to the model file including the file name, `rfModels.scs.`

3. In the Model Library Setup form, click on *Add*.

The completed form appears like the one below.



4. In the Model Library Setup form, click *OK*.
5. In the Simulation window, use *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not an analysis is enabled.)

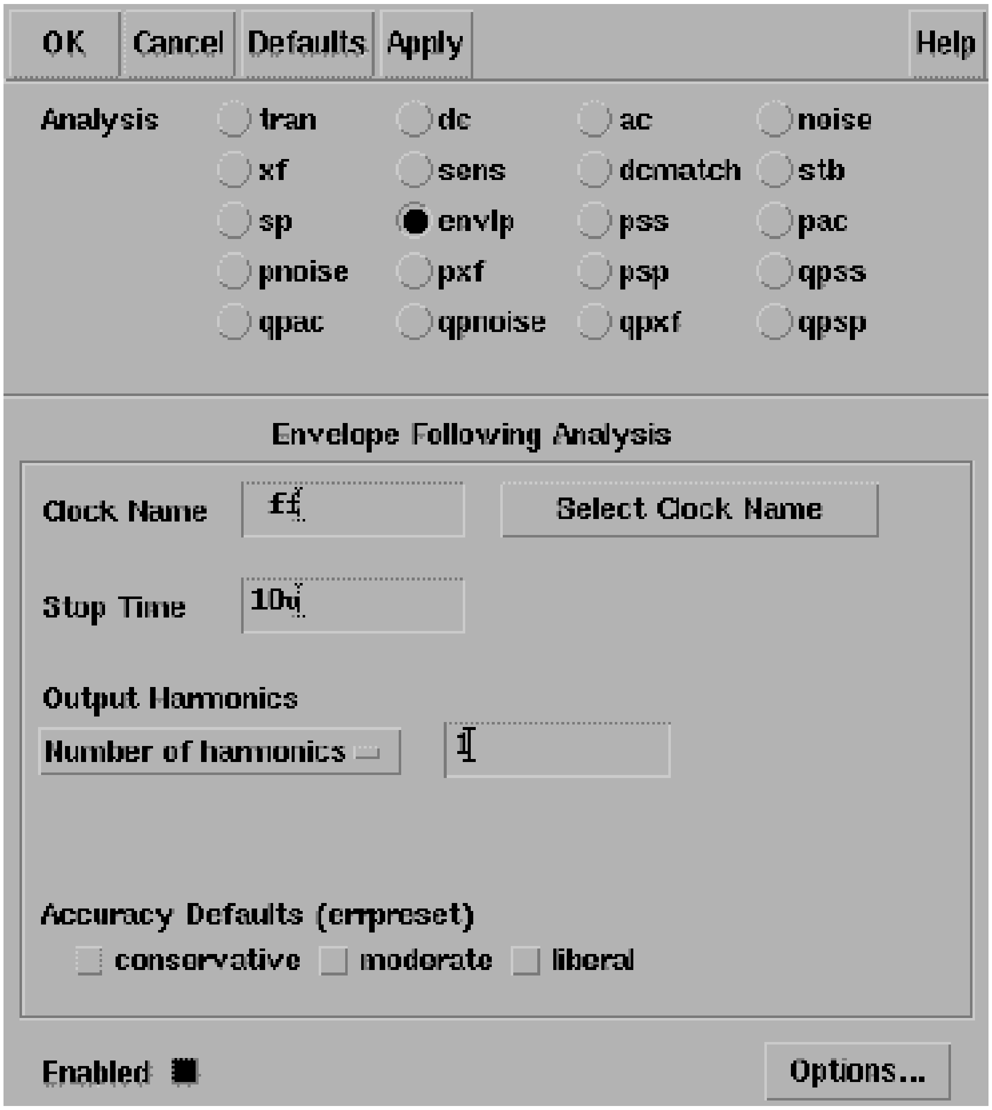
Setting Up the Envelope Analysis

1. Choose *Analyses – Choose* in the Simulation window.

   The Choosing Analyses form appears.

2. In the Choosing Analyses form, click on *envlp.*
   - **a.** Enter *ff* in the *Clock Name* field.
   - **b.** Enter `10u` in the *Stop Time* field.
   - **c.** In the *Output Harmonics* cyclic field, select *Number of harmonics*.
   - **d.** Enter `1` in the *Number of harmonics* field.

The correctly filled out form appears below.



3. In the Choosing Analyses form, click *OK.*

Running the Simulation

1. In the Simulation window, choose *Simulation – Netlist and Run*.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Plotting the Baseband Equivalent Output Signals

1. In the Simulation window, choose *Results-Direct Plot-Main Form*.

   The Direct Plot form appears.

2. In the Direct Plot form, do the following:

   a. Highlight *Replace* for *Plot Mode.*

   b. Highlight *envlp* for *Analysis*.

   c. Highlight *Voltage* for *Function*.

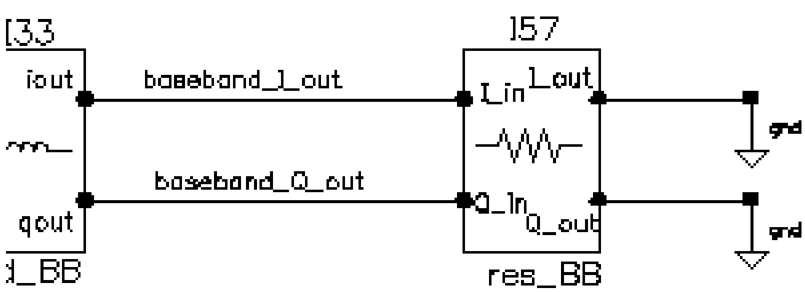
   d. Highlight *time* for *Sweep*.

      Notice the *Net* selection in the *Select* cyclic field and the message *Description: Envelope Voltage vs Time*

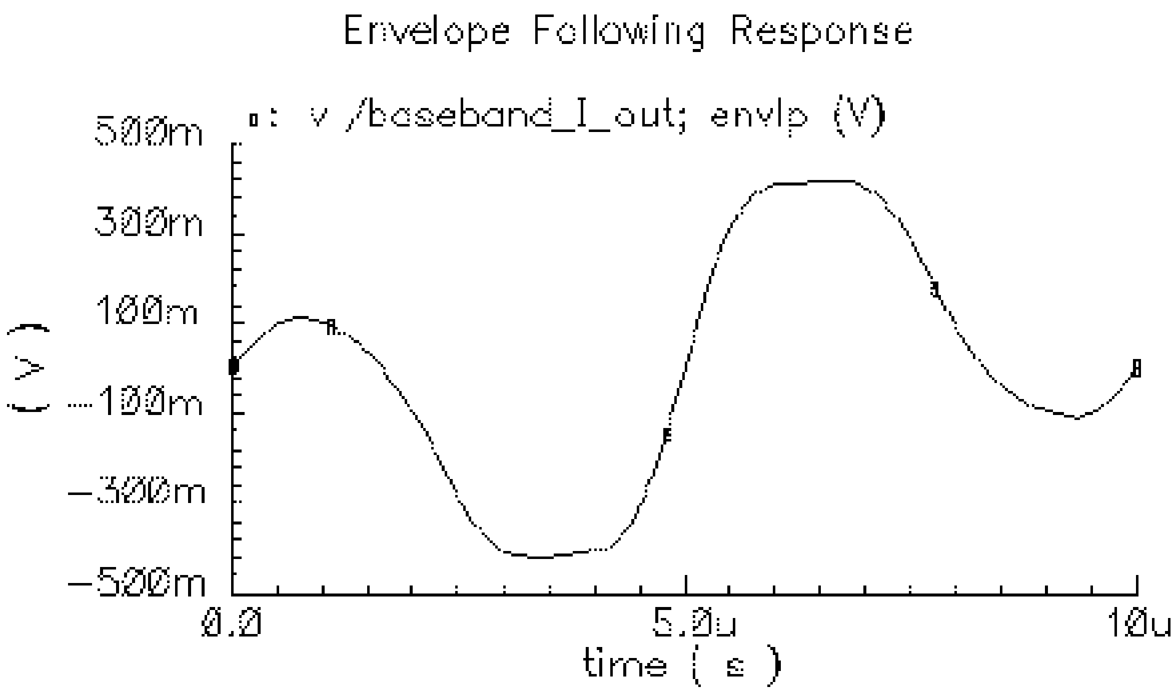
   e. Following the message at the bottom of the Direct Plot form

      Select Net on schematic...

      Click on the `baseband_I_out` net.



The first trace appears in the Waveform window.



3. In the Direct Plot form, do the following:

a. Leave *Voltage* set for *Function* and *time* set for *Sweep*.

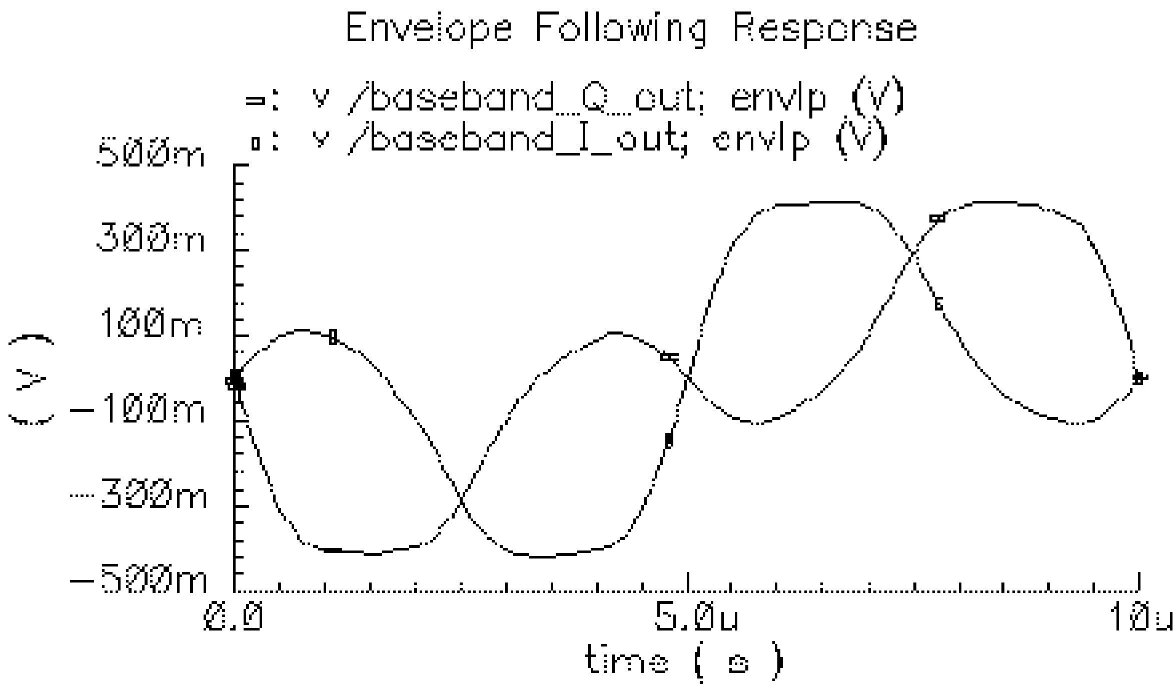
b. Highlight *Append* for *Plot Mode.*

c. Following the message at the bottom of the Direct Plot form

Select Net on schematic...

Click on the `baseband_Q_out` net.

The second trace is added to the Waveform window. Both baseband equivalent output signals for the baseband model are plotted.



4. In the Direct Plot form, do the following:

a. Leave *Append* for *Plot Mode* and *Voltage* for *Function.*

b. Highlight *harmonic time* for *Sweep*.

c. Highlight *Real* for *Modifier*.

5. Following the message at the bottom of the form,

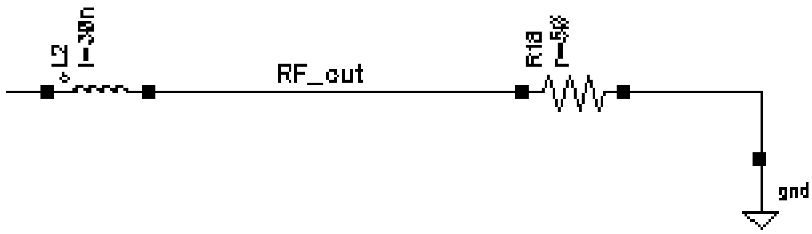
Select Harmonic Number on this form...

Select 1 for harmonic number.

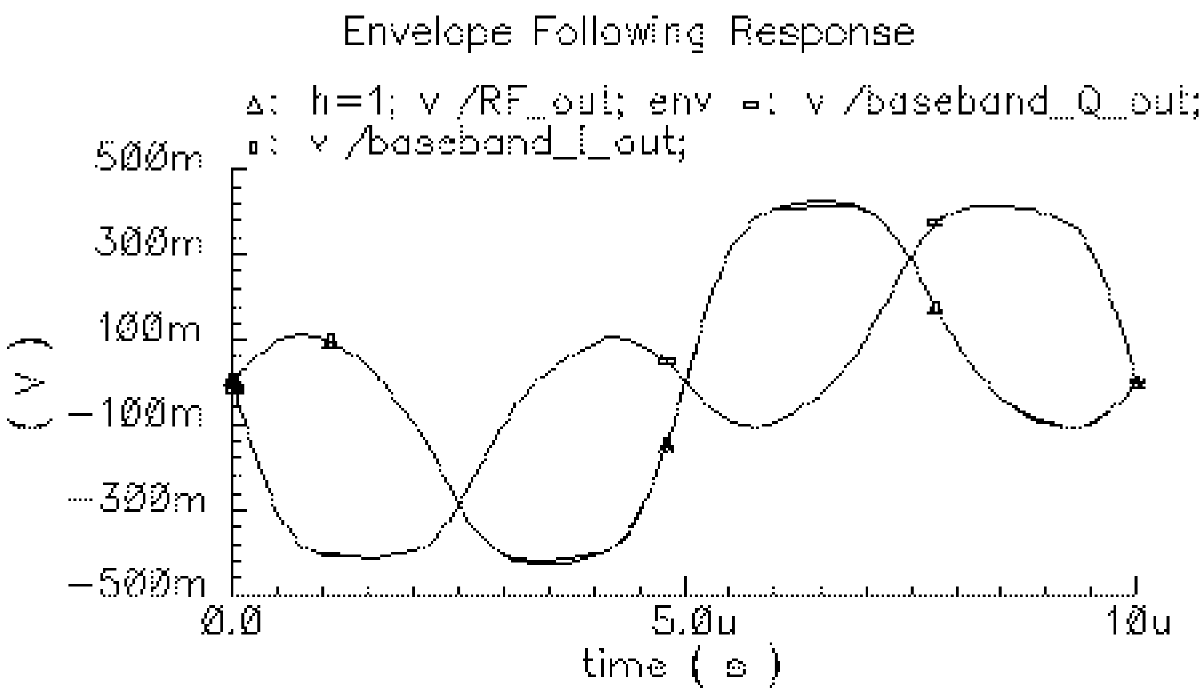
6. Following the next message at the bottom of the form

Select Net on schematic...

Click on the `RF_out` net.



A third trace is added to the Waveform window.



7. In the Direct Plot form, do the following:

    a. Leave *Append* for *Plot Mode*, *Voltage* for *Function*, *harmonic time* for *Sweep*, and 1 for *Harmonic Number*.

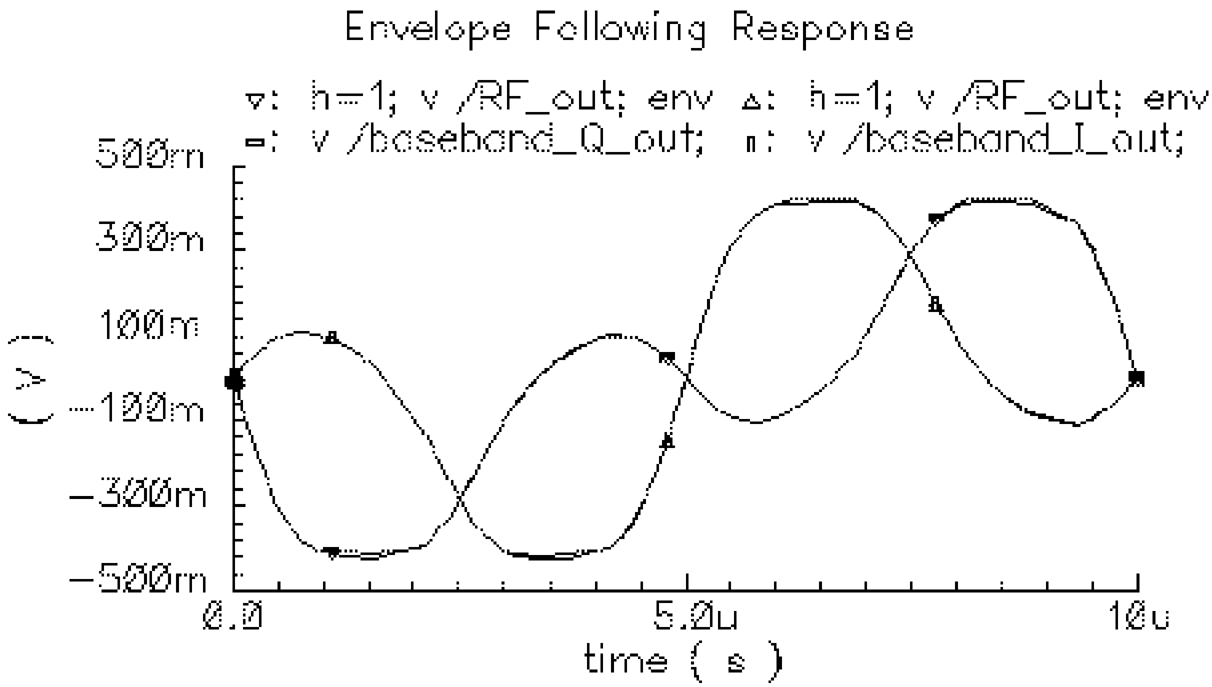
    b. Highlight *Imaginary* for *Modifier*.

8. Following the message at the bottom of the form

    Select Net on schematic...

    Click on the `RF_out` net again.

A fourth trace is added to the Waveform window. Both baseband equivalent output signals for the passband model are added to the plot.



In the Waveform display window you should now see what at first appears to be two traces. When you look more closely, you should see that each trace is actually two traces, one nearly on top of the other, making a total of four traces.

The plot resulting from this example illustrates how well baseband modeling corresponds to the time-varying fundamental Fourier component computed by Envelope Following analysis and raises two questions:

- Why use baseband models when Envelope Following analysis gives the same results?
- Why not use baseband models all the time?

Running a transient analysis with only the baseband models answers the first question. If from the Simulation window you deactivate the passband circuit by setting the `carrier_pb` variable to zero, disable the Envelope Following analysis, and set up and run a 10µs transient analysis, you will observe the same baseband results, but the transient simulation will run over 100 times faster.

Examining the Envelope Following results answers the second question. If you look closely at both waveforms you will notice that the baseband waveforms clip at a slightly lower level than the Envelope Following waveforms. This is because hard limiting of the carrier generates higher-than-third-order harmonics and the behavioral baseband model only simulates third order non-linearities.

Library Overview

The *rfLib* include three kinds of models to support baseband modeling:

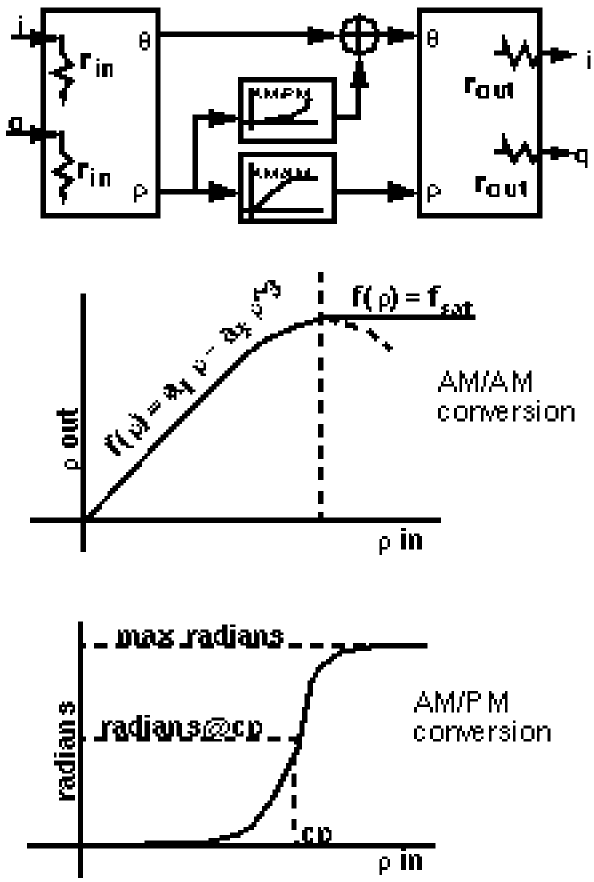
- Instrumentation Models
- Non-Linear Memoryless Models
- Linear Models With Memory

The instrumentation models provide stimuli, diagnostics, and performance metrics relevant to the DSP system.

Both the linear models with memory and the non-linear memoryless models simulate the function blocks in an RF architecture and are specified in terms of common RF metrics. The RF function block models include input referenced white Gaussian noise as specified by noise figure. The *rfLib* includes models for the following RF function blocks—amplifiers, mixers, filters, and phase shifters; where *filters* includes single resistors, capacitors, and inductors.

The non-linear models simulate AM/AM conversion [1] with a third-order polynomial that saturates at the peak of the transfer curve. The polynomial is specified by the gain and either the input-referred IP3 or the output-referred 1 dB compression point. Only the non-linear baseband models simulate AM/PM conversion. AM/PM conversion [1] is an important effect that is hard, if not impossible, to simulate with passband behavioral models. Figure 12-4 on page 923 shows the basic baseband non-linearity.

Figure 12-4 Basic, Baseband Non-Linearity



The linear models are the key to simulating loading effects at baseband. In RF integrated circuits, loading effects are important because it is often hard to integrate impedance matching networks. The baseband models of reactive elements differentiate our approach from the spreadsheet-based approaches to RF system design. The baseband capacitor and inductor models (*cap_BB* and *ind_BB* in *top_dwnBB*) let you simulate reactive loading effects in the time domain, where non-linearities are more naturally modeled.

The baseband models of reactive elements also play a key role in modeling filters. Most digital communications text books [1,2] explain that you can model a passband transfer function at baseband by simply frequency-shifting the transfer function. What these books do not describe is how to implement the resulting transfer function in a general circuit simulator such as Spectre. The shifted transfer function usually lacks complex conjugate symmetry about zero frequency and therefore has a complex impulse response.

The first consequence of modeling RF function blocks at baseband is that all equivalent baseband models have four terminals instead of two:

- One set of terminals represents the in-phase signals, $i_{in}(t)$ and $i_{out}(t)$
- The other set of terminals represents the quadrature signals, $q_{in}(t)$ and $q_{out}(t)$

  Both sets of terminals are illustrated in Figure 12-2 on page 909.

The mathematics illustrated in Figure 12-5 on page 925 and Figure 12-6 on page 926 summarize the ideas behind a time-varying coordinate transformation that models reactive elements at baseband. The mathematics apply to capacitors as well as inductors.

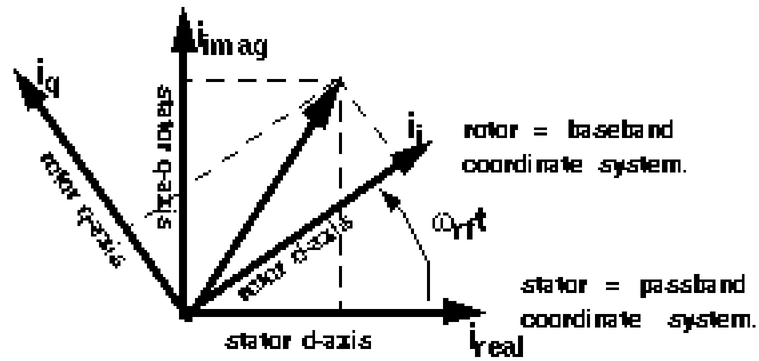
There is a well-documented but little-known electro-mechanical analogy for the derivation of the inductor baseband equivalent model. The four inductor terminals resemble the stator windings of a two-phase rotating machine with shaft speed equal to the RF carrier frequency. Modulation is mathematically analogous to the flux linking a stator winding due to currents in orthogonal rotor windings. The flux depends on the shaft angle just as a modulated signal depends on the carrier phase. Transforming the vectorial equation for $v=Ldi/dt$ to the rotor reference frame suppresses the RF carrier and introduces a *speed voltage* [3,4,5,6,7,8,9], or back electro-motive force (back EMF), that couples the differential equations.

An expression for the real current (i.e. the passband current) appears in Figure 12-5 on page 925. The real current is modeled as the projection of a two-dimensional rotating vector onto a stationary axis, the *real* axis. The vector rotates with an angular velocity equal to the RF carrier frequency.

Figure 12-5 Passband Current for an Inductor

$$i_{real} = i_i(t)\cos(\omega_{rf}t) - i_q(t)\sin(\omega_{rf}t) = Real[(i_i+j*i_q)*e^{j\omega_{rf}t}]$$

$$i_{imag} = i_i(t)\sin(\omega_{rf}t) + i_q(t)\cos(\omega_{rf}t) = Imag[(i_i+j*i_q)*e^{j\omega_{rf}t}]$$



$$i=(i_i+j*i_q)*e^{j\omega_{rf}t}$$

$$v=(v_i+j*v_q)*e^{j\omega_{rf}t}$$

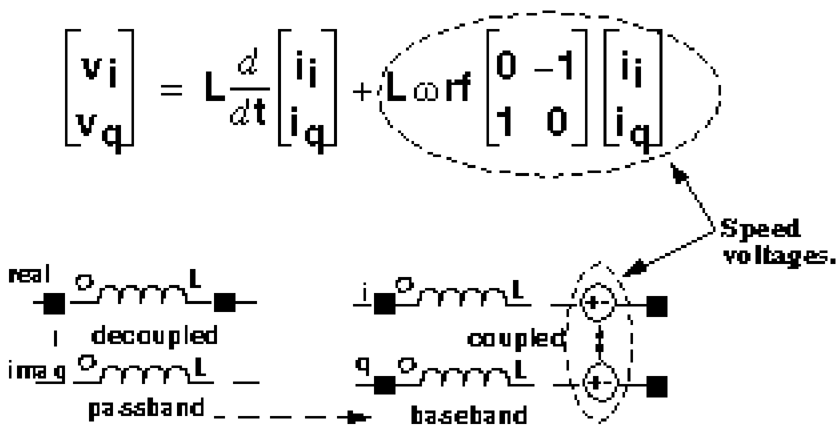
$$v=L\,di/dt$$

$$(v_i+j*v_q)\cancel{e^{j\omega_{rf}t}}=L*[d(i_i+j*i_q)/dt + j*\omega_{rf}*(i_i+j*i_q)]\cancel{e^{j\omega_{rf}t}}$$

The rotating vector also has a projection onto another stationary axis orthogonal to the real axis. In the baseband literature, the orthogonal projection is the Hilbert transform of the real signal. The constitutive relationship of the inductor, *v=Ldi/dt*, is expressed in terms of coordinates in a reference frame that rotates with the vector.

Figure 12-6 on page 926 shows the constitutive inductor relationship between voltage and current in the rotating reference frame. Note that the trigonometric terms, the terms that slow simulation speed, are gone and the two projections are now coupled through *speed voltages*. The term speed voltage comes from the fact that the voltages depend on the angular speed of the rotating reference frame. In motor theory, that speed is the shaft speed. Speed voltage is similar to the back EMF in a motor. Because of speed voltages, baseband models of filters and reactive elements must have their carrier frequency specified. The carrier frequency is the frequency for which the baseband signals are referenced. For example, the carrier frequency for an RF filter would be the RF frequency while the carrier frequency for an IF filter would be the IF frequency.

Figure 12-6 Relationship Between Voltage and Current for an Inductor



The baseband counterparts of the passband filter models are built up from inductors and capacitors modeled in the rotating reference frame.

In the complex expression for *v=Ldi/dt,* if you replace *d/dt* with $j\omega$, you find that the impedance of the inductor changes from $jL\omega$ to $jL(\omega+\omega_{rf})$. The same holds for capacitors, which means a filter transfer function, $H(\omega)$, has a baseband equivalent equal to $H(\omega+\omega_{rf})$. This is simply the original passband transfer function shifted to the left by an amount equal to the carrier frequency. Our time domain baseband models are consistent with the text book frequency domain explanation of baseband modeling.

Warnings You Can Ignore

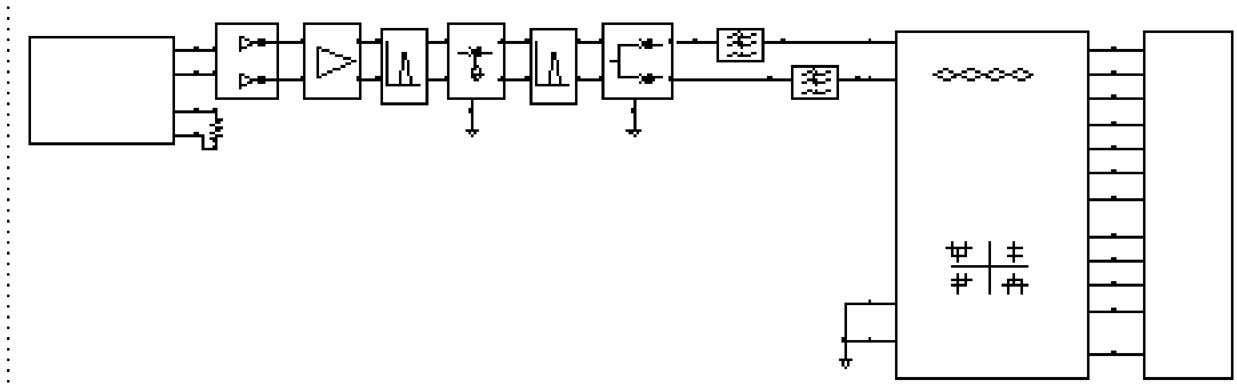
When you use the filters you will see warnings of *No DC path from ... to ground...*. You can ignore those warnings.

When you use the modulators and mixers you will see warnings about *$realtime*. You can ignored these warnings too. *$realtime* was not changed to *$abstime* so that the library would work with older software versions.

Use Model and Design Example

This section describes how to use the baseband models during the architectural design phase. The following examples show you how to

- Construct a baseband model for a simple receiver



- Use the Circuit Optimizer to balance specifications among the function blocks
- Create a passband testbench for the receiver

The design goals were chosen arbitrarily. The example is meant simply to illustrate how to use the library and is a derivative of the design found in [10]. If you find that some parameters are not specified, leave them as default values. You construct the receiver from left to right, from input to output.

Opening a New Schematic Window

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

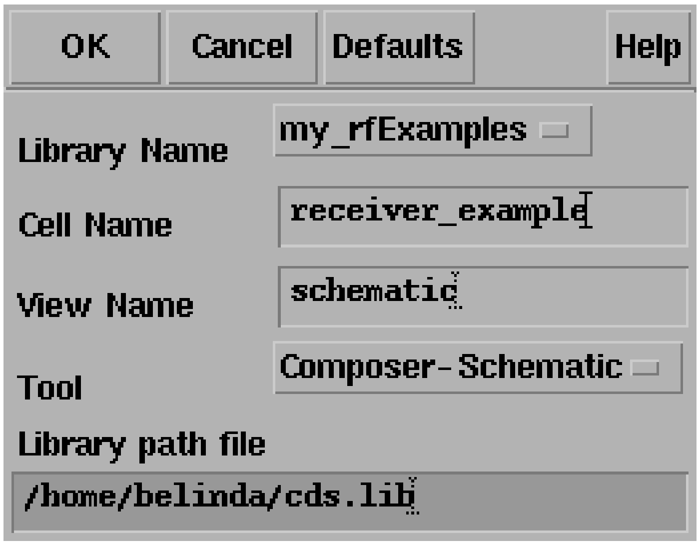
2. In the Create New File form,

   a. Choose *my_rfExamples* in the *Library Name* cyclic field. (Choose the editable copy of *rfExamples* you created as described in Chapter 3.)

   b. Enter `receiver_example` in the *Cell Name* field.

   c. Select *Composer-Schematic* in the *Tool* cyclic field. `Schematic` appears in the *View Name* field.

The completed form appears like the one below.



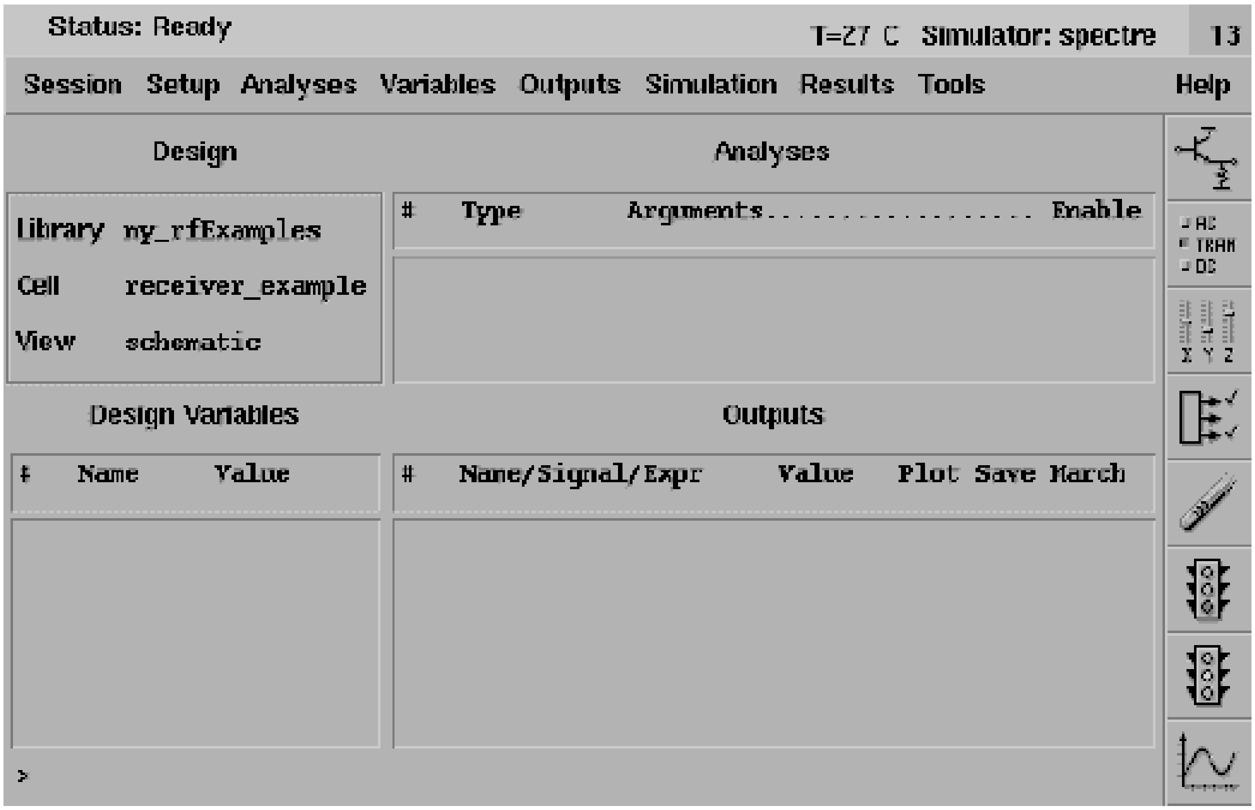
3. Click *OK*.

An empty Schematic window for the *receiver_example* appears.

Opening the Analog Environment

1. In the Schematic window, choose *Tools – Analog Environment*.

The Simulation window opens.



The *Library*, *Cell*, and *View* names appear in the *Design* section of the Simulation window.

2. Set the simulator options from the Simulator window as described in “Choosing Simulator Options” on page 912.

3. Set up the model libraries from the Simulator window as described in “Setting Up Model Libraries” on page 914.

Constructing the Baseband Model for the Receiver

Construct the receiver in the Schematic window by adding blocks from left to right, from input to output, as listed in Table 12-1.

Except for the resistor, ground, and port models (which come from the *analogLib*), all blocks come from the *rfLib*. Unless otherwise instructed, leave the port resistances at their default value of 50 Ohms.

Table 12-1 Blocks Used to Create the Receiver

Block Name and Reference	Element Name	Library and Category
CDMA signal source — See Adding the CDMA Signal Source	`CDMA_reverse_xmit`	From the *measurement* category in *rfLib*.
Resistor—attach to CDMA signal source	`res`	From *analogLib*
Driver — See Adding the Driver	`BB_driver`	From the *measurement* category in *rfLib*.
Low noise amplifier — See Adding the Low Noise Amplifier	`LNA_BB`	From the *top_dwnBB* category in *rfLib*.
Butterworth bandpass filter — See Adding a Butterworth Band Pass Filter	`BB_butterworth_bp`	From the *top_dwnBB* category in *rfLib*
RF-to-IF mixer — See Adding an RF-to-IF Mixer	`dwn_cnvrt`	From the *top_dwnBB* category in *rfLib*
Butterworth bandpass filter — See Adding Another Butterworth Bandpass Filter	`BB_butterworth_bp`	From the *top_dwnBB* category in *rfLib*
IQ demodulator — See Adding an IQ Demodulator	`IQ_demod_BB`	From the *top_dwnBB* category in *rfLib*
Butterworth lowpass filters (create two) — See Adding Two Butterworth Lowpass Filters	`butterworth_lp`	From the *top_dwnPB* category in *rfLib*
Instrumentation model — See Adding an Instrumentation Block	`offset_comms_instr`	From the *measurement* category in *rfLib*.
Terminator — See Adding an Instrumentation Terminator	`instr_term`	From the *measurement* category in *rfLib*
Grounds—attach to RF-to-IF mixer, IQ demodulator, and Instrumentation model	`gnd`	From *analogLib*

Adding the CDMA Signal Source

Add the first receiver block, a CDMA signal source (`CDMA_reverse_xmit`), to the schematic.

1. In the Schematic window, choose *Add – Instance*.

   The Add Instance form appears. It may be empty or it may display information for a previously added element. The default for the *View* field is `symbol`.

2. In the Add Instance form, click *Browse*.

   The Library Browser - Add Instance form appears.

3. In the Library Browser - Add Instance form,

   a. If necessary, click *Show Categories* to display the *Category* column so you can view the elements (or cells) in the *rfLib* by category.

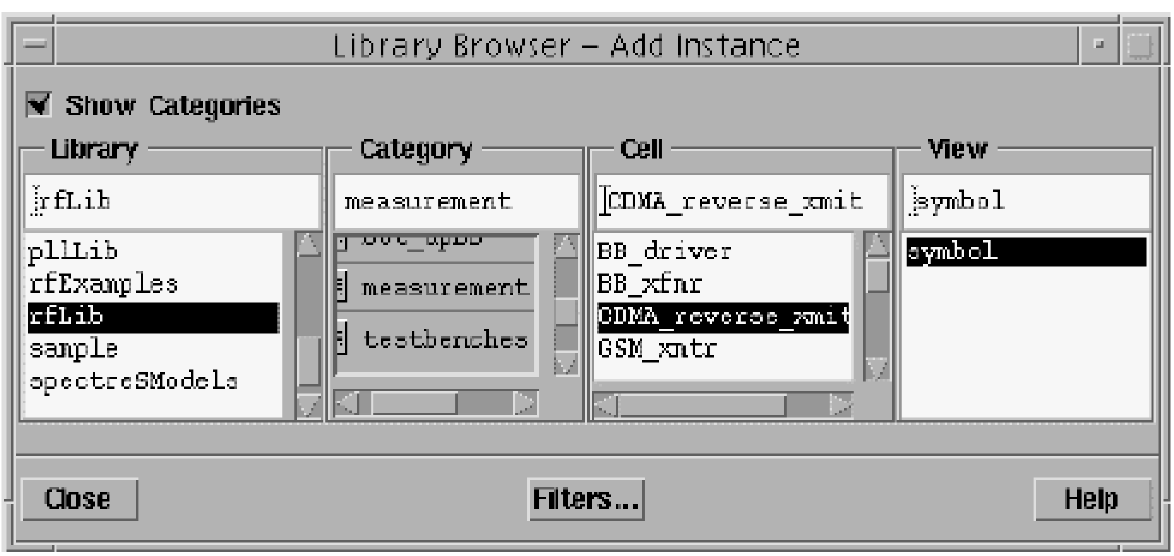
   b. In the *Libraries* column, click *rfLib* to display categories of elements in *rfLib*.

      The *Everything* category is displayed by default and all cells in *rfLib* are listed in the *Cells* column. (In the Add Instance form, *rfLib* displays in the *Library* field.)

   c. In the *Category* column, click *measurement* to list only the cells in the *measurement* category.

   d. In the *Cell* column, click *CDMA_reverse_xmit*.

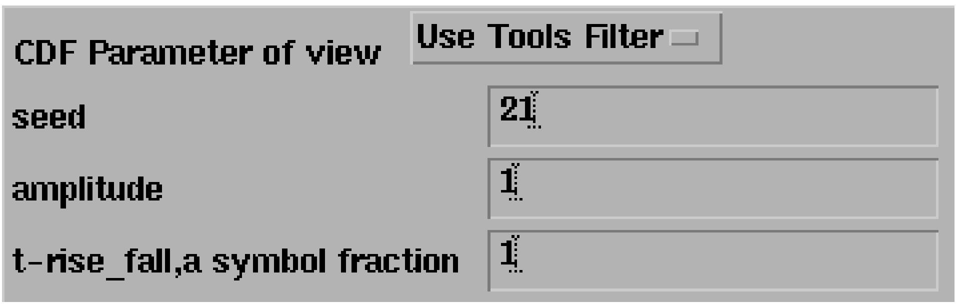
   In the Library Browser, cell *CDMA_reverse_xmit* and it's default view *symbol* are both selected.



In the Add Instance form,

- ❑ `rfLib` appears in the *Library* field
- ❑ `CDMA_reverse_xmit` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field

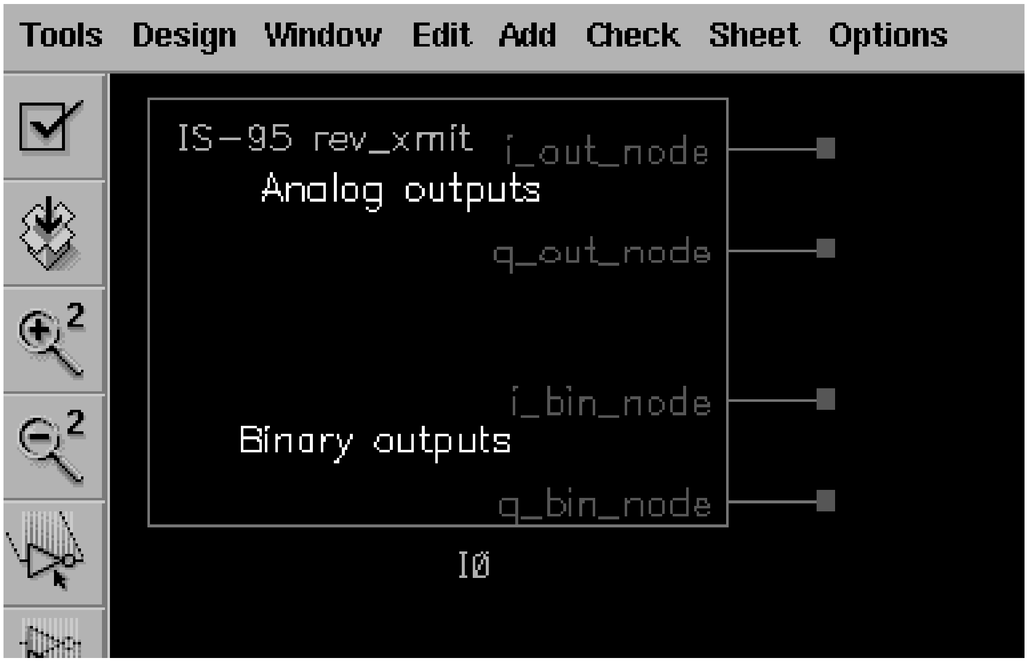
The CDF parameters for the element and their default values appear at the bottom of the form.



4. To place a *CDMA_reverse_xmit* block in the schematic, move the cursor over the Schematic window. The outline for the *CDMA_reverse_xmit* symbol is attached to the cursor.

Move the cursor near the top left corner of the schematic and click to place the *CDMA_reverse_xmit* block. This block models a CDMA signal source.

5. Click *Esc* to remove the symbol from the cursor.



Add a Resistor to the CDMA Signal Source

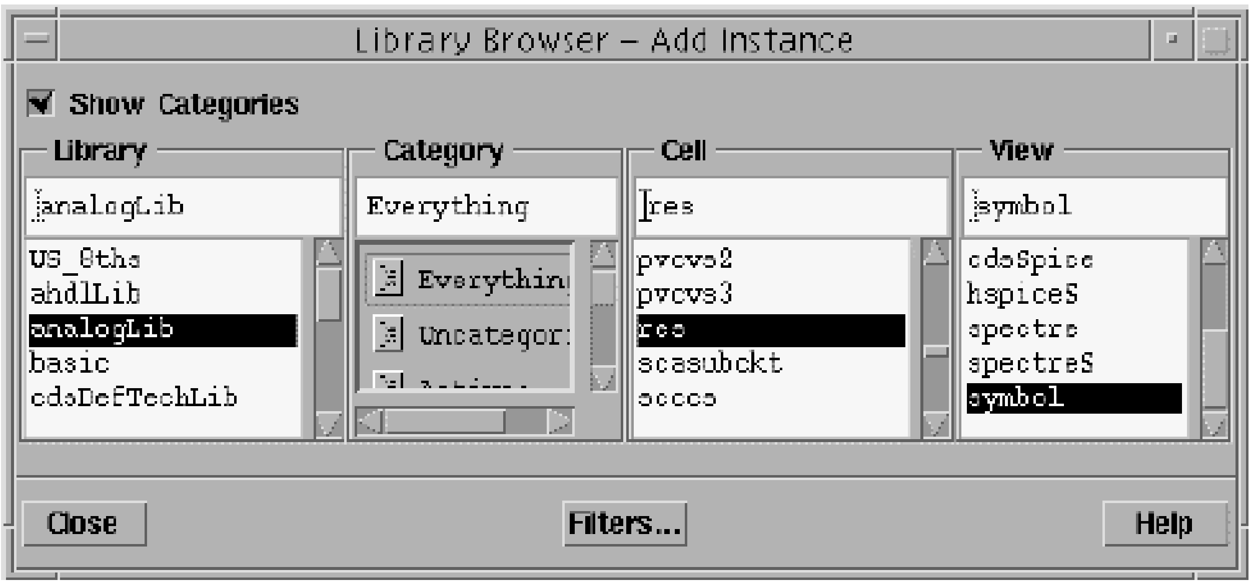
Since this example does not use the binary output nodes (`i_bin_node` and `q_bin_node`) on the CDMA signal source, connect a resistor between these nodes to avoid unused pin warnings.

1. In the *Libraries* column of the Library Browser - Add Instance form, click *analogLib* to display elements in *analogLib*.

   If *Show Categories* is selected, the *Everything* category is displayed by default and all cells in *analogLib* are listed in the *Cells* column.

2. Scroll through the list of cells in *analogLib* to locate the resistor cell, *res*.
3. Click *res* in the *Cell* column.

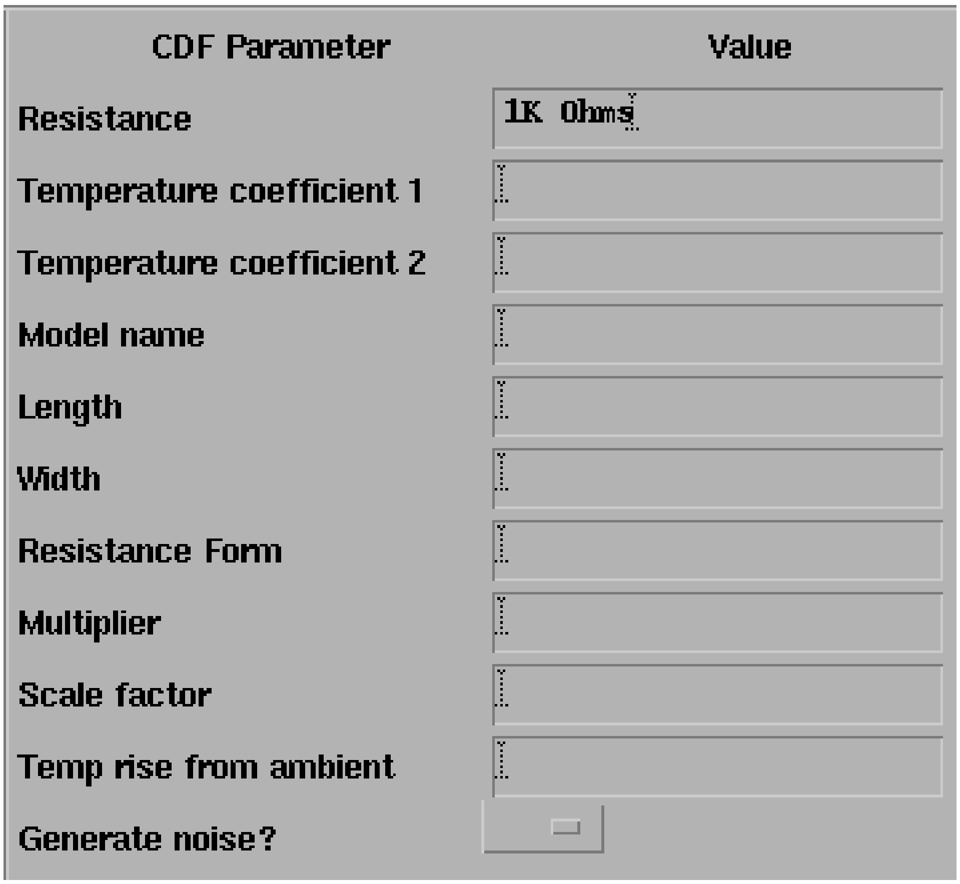
The cell *res* and it’s default view *symbol* are both selected.



In the Add Instance form,

- ❑ `analogLib` displays in the *Library* field
- ❑ `res` displays in the *Cell* field
- ❑ and `symbol` displays in the *View* field

The CDF parameters for the element and their default values appear at the bottom of the form.



4. Move the cursor over the Schematic window.
5. Click to place the top resistor terminal in line with the top binary output node (`i_bin_out`) on the lower right side of the *CDMA_reverse_xmit* block
6. Click *Esc* to remove the symbol from the cursor.

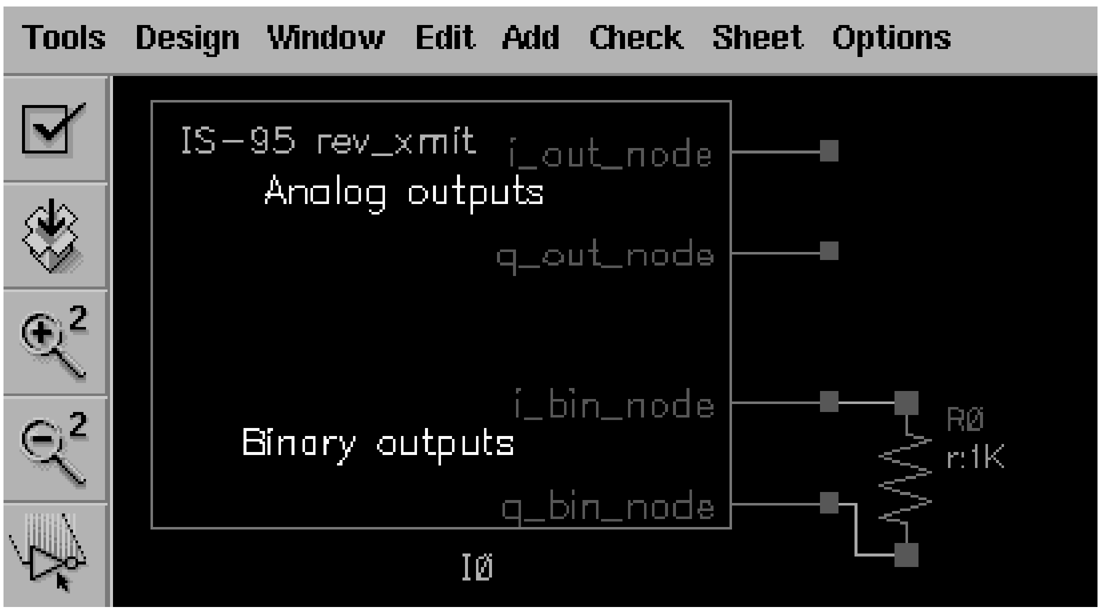
Wiring the Resistor to the CDMA Signal Source

Wire the resistor to the binary outputs, `i_bin_node` and `q_bin_node`, of the CDMA signal source.

1. To wire the resistor to the *CDMA_reverse_xmit* block, in the Schematic window choose *Add - Wire (narrow)*.

2. Click `i_bin_node` on the `CDMA_reverse_xmit` block then move the cursor and click the top node of the resistor.
3. Click `q_bin_node` on the `CDMA_reverse_xmit` block then move the cursor and click the bottom node of the resistor.
4. Click *Esc* to stop wiring.

   The CDMA signal source and resistor wired together appear as follows.



Adding the Driver

Add a driver block to the right of the CDMA signal source block.

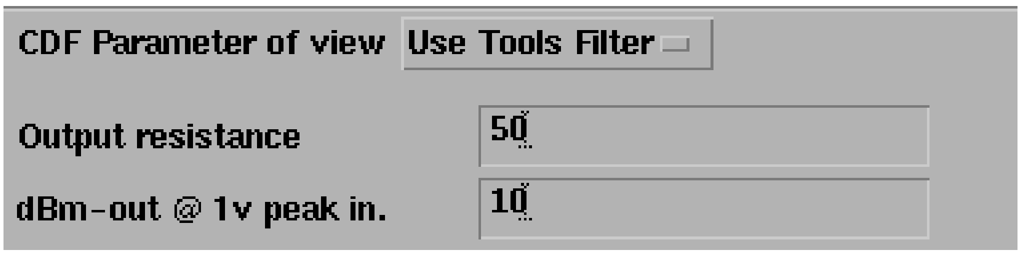
1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.
2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
3. In the Library Browser - Add Instance form, make the following selections.

Library	Category	Cell	View
rfLib	*measurement*	*BB_driver*	*symbol*

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field,
- ❑ `BB_driver` displays in the *Cell* field and
- ❑ `symbol` displays in the *View* field.

The CDF parameters and their default values appear at the bottom of the Add Instance form.

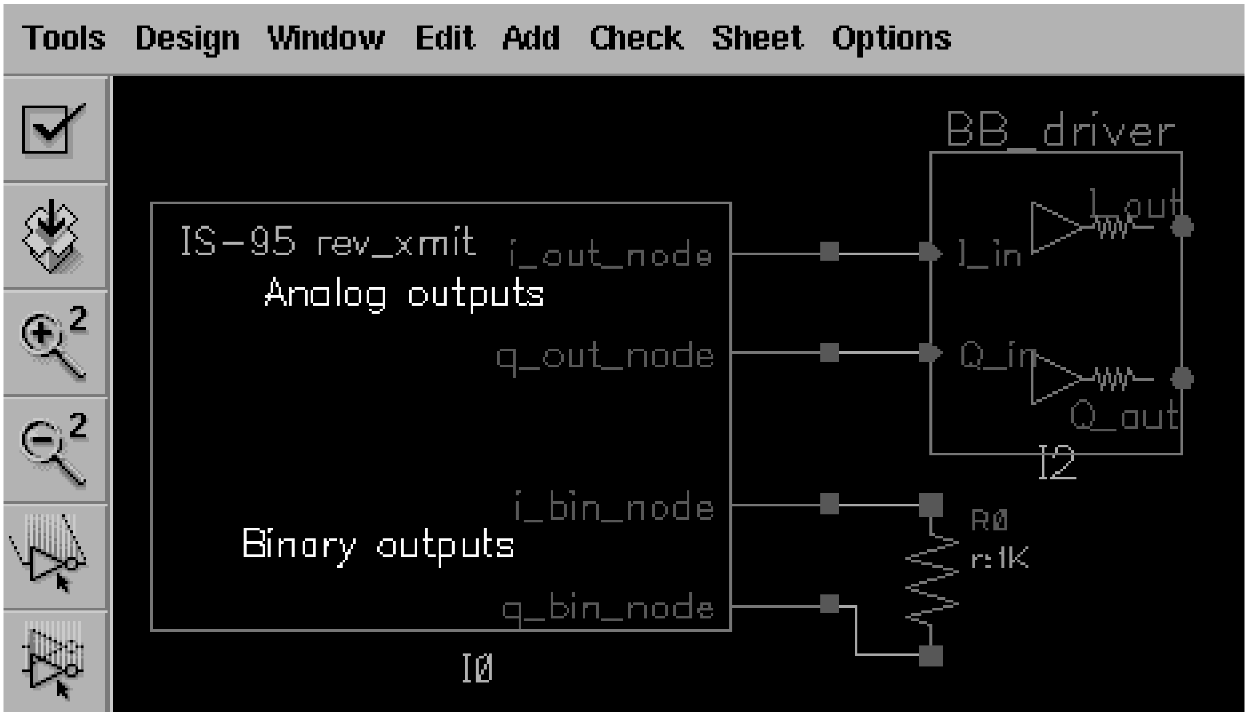


4. Move the cursor over the Schematic window.
5. Click to place the *BB_driver* to the right of the *CDMA_reverse_xmit* block. Align the input pins of the driver with the analog output pins of the CDMA signal source.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Signal Source to the Driver

1. To wire the *BB_driver* block to the *CDMA_reverse_xmit* block, in the Schematic window choose *Add — Wire (narrow)*.
2. Click `i_out_node` on the *CDMA_reverse_xmit* block then click `I_in` on the *BB_driver* block.
3. Click `q_out_node` on the *CDMA_reverse_xmit* block then click `Q_in` on the *BB_driver block*.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.



Modifying Parameter Values for the Driver

Edit the value of the *BB_driver* CDF parameter *dBm-out@1v peak in driver* as follows.

1. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the values of CDF (component description format) properties for the driver and modify the schematic for this simulation.

2. In the Schematic window, click on the *BB_driver* block.
3. The Edit Object Properties form changes to display information for the *BB_driver* block
4. Change the *dBm-out@1v peak in* parameter value as follows.

Parameter Name	Value
dBm-out@1v peak in	*-16*

The driver converts 1 peak volt from the CDMA signal source to –16 dBm referenced to the output resistance of the driver.

Adding the Low Noise Amplifier

Add a low noise amplifier to the right of the driver.

1. In the Schematic window, choose *Add—Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click *Show Categories* at the top of the Library Browser, to display the *Category* column.)

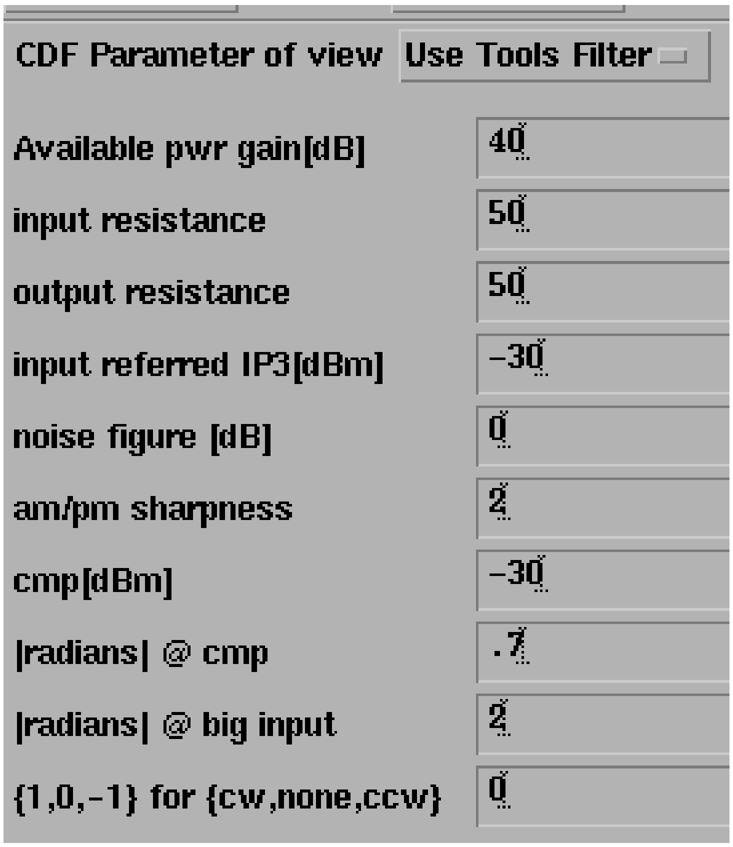
Library	Category	Cell	View
rfLib	top_dwnBB	LNA_BB	symbol

   In the Library Browser, cell *LNA_BB* and it's default view *symbol* are both selected.

   At the top of the Add Instance form,

   - ❑ `rfLib` displays in the *Library* field
   - ❑ `LNA_BB` displays in the *Cell* field
   - ❑ `symbol` displays in the *View* field

The CDF parameters for the element and their default values display at the bottom of the LNA_BB Add Instance form.

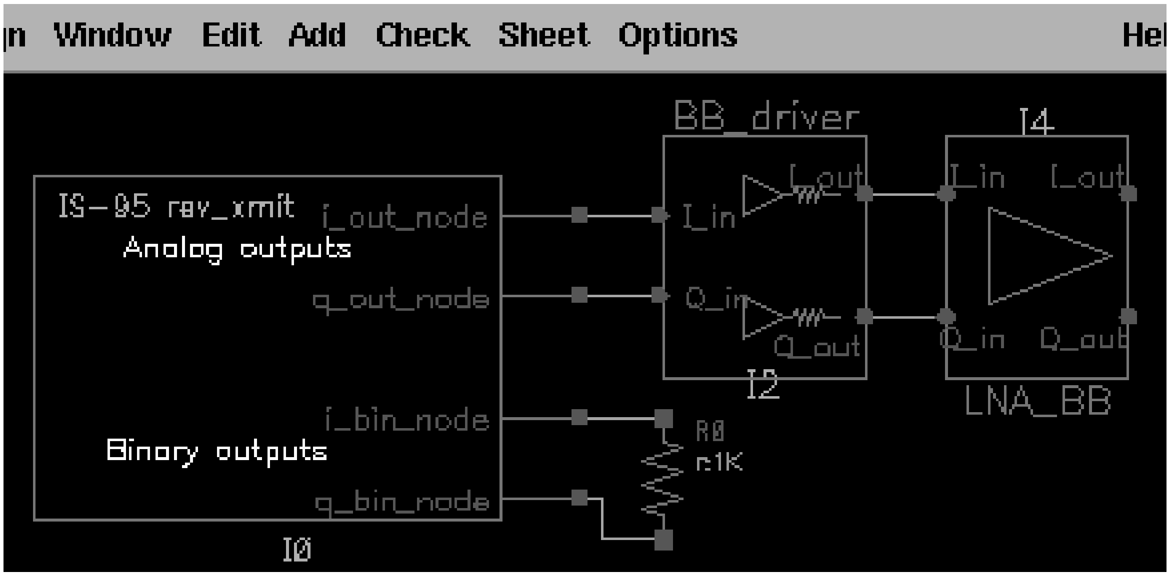


4. Move the cursor over the Schematic window. The outline for the LNA symbol is attached to the cursor. Align the input pins of the LNA with the output pins of the driver.
5. Click to place the *LNA_BB* block to the right of the *BB_driver* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Driver to the LNA

1. To wire the *LNA_BB* block to the *BB_driver* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *BB_driver* block then click `I_in` on the *LNA_BB* block.
3. Click `Q_out` on the *BB_driver* block then click `Q_in` on the *LNA_BB* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.



Modifying Parameter Values for the LNA

1. Edit the CDF parameter values for the LNA.

a. Choose *Edit—Properties—Objects* in the Schematic window.

The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the LNA and modify the schematic for this simulation.

b. In the Schematic window, click on the *LNA*.

The Edit Object Properties form changes to display information for the LNA.

c. Change the CDF parameter values for the LNA as follows.

Parameter Name	**Value**
Available pwr gain [dB]	`lna_gain`
Input resistance	`50`
Output resistance	`300`
Input referred IP3 [dBm]	`lna_ip3`
Noise figure [dB]	`10`

Parameter Name	Value
am/pm sharpness	`2`
cmp [dBm]	`lna_ip3`
|radians| @ cmp	`.05`
|radians| @ big input	`.7`
{1, 0, -1} for {cw, none, ccw}	`1`

Adding a Butterworth Band Pass Filter

Add a Butterworth band pass filter to the right of the low noise amplifier.

1. In the Schematic window, choose *Add—Instance* to display the Add Instance form.

   Information for the LNA is still displayed in the form.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selection.

Library	Category	Cell	View
rfLib	*top_dwnBB*	*BB_butterworth_bp*	*symbol*

   In the Library Browser, cell *BB_butterworth_bp* and it's default view are both selected.

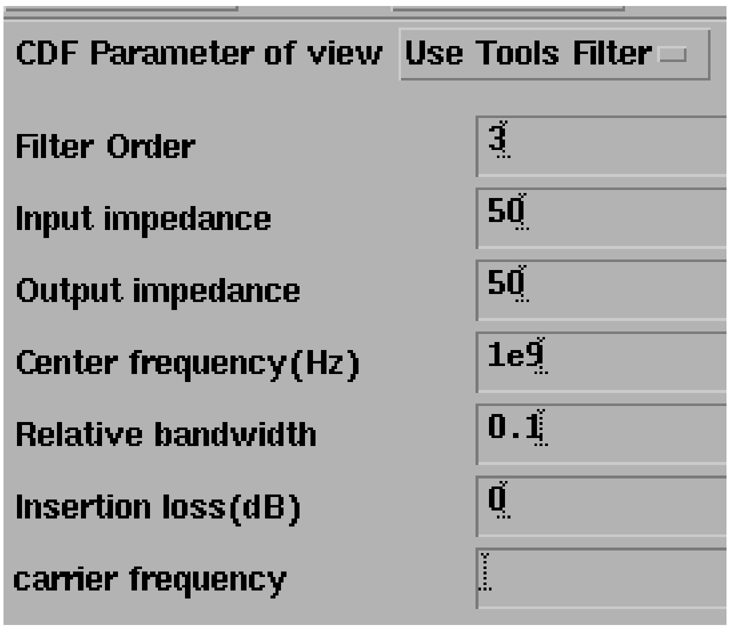
   At the top of the Add Instance form,

   ❑ `rfLib` displays in the *Library* field

   ❑ `BB_butterworth_bp` displays in the *Cell* field

   ❑ `symbol` displays in the *View* field.

The CDF parameters for the Butterworth band pass filter and their default values display at the bottom of the Add Instance form.

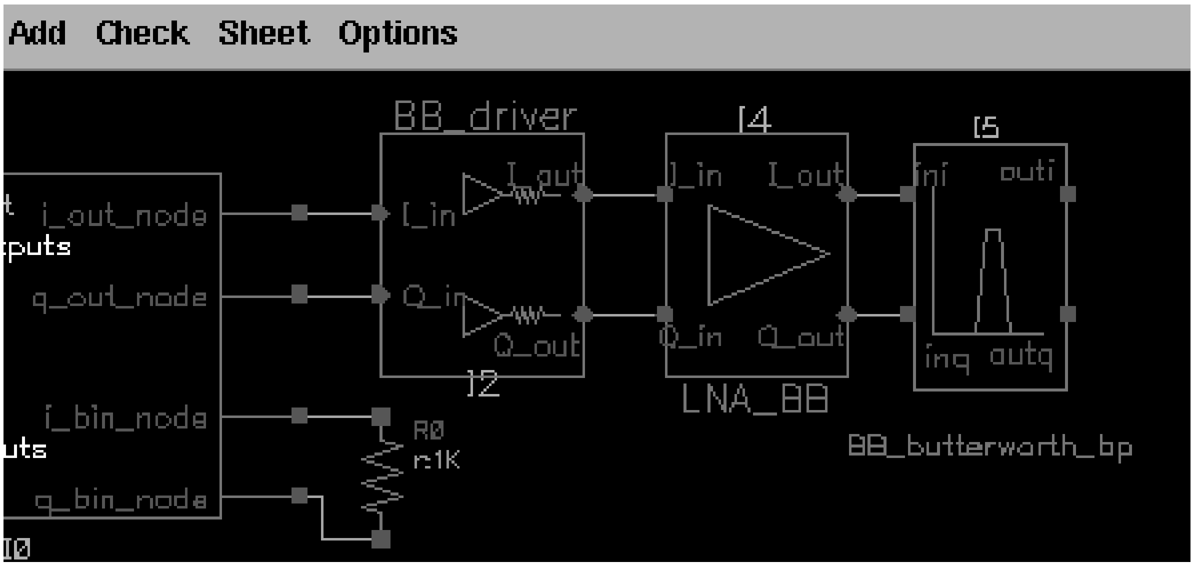


4. Move the cursor over the Schematic window. The outline for the filter symbol is attached to the cursor. Align the input pins of the filter with the output pins of the LNA.
5. Click to place the *BB_butterworth_bp* block to the right of the *LNA_BB* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the LNA to the Filter

1. To wire the *BB_butterworth_bp* block to the *LNA_BB* block, in the Schematic window choose *Add—Wire (narrow)*.
2. Click `I_out` on the *LNA_BB* block then click `ini` on the *BB_butterworth_bp* block.
3. Click `Q_out` on the *LNA_BB* block then click `inq` on the *BB_butterworth_bp* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.



Modifying Parameter Values for the Band Pass Filter

Edit the CDF parameter values for the Butterworth band pass filter.

1. Choose *Edit—Properties—Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the filter and modify the schematic for this simulation.

2. In the Schematic window, click on the filter.

   The Edit Object Properties form changes to display information for the filter.

3. Edit the parameter values to match those in Table 12-2.

4. Click *OK*.

Table 12-2 CDF Parameter Values for the Butterworth Filter

Parameter Name	Value
Filter order	3
Input impedance	50
Output impedance	50

Table 12-2 CDF Parameter Values for the Butterworth Filter

Parameter Name	Value
Center frequency (Hz)	`frf`
Relative bandwidth	`rf_rbw`
Insertion loss (dB)	`3`
Carrier frequency	`frf`

Specify the *Carrier frequency* parameter value for the baseband equivalent model of the Butterworth band pass filter, just as you do for any reactive element. As shown in Figure 12-6 on page 926, the carrier frequency is used to compute speed voltages. Since filters are built up from inductors and capacitors which have speed voltages, you must specify the carrier frequency for filters.

When a filter follows an RF-to-IF mixer, its *Carrier frequency* parameter value is the IF frequency.

- The *Carrier frequency* is the frequency value to which the baseband signals are referenced.
- The *Center frequency* is the frequency for which a filter is designed. The *Center frequency* parameter value for a bandpass filter does not have to equal the *Carrier frequency* parameter value.

Adding an RF-to-IF Mixer

Add an RF-to-IF mixer (*dwn_cnvrt*) block to the right of the bandpass filter block.

1. In the Schematic window, choose *Add—Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.
3. In the Library Browser - Add Instance form, make the selections indicated in Table 12-3.

Table 12-3 Library Browser selections for the RF to IF Mixer

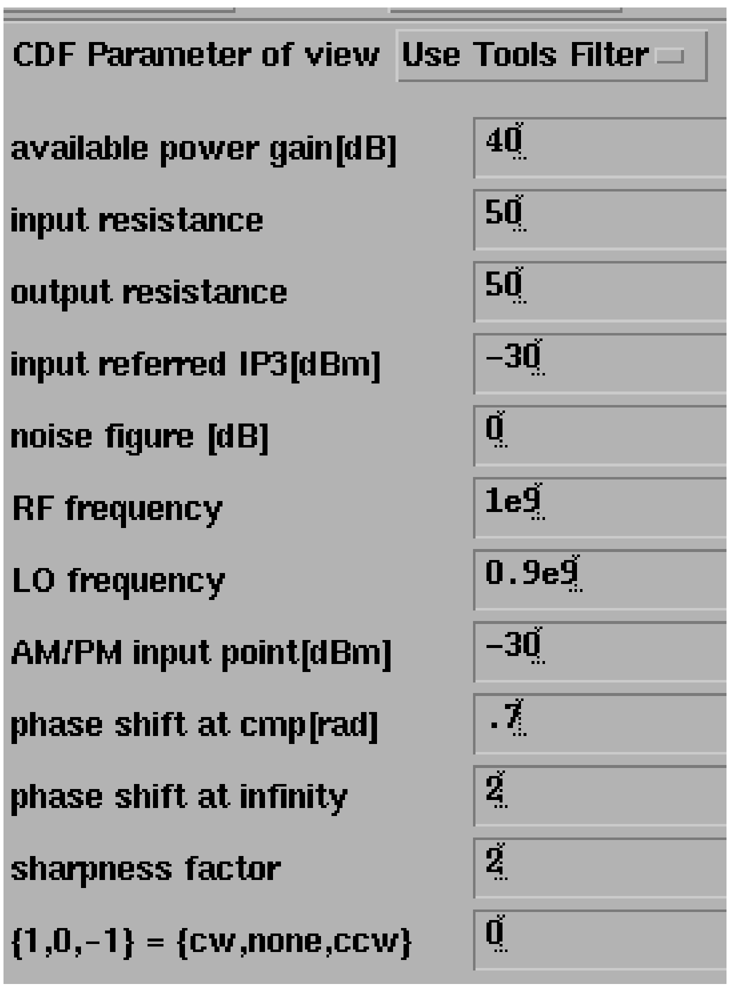
Library	Category	Cell	View
rfLib	*top_dwnBB*	*dwn_cnvrt*	*symbol*

In the Library Browser, cell *dwn_cnvrt* (the RF-to-IF mixer) and it's default view *symbol* are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field
- ❑ `dwn_cnvrt` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field.

The CDF parameters for the down converter and their default values display at the bottom of the form.

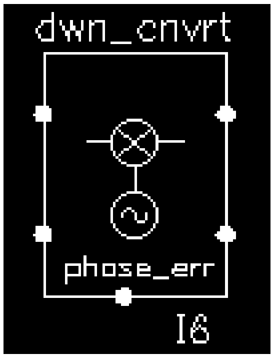


4. Move the cursor over the Schematic window. The outline for the RF-to-IF Mixer symbol is attached to the cursor. Align the input pins of the mixer with the output pins of the filter.
5. Click to place the *dwn_cnvrt* block to the right of the *BB_butterworth_bp* block.

6. Click *Esc* to remove the symbol from the cursor.

Grounding the phase_err Pin on the Mixer

It is necessary to ground the phase error (*phase_err*) pin on the bottom of the RF-to-IF mixer.



1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

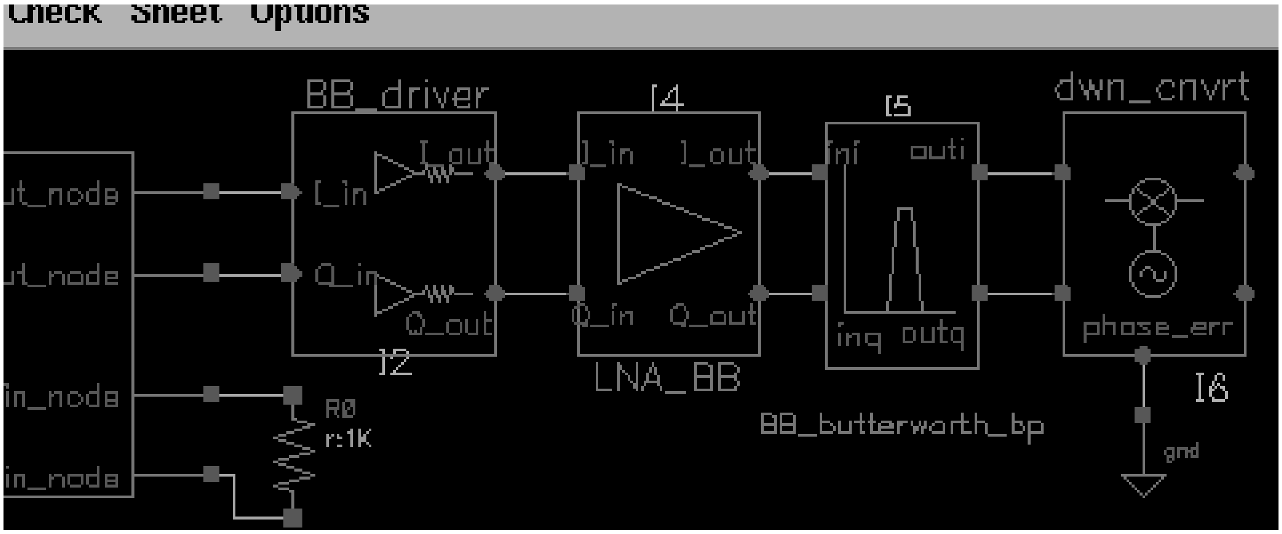
2. In the Add Instance form, type
   - ❑ `analogLib` in the *Library* field
   - ❑ `gnd` in the *Cell* field
   - ❑ `symbol` in the *View* field.
3. Move the cursor over the Schematic window.
4. Click to place the ground terminal in line with the phase error node (`phase_err`) on the bottom of the *dwn_cnvt* block.
5. Click *Esc* to remove the symbol from the cursor.

Wiring the Filter and Ground to the Mixer

1. To wire the *dwn_cnvrt* block to the *BB_butterworth_bp* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `outi` on the *BB_butterworth_bp* block then click `I_in` on the *dwn_cnvrt* block.
3. Click `outq` on the *BB_butterworth_bp* block then click `Q_in` on the *dwn_cnvrt* block.
4. Click the port on the *gnd* block then click `phase_err` on the *dwn_cnvrt* block.

5. Click *Esc* to stop wiring.

   The schematic now appears as follows.



Modifying Parameter Values for the RF-to-IF Mixer

1. Edit the CDF parameter values for the RF-to-IF mixer (*dwn_cnvrt*) as listed in Table 12-4 on page 948.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

      The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *dwn_cnvrt* and modify the schematic for this simulation.

   b. In the Schematic window, click on *dwn_cnvrt*.

      The Edit Object Properties form changes to display information for *dwn_cnvrt*.

   c. Change the parameter values to match those in Table 12-4.

   d. Click OK.

Table 12-4 CDF Parameter Values for the RF-to-IF Mixer

Parameter Name	Value
available power gain[dB]	`if_mx_gain`
Input resistance	`50`

Table 12-4 CDF Parameter Values for the RF-to-IF Mixer

Parameter Name	Value
output resistance	`50`
input referred ip3[dBm]	`if_mx_ip`
noise figure [dB]	`10`
RF frequency	`frf`
LO frequency	`flo1`
AM/PM input point[dBm]	`-30`
phase shift at cmp[rad]	`.7`
phase shift at infinity	`2`
sharpness factor	`2`
{1,0,–1} = {cw,none,ccw}	`0`

Adding Another Butterworth Bandpass Filter

Add another Butterworth band pass filter block to the right of the RF-to-IF Mixer block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   Information for the mixer is still displayed in the form and the outline of the mixer is still attached to the cursor.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selection.

Library	Category	Cell	View
rfLib	*top_dwnBB*	*BB_butterworth_bp*	*symbol*

In the Library Browser, cell *BB_butterworth_bp* and it's default view are both selected.

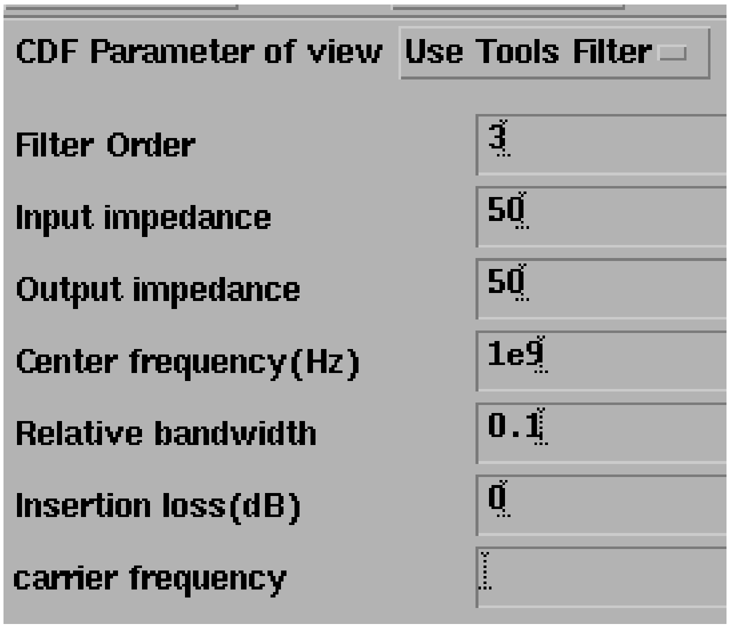
At the top of the Add Instance form,

❑ `rfLib` displays in the *Library* field

❑ `BB_butterworth_bp` displays in the *Cell* field

- ❑ `symbol` displays in the *View* field.

The CDF parameters for the Butterworth band pass filter and their default values display at the bottom of the Add Instance form.

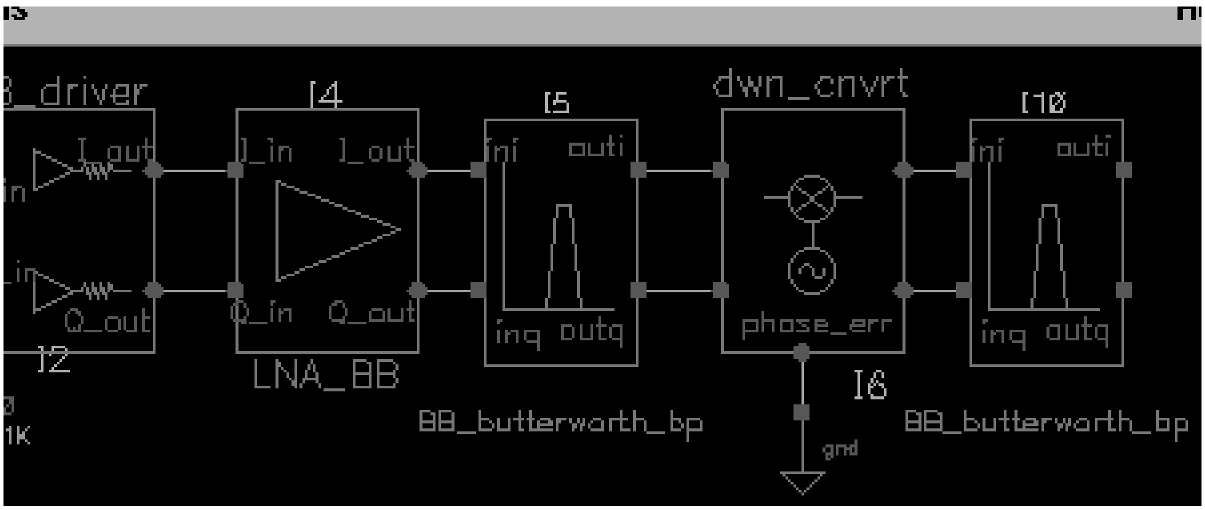


4. Move the cursor over the Schematic window. The outline for the filter symbol is attached to the cursor. Align the input pins of the filter with the output pins of the LNA.
5. Click to place the *BB_butterworth_bp* block to the right of the *LNA_BB* block.
6. Click *Esc* to remove the symbol from the cursor.

Wiring the Mixer to the Filter

1. To wire the *BB_butterworth_bp* block to the *dwn_cnvrt* block, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *dwn_cnvrt* block then click `ini` on the *BB_butterworth_bp* block.
3. Click `Q_out` on the *dwn_cnvrt* block then click `inq` on the *BB_butterworth_bp* block.
4. Click *Esc* to stop wiring.

5. The schematic now appears as follows.



Modifying Parameter Values for the Band Pass Filter

1. Edit the CDF parameter values for the Butterworth band pass filter as listed in Table 12-5 on page 951.

a. Choose *Edit – Properties – Objects* in the Schematic window.

The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for the filter and modify the schematic for this simulation.

b. In the Schematic window, click on the filter.

The Edit Object Properties form changes to display information for the filter.

c. Change the parameter values to match those in Table 12-5.

d. Click OK.

Table 12-5 CDF Parameter Values for the Second Butterworth Filter

Parameter Name	Value
Filter Order	3
Input impedance	50
Output impedance	50
Center frequency (Hz)	`–frf+flo1`

Table 12-5 CDF Parameter Values for the Second Butterworth Filter

Parameter Name	Value
Relative bandwidth	`if_rbw`
Insertion loss (dB)	`1`
Carrier frequency	`–frf+flo1`

As for the first band pass filter, specify the carrier frequency for the baseband equivalent model of the Butterworth band pass filter, just as you do for any reactive element. As shown in Figure 12-6 on page 926, the carrier frequency is used to compute speed voltages. Since filters are built up from inductors and capacitors which have speed voltages, you must specify the carrier frequency for filters.

When a filter follows an RF-to-IF mixer, its carrier frequency is the IF frequency. The carrier frequency is the frequency to which the baseband signals are referenced. The center frequency of the bandpass filter does not have to equal the carrier frequency. The center frequency is the frequency for which a filter is designed.

Adding an IQ Demodulator

Add an IQ Demodulator (*IQ_demod_BB*) block to the right of the bandpass filter block.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form,

   a. If necessary, click *Show Categories* to display the *Category* column.

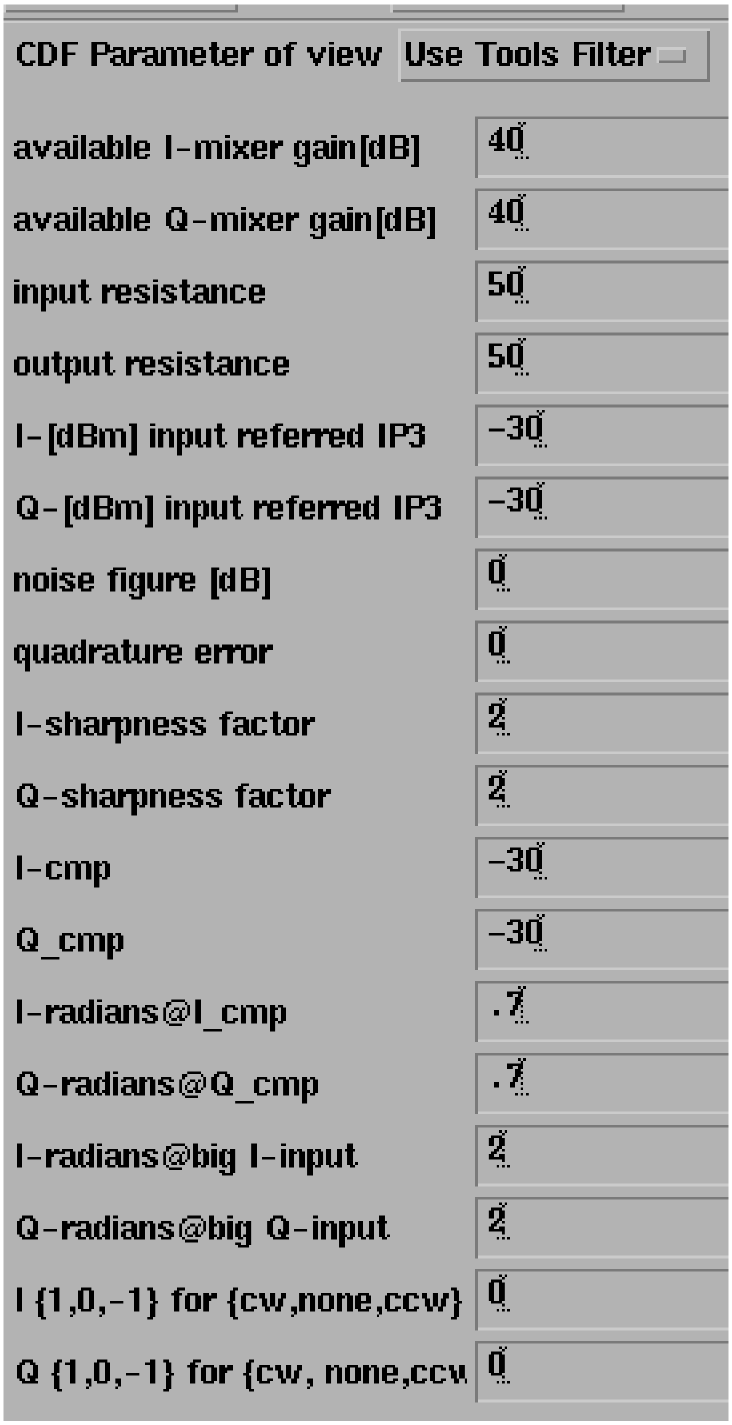
   b. Click *rfLib* to display elements in *rfLib*. The *Everything* category is displayed by default and all cells in *rfLib* are listed in the *Cells* column.

   c. In the *Category* column, click *top_dwnBB* to display cells in the *top_dwnBB* category.

   d. In the *Cell* column, click *IQ_demod_BB*.

In the Library Browser, cell *IQ_demod_BB* (the IQ demodulator) and it's default view *symbol* are both selected.

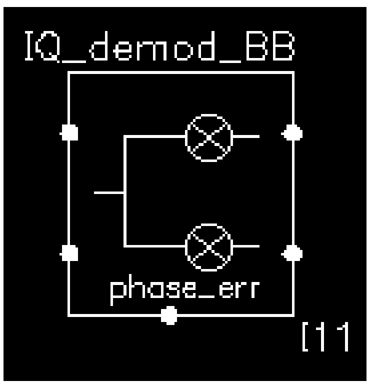
At the top of the Add Instance form, `rfLib` displays in the *Library* field, `IQ_demod_BB` displays in the *Cell* field and `symbol` displays in the *View* field. The CDF parameters for the IQ demodulator and their default values display at the bottom of the form.



4. Move the cursor over the Schematic window. The outline for the IQ demodulator symbol is attached to the cursor. Align the input pins of the IQ demodulator with the output pins of the filter and click to place the *IQ_demod_BB* block to the right of the second *BB_butterworth_bp* block. Click *Esc* to remove the symbol from the cursor.

Grounding the phase_err Pin on the IQ Demodulator

It is necessary to ground the phase error (*phase_err*) pin on the bottom of the IQ demodulator.



1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

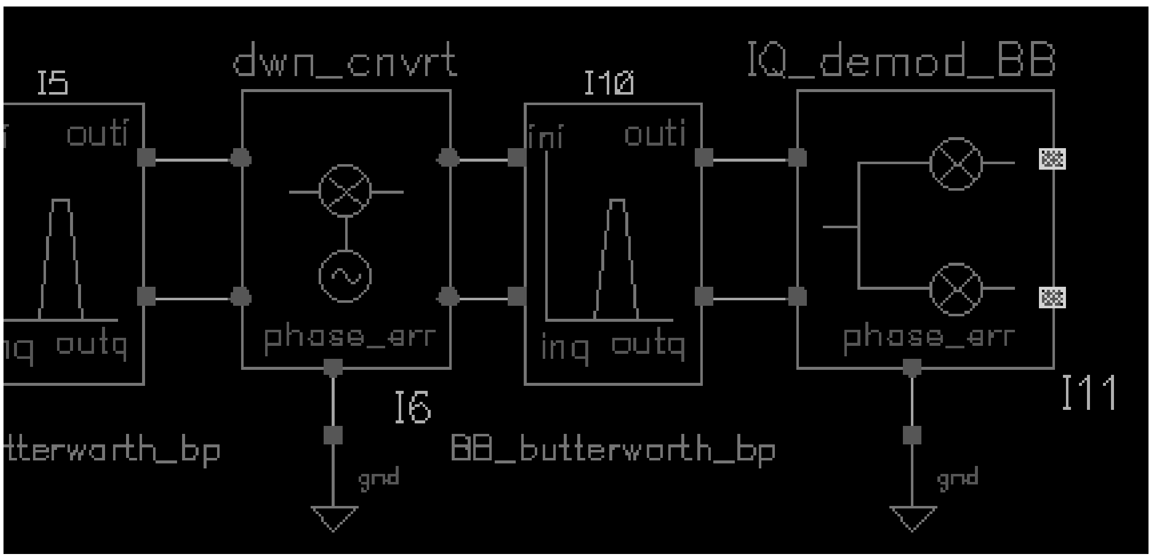
2. In the Add Instance form, type
   - ❑ `analogLib` in the *Library* field
   - ❑ `gnd` in the *Cell* field
   - ❑ `symbol` in the *View* field.
3. Move the cursor over the Schematic window.
4. Click to place the ground terminal in line with the phase error node (`phase_err`) on the bottom of the *IQ_demod_BB* block.
5. Click *Esc* to remove the symbol from the cursor.

Wiring the Filter and Ground to the IQ Demodulator

1. To wire the *IQ_demod_BB* block to the *BB_butterworth_bp* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.

2. Click `outi` on the *BB_butterworth_bp* block then click `I_in` on the *IQ_demod_BB* block.
3. Click `outq` on the *BB_butterworth_bp* block then click `Q_in` on the *IQ_demod_BB* block.
4. Click the port on the *gnd* block then click `phase_err` on the *IQ_demod_BB* block.
5. Click *Esc* to stop wiring.

   The schematic now appears as follows.



Modifying Parameter Values for the IQ Demodulator

1. Edit the CDF parameter values for the IQ Demodulator (*IQ_demod_BB*) as listed in Table 12-6 on page 957.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *IQ_demod_BB* and modify the schematic for this simulation.

   b. In the Schematic window, click on *IQ_demod_BB*.

   The Edit Object Properties form changes to display information for *IQ_demod_BB*.

   c. Change the parameter values to match those in Table 12-6.

d. Click OK.

Table 12-6 CDF Parameter Values for the IQ Demodulator

Parameter Name	Value
available I-mixer gain[dB]	0
available Q-mixer gain[dB]	0
Input resistance	50
output resistance	50
I–[dBm] input referred IP3	40
Q–[dBm] input referred IP3	40
noise figure [dB]	2
quadrature error	0
I–sharpness factor	2
Q–sharpness factor	2
I_cmp	–30
Q_cmp	–30
I–radians@I_cmp	.7
Q–radians@Q_cmp	.7
I–radians@big I-input	2
Q–radians@big Q-input	2
I {1,0,–1} for {cw,none,ccw}	0
Q {1,0,–1} for {cw,none,ccw}	0

Adding Two Butterworth Lowpass Filters

1. Add two Butterworth low pass filters to the right of the IQ demodulator block.
   - ❑ Align one filter with the demodulator’s `i_out` pin.
   - ❑ Align the other filter with its `q_out` pin.
2. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

3. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

4. In the Library Browser - Add Instance form, make the following selections.

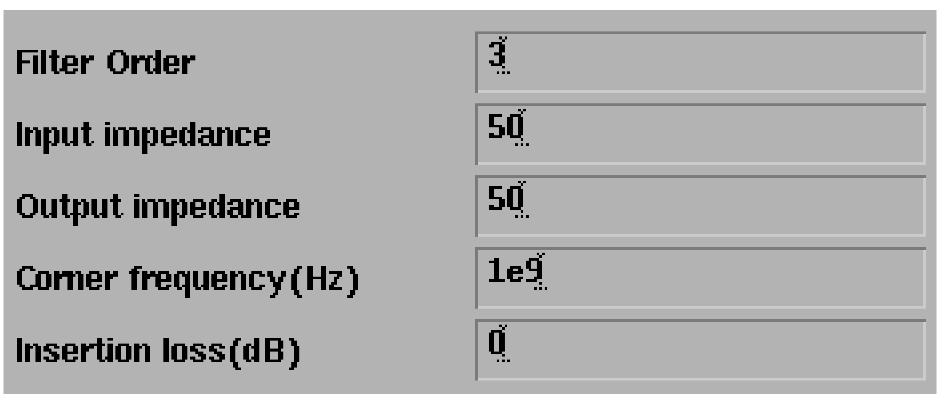
Library	Category	Cell	View
rfLib	*top_dwnPB*	*butterworth_lp*	*symbol*

In the Library Browser, cell *butterworth_lp* and it's default view are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library* field
- ❑ `butterworth_lp` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field.

The CDF parameters for the Butterworth low pass filter and their default values display at the bottom of the Add Instance form.



5. Move the cursor over the Schematic window. The outline for the butterworth low pass filter symbol is attached to the cursor.

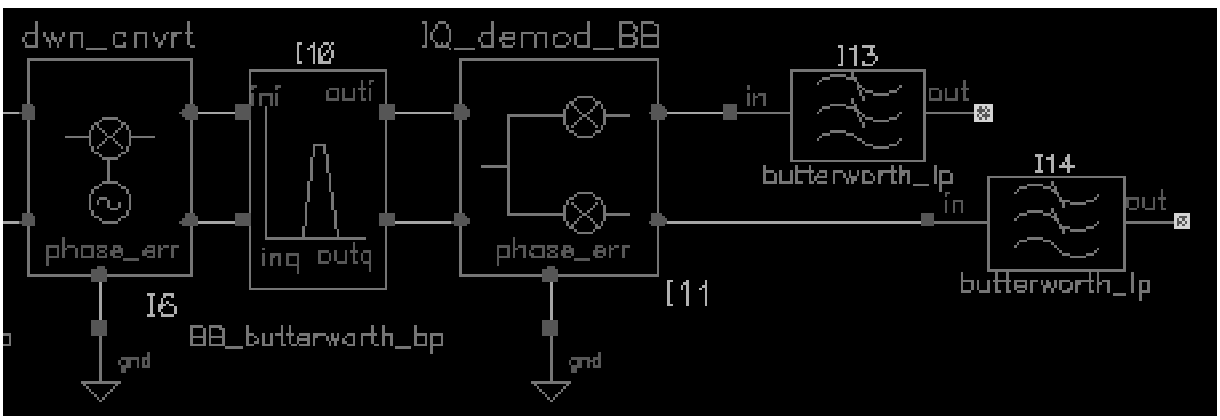
   Align the *in* pin of the first butterworth low pass filter with the *I_out* pin (the top pin) of the IQ demodulator and click to place the filter close to the demodulator. Align the *in* pin of the second butterworth low pass filter with the *Q_out* pin (the bottom pin) of the IQ demodulator. You will have to place it further from the demodulator in order to align it with the *Q_out* pin.

6. Click *Esc* to remove the symbol from the cursor.

Wiring the IQ Demodulator to the Filters

1. To wire the *IQ_demod_BB* block to the *butterworth_lp* blocks, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `I_out` on the *IQ_demod_BB* block then click `in` on the first *butterworth_lp* block.
3. Click `Q_out` on the *IQ_demod_BB* block then click `in` on the second *butterworth_lp* block.
4. Click *Esc* to stop wiring.

The schematic now appears as follows.



Modifying Parameter Values for Both Low Pass Filters

1. Edit the CDF parameter values for the Butterworth low pass filters as listed in Table 12-7 on page 960.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

   The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for each filter and modify the schematic for this simulation.

   b. In the Schematic window, click on the first low pass filter.

   The Edit Object Properties form changes to display information for the filter.

c. Change the parameter values to match those in Table 12-7.

Table 12-7 CDF Parameter Values for the Butterworth Low Pass Filters

Parameter Name	Value
Filter Order	3
Input impedance	50
Output impedance	50
Corner frequency (Hz)	10M
Insertion loss (dB)	0

d. Click Apply.

e. In the Schematic window, click on the second low pass filter.

The Edit Object Properties form displays the information you entered for the filter as shown in Table 12-7.

f. Click OK.

Adding an Instrumentation Block

Add an instrumentation block (*offset_comms_instr*) block to the right of the low pass filter blocks.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click Show Categories at the top of the Library Browser, to display the Category column.)

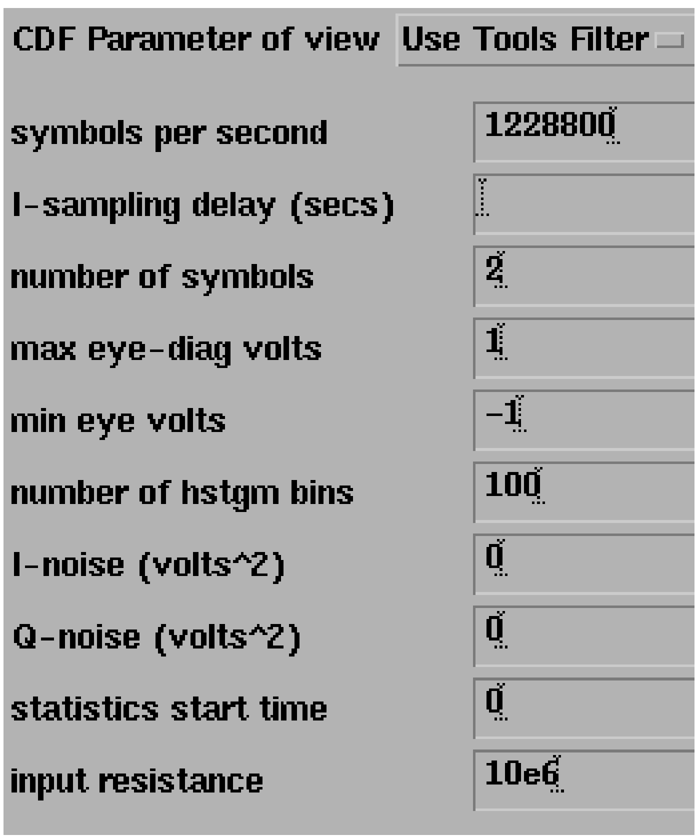
Library	Category	Cell	View
rfLib	*measurement*	*offset_comms_instr*	*symbol*

In the Library Browser, cell *offset_comms_instr* and it’s default view *symbol* are both selected.

At the top of the Add Instance form,

- ❑ `rfLib` displays in the *Library*
- ❑ `offset_comms_instr` displays in the *Cell* field
- ❑ `symbol` displays in the *View* field

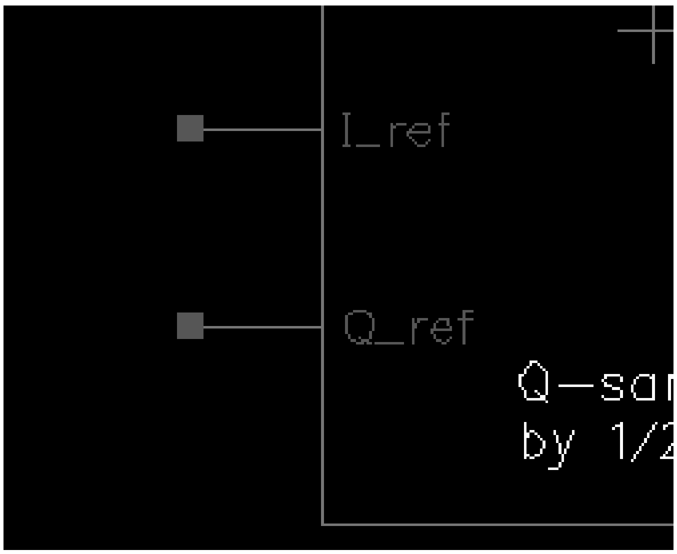
The CDF parameters for the *offset_comms_instr* and their default values display at the bottom of the form.



4. Move the cursor over the Schematic window. The outline for the *offset_comms_instr* block symbol is attached to the cursor. Align the *I_in* and *Q_in* pins of the *offset_comms_instr* block with the *out* pins of the butterworth low pass filters and click to place the *offset_comms_instr* block to the right of the low pass filter blocks.
5. Click *Esc* to remove the symbol from the cursor.

Grounding the Reference Pins on the Instrumentation Block

It is necessary to ground the reference pins (*I_ref* and *Q_ref*) pin near the lower left corner of the instrumentation block.



Grounding the phase_err Pin on the Mixer

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, type
   - ❑ `analogLib` in the *Library* field
   - ❑ `gnd` in the *Cell* field
   - ❑ `symbol` in the *View* field.

3. Move the cursor over the Schematic window.

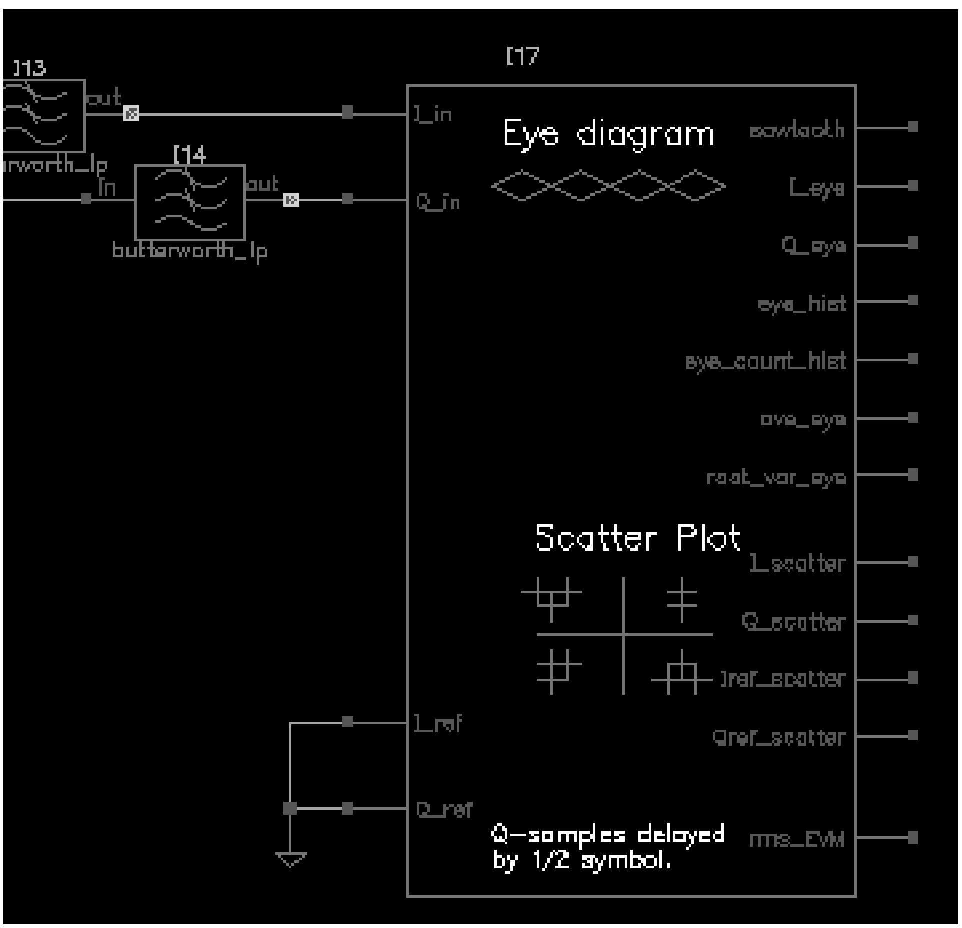
4. Click to place the ground terminal in line with the `Q_ref` node on the bottom of the *offset_comms_instr* block.

5. Click *Esc* to remove the symbol from the cursor.

Wiring the Filter and Ground to the Instrumentation Block

1. To wire the low pass filters to the *offset_comms_instr* block and the ground, in the Schematic window choose *Add - Wire (narrow)*.
2. Click `out` on the upper *butterworth_lp* block (the low pass filter connected to the `I_out` node on the IQ demodulator) then click `I_in` on the *offset_comms_instr* block.
3. Click `out` on the lower *butterworth_lp* block (the low pass filter connected to the `Q_out` node on the IQ demodulator) then click `Q_in` on the *offset_comms_instr* block.
4. Click the port on the *gnd* block then click the `Q_ref` node on the *offset_comms_instr* block.
5. Click the port on the *gnd* block then click the `I_ref` node on the *offset_comms_instr* block.
6. Click *Esc* to stop wiring.

   The schematic now appears as follows.



Modifying Parameter Values for the Instrumentation Block

1. Edit the CDF parameter values for the *offset_comms_instr* as listed in Table 12-8 on page 964.

   a. Choose *Edit – Properties – Objects* in the Schematic window.

      The Edit Object Properties form appears. You use this form to change the list of CDF (component description format) properties for *offset_comms_instr* and modify the schematic for this simulation.

   b. In the Schematic window, click on the *offset_comms_instr* block.

      The Edit Object Properties form changes to display information for *offset_comms_instr* block.

   c. Change the parameter values to match those in Table 12-8.

   d. Click OK.

Table 12-8 CDF Parameter Values for the Instrumentation Block

Parameter Name	Value
symbols per second	1228800
I-sampling delay (secs)	134n
number of symbols	2
max eye-diag volts	1
min eye volts	–1
number of hstgm bins	100
I-noise (volts^2)	0
Q-noise (volts^2)	0
statistics start time	30u
input resistance	50

Adding an Instrumentation Terminator

Add an instrumentation termination (*instr_term*) block to the right of the instrumentation block. The *instr_term* block terminates the outputs on the instrumentation block and prevents unused pin warnings.

1. In the Schematic window, choose *Add – Instance* to display the Add Instance form.

   The Add Instance form appears. It may be empty or it may display information for a previously added element.

2. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

3. In the Library Browser - Add Instance form, make the following selections. (If necessary, click Show Categories at the top of the Library Browser, to display the Category column.)

Library	Category	Cell	View
rfLib	*measurement*	*instr_term*	*symbol*

   In the Library Browser, cell *instr_term* and it's default view *symbol* are both selected.

   At the top of the Add Instance form,

   ❑ `rfLib` displays in the *Library*

   ❑ `instr_term` displays in the *Cell* field

   ❑ `symbol` displays in the *View* field

   There are no CDF parameters for the *instr_term* cell.

4. Move the cursor over the Schematic window. The outline for the *instr_term* symbol is attached to the cursor. Move the *instr_term* block to the right of the instrumentation block and align the input pins of the instrumentation termination block with the output pins of the instrumentation block.

5. Click to place the *instr_term* block.

6. Click *Esc* to remove the symbol from the cursor.

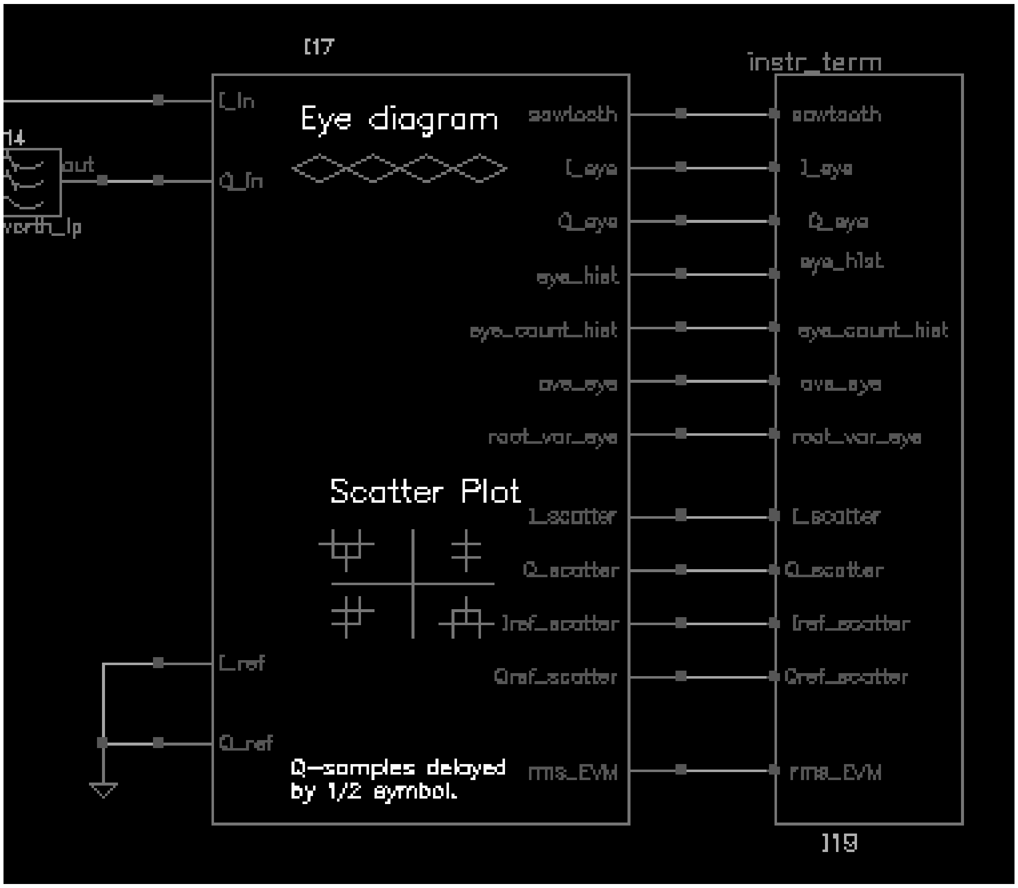
Wiring the Termination Block to the Instrumentation Block

1. To wire the instrumentation (*offset_comms_instr)* block to the Instrumentation terminator (*instr_term)* block, in the Schematic window choose *Add - Wire (narrow)*.

2. Wire the aligned pins straight across.

3. Click *Esc* to stop wiring.

The schematic should look as follows.

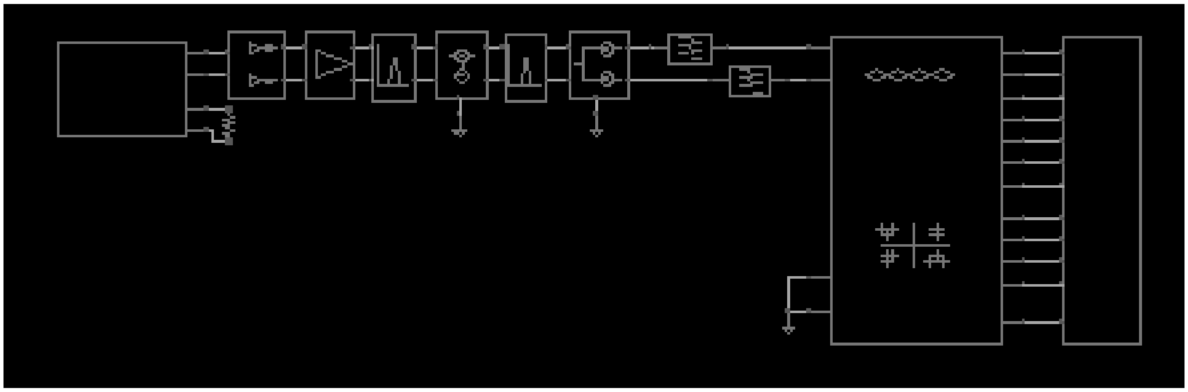


4. In the Schematic window, choose *Design — Design Check and Save*.

The completed schematic is verified and saved.

The schematic for the complete receiver model should look like the one in Figure 12-7 on page 967.

Figure 12-7 Completed receiver model

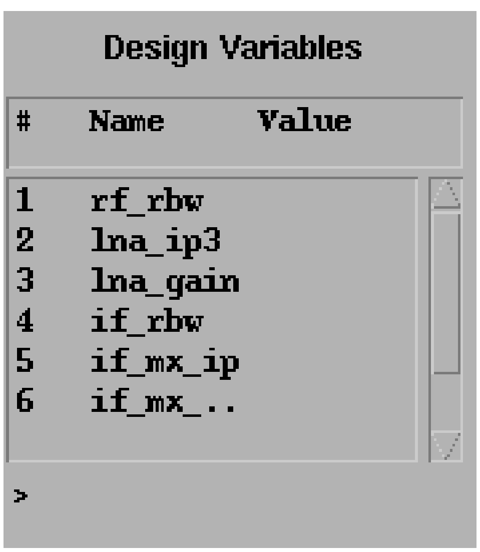


Setting Variable Values for the Receiver Schematic

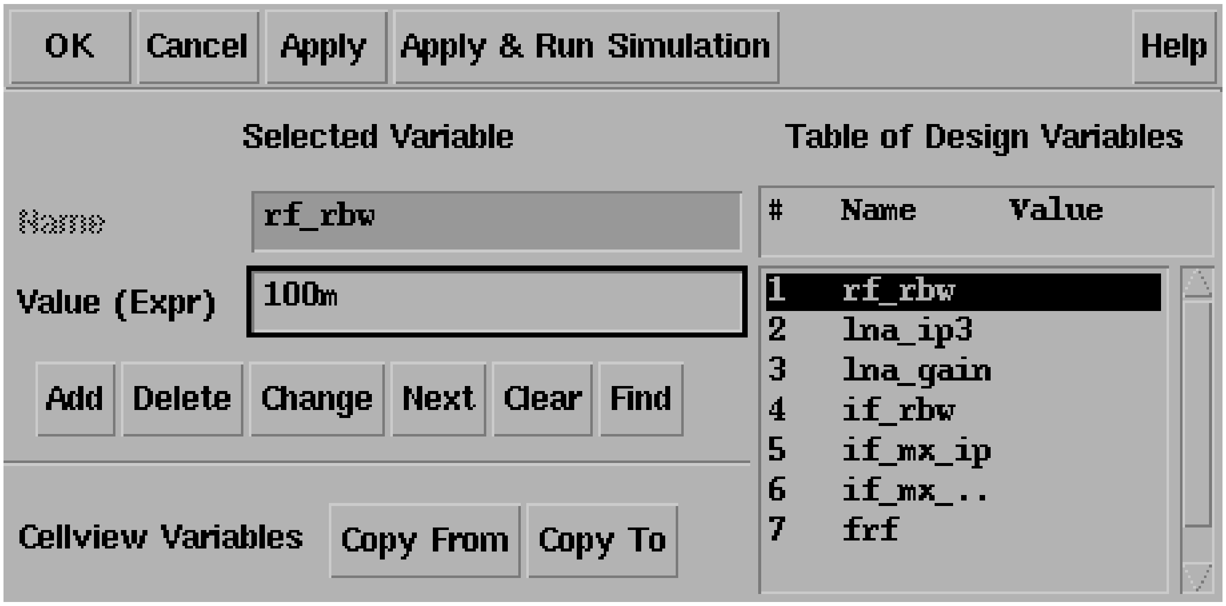
Copy the variables you entered as CDF parameters for the individual blocks from the receiver schematic to the Simulation window. Then edit each variable to give it the value specified in Table 12-9 on page 968.

1. In the Simulation window, use *Variables — Copy From Cellview* to copy the variables from the receiver schematic to the Design Variables area on the Simulation window.

   The copied variables display in the *Design Variables* area in the Simulation window.



2. In the Simulation window, choose *Variables — Edit* to open the Editing Design Variables form.



Adding the Values to the Copied Variables

In the Editing Design Variables form, one by one, select each variable in the *Table of Design Variables* and associate with each one, the value listed in Table 12-9.

Table 12-9 Values for Receiver Variables

Variable	Value
`lna_gain`	15
`lna_ip3`	–5
`if_mx_gain`	10
`if_mx_ip`	35
`frf`	2.14G
`flo1`	2.354G
`if_rbw`	200m
`rf_rbw`	100m

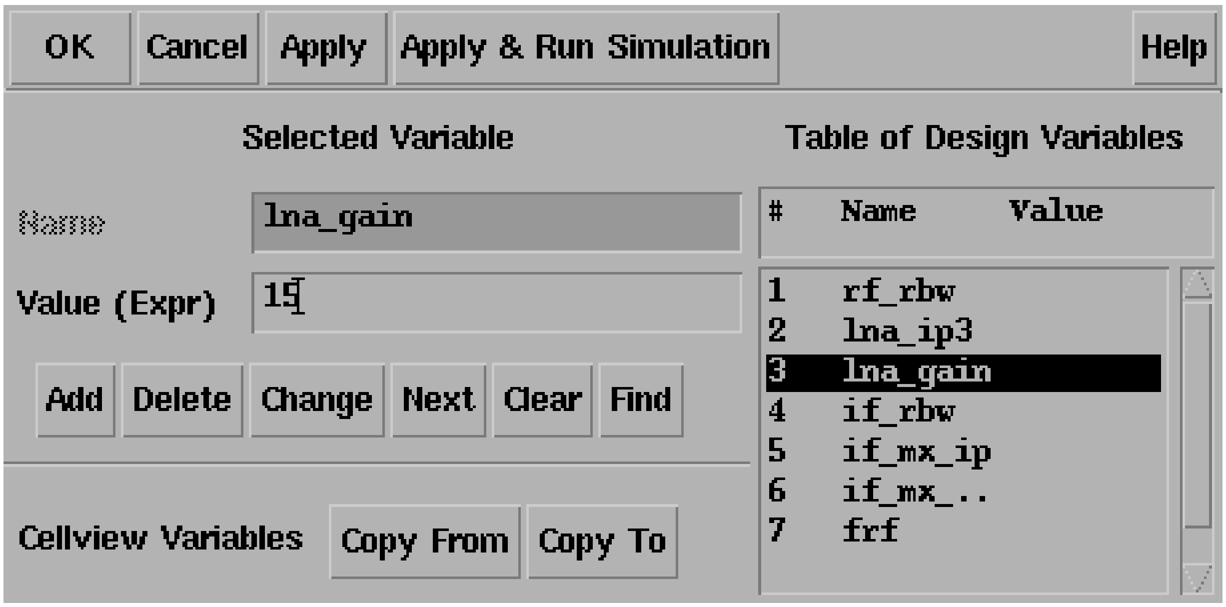
To associate a value with a design variable

1. In the *Table of Design Variables*, click `lna_gain`.

   `lna_gain` displays in the *Name* field.

2. In the *Value (Expr)* field, enter the number `15,` the value from Table 12-9 on page 968.

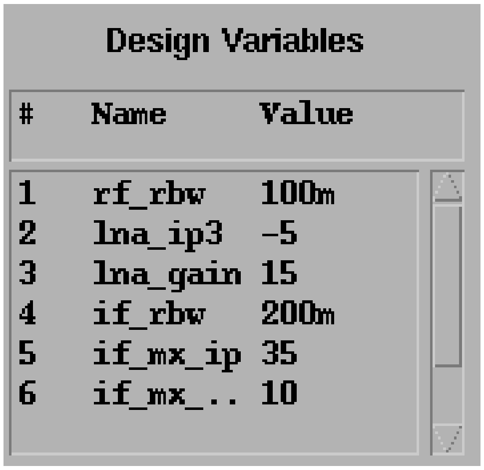
3. Click *Change* to list the variable name and its value from the *Table of Design Variables.*



4. Repeat these steps for the remaining variables listed in the *Table of Design Variables* to associate the values from Table 12-9 on page 968 with the variable names.

5. Click *OK* in the Editing Design Variables form after you have added all the variable values.

The table of *Design Variables* in the Simulation window is updated and the Editing Design Variables form is closed.

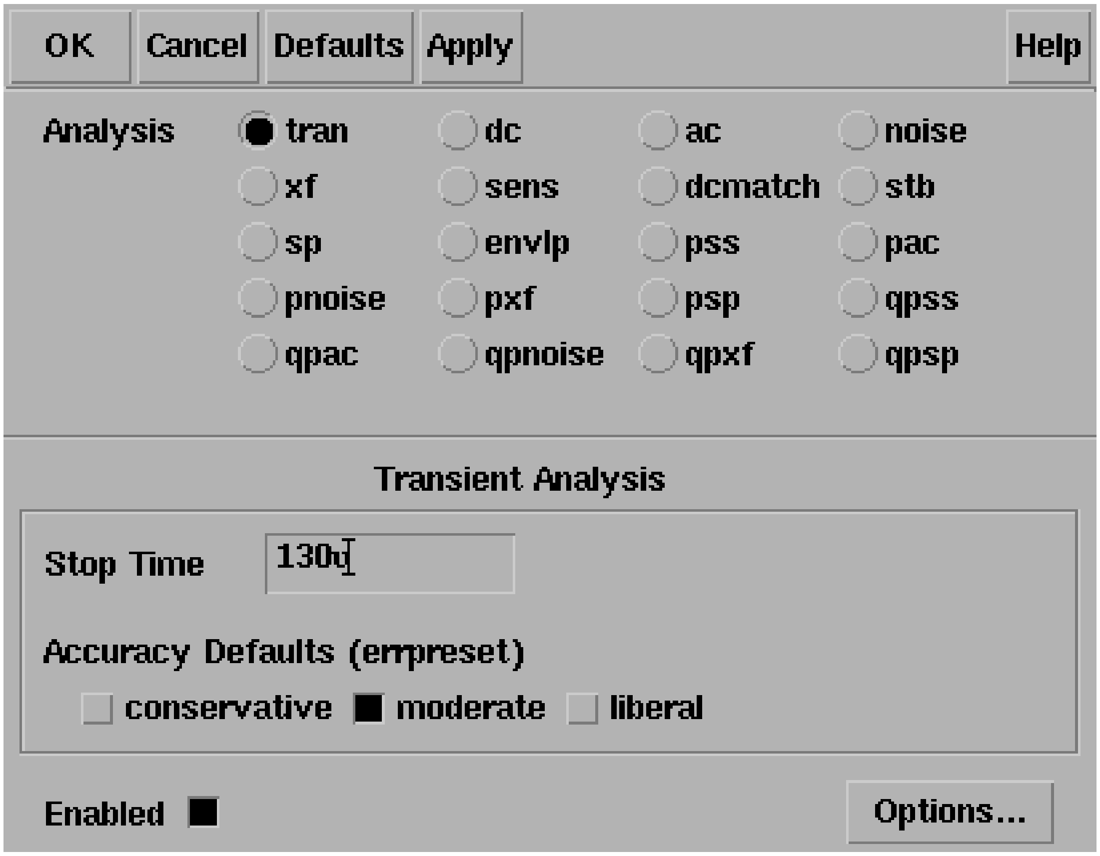

Design Variables

#	Name	Value
1	rf_rbw	100m
2	lna_ip3	-5
3	lna_gain	15
4	if_rbw	200m
5	if_mx_ip	35
6	if_mx_..	10

Setting Up and Running a Transient Analysis

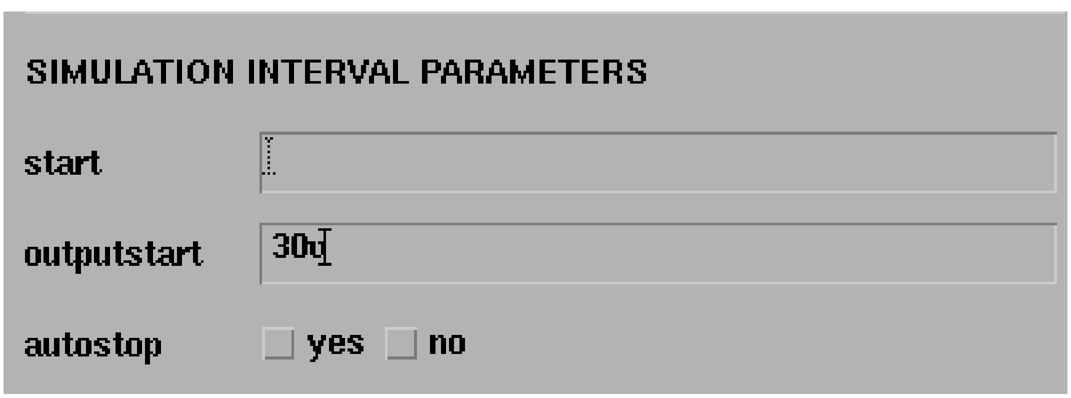
1. In the Simulation window, choose *Analyses—Choose* to display the Choosing Analyses form.
2. In the Choosing Analyses form, if necessary, click *tran* to select a transient analysis.
3. In the Choosing Analyses form, enter `130u` in the *Stop Time* field.

4. Highlight *moderate* for *Accuracy Defaults (errpreset)*.



5. In the Choosing Analyses form, click *Options* to display the Transient Options form.

6. In the Transient Options form, enter `30u` in the *outputstart* field.



By delaying the output start, you remove start-up transients from the eye-diagrams and scatter plots.

7. Click *OK* in the Transient Options form.
8. Click *OK* in the Choosing Analyses form.
9. If you have not already done so, set up the simulator and model libraries with the following steps.
   a. Set the simulator options from the Simulator window as described in "Choosing Simulator Options" on page 912.
   b. Set up the model libraries from the Simulator window as described in "Setting Up Model Libraries" on page 914.
10. In the Simulation window, choose *Simulation—Netlist and Run*.

    Messages display in the CIW. The simulation log window opens. Watch for messages stating that the simulation has completed successfully.

    Watch the CIW for messages stating the simulation is running and that it has completed successfully.

Examining the Results: Eye Diagram, Histogram, and Scatter Plot

In this section we examine the results of the transient analysis of the receiver.

Plotting the Eye Diagram (and Transient Response)

First plot an eye diagram.

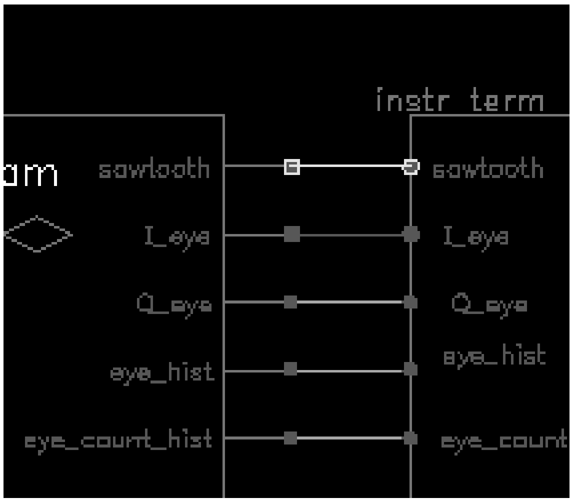
1. In the Simulation window, choose *Results—Direct Plot—Transient Signal*.

   This displays the Waveform window.
2. Following the prompts at the bottom of the Waveform window,

   > Select nodes or terminals, press <esc> to finish selection

   In the Schematic window:
   a. Click the *sawtooth* net from the instrumentation (*offset_comms_instr)* block.

b. Click the *I_eye* net from *offset_comms_instr* block.



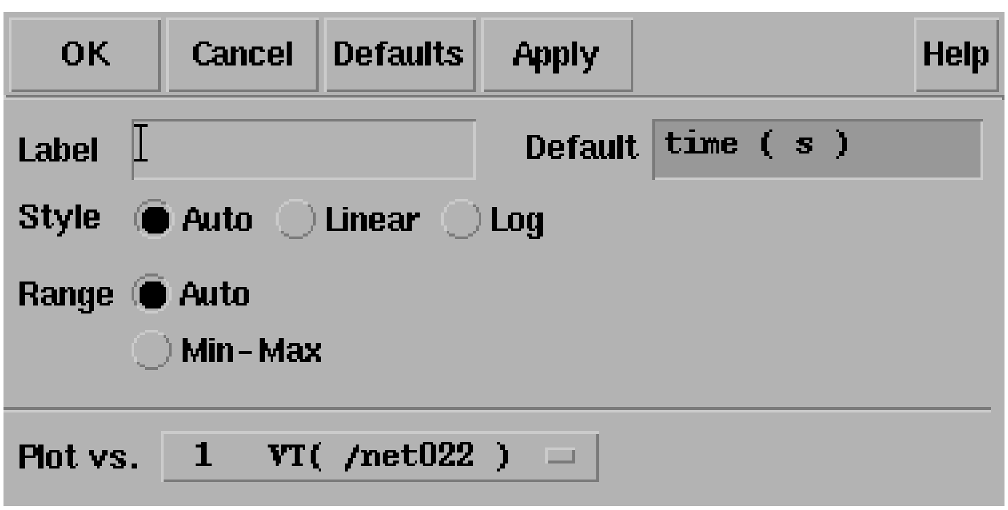
c. Press *Esc* to indicate that you have finished selecting outputs.

This creates a plot of Transient Response in the Waveform window.

3. In the Waveform window, select *Axes — X Axis.*

The X Axis form displays. In the X Axis form:

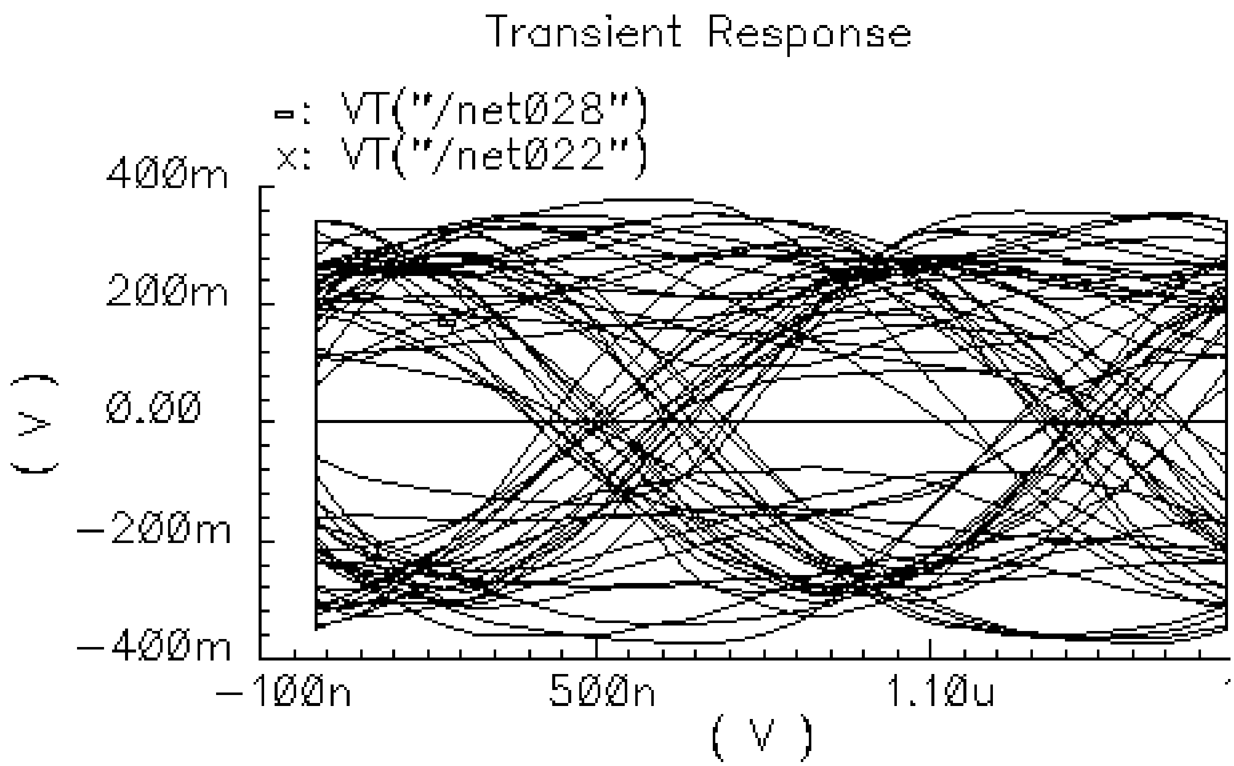
a. In the *Plot vs.* cyclic field, select *1 VT(/net022)* to plot the *sawtooth* output on the x-axis. (The net number you see may be different.)



b. Click *OK*.

You should see the eye-diagram shown in Figure 12-8 on page 974.

Figure 12-8 Eye-diagram



The I-sampling delay parameter in the instrumentation block (*I_del*) is chosen with respect to this eye diagram. The delay is the time when the eye opens the widest.

The instrumentation block will sample the input waveforms with this delay to compute all statistics and to produce scatter plots.

4. In the Waveform window, choose *Window — Close*.

Generate the Histogram

Now generate a histogram of the I-voltage at the sampling times.

1. In the Waveform window, choose *Window—Reset* to clear the Waveform window.
2. In the Simulation window, choose *Results—Direct Plot—Transient Signal*.

   This displays the Waveform window.
3. Following the prompts at the bottom of the Waveform window,

   In the Schematic window:

   a. Click the *eye_hist* net from the instrumentation (*offset_comms_instr)* block.

   b. Click the *eye_count_hist* net from *offset_comms_instr*.

   c. Press *Esc* to indicate that you have finished selecting outputs.

   This creates a plot in the Waveform window.

4. In the Waveform window, select *Axes — X Axis.*

The X Axis form displays. In the X Axis form:

   a. In the *Plot vs.* cyclic field, select *1 VT(/net029)* to plot the *eye_hist* output on the x-axis.

   b. Click *OK*.

This creates a plot in the Waveform window.

5. In the Waveform window, select *Curves — Options.*

The Plot Style form displays. In the Plot Style form,

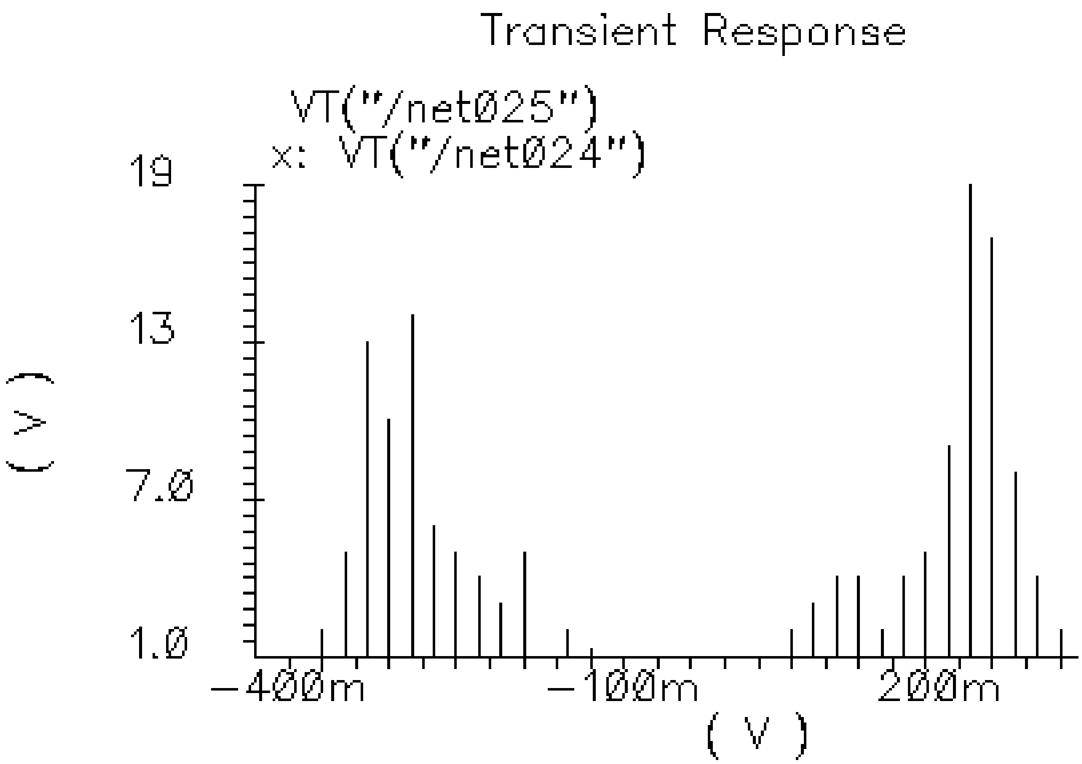
   a. In the *Plotting Style* cyclic field, select *bar*.

   b. In the *Number of Ticks* field, enter `0`.

   c. Click *OK*.

You should see the histogram in Figure 12-9 on page 976.

Figure 12-9 Histogram



6. In the Waveform window, choose *Window — Close*.

Generating the Scatter Plot

Generate a scatter plot of the received symbols.

1. In the Simulation window, choose *Results—Direct Plot—Transient Signal*.

   This displays the Waveform window.

2. Following the prompts at the bottom of the Waveform window,

   In the Schematic window:

   a. Click the *I_scatter* output from the instrumentation (*offset_comms_instr)* block.

   b. Click the *Q_scatter* output from *offset_comms_instr*.

   c. Click *Esc* to indicate that you have finished selecting outputs.

      This creates a plot in the Waveform window.

3. In the Waveform window, select *Axes — X Axis.*

The X Axis form displays. In the X Axis form:

a. In the *Plot vs.* cyclic field, select *1 VT(/net034)* to plot the *I_scatter* output on the x-axis.

b. Click *OK*.

4. In the Waveform window, select *Curves — Options.*

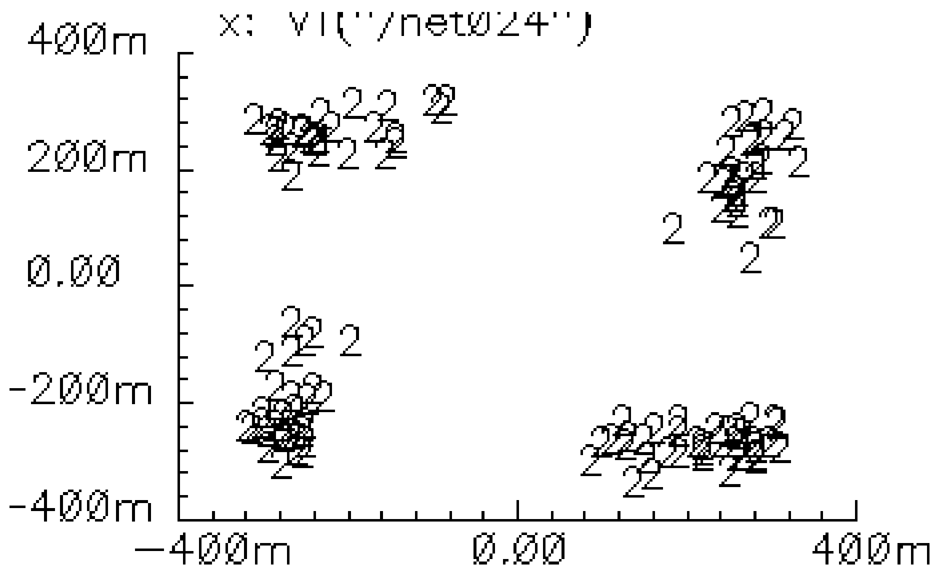
The Plot Style form displays. In the Plot Style form:

a. in the *Plotting Style* cyclic field, select *Data Points Only*.

b. Click *OK*.

You should see the scatter plot shown in Figure 12-10 on page 977.

Figure 12-10 Scatter Plot



5. In the Waveform window, choose *Window — Close*.

The Various Instrumentation Blocks

The CDMA source (*CDMA_reverse_xmit)* produced offset QPSK symbols. Offset QPSK modulation avoids traversing the origin by staggering the digital changes in the I and Q signals. Running the baseband trajectory through the origin increases spectral regrowth in the transmitters.

The instrumentation block (*offset_comms_instr*) samples the I and Q signals at different times then plots the two staggered samples against each other. The resulting scatter plot shows the received symbols. A scatter plot of the unstaggered samples reveals only what is happening in one dimension, either the I or Q dimension.

For non-offset QPSK and QAM modulation schemes, use the *comms_instr* instrumentation block instead of the *offset_comms_instr* block.

Measuring RMS EVM

You can use the same instrumentation block (*offset_comms_instr*) to compute root-mean-squared error vector magnitude (RMS EVM). The error vector is the vectorial difference between the ideal received symbol and the actual received symbol.

- EVM (error vector magnitude) is the magnitude of the error vector.
- RMS EVM is the root-mean-squared value of a sequence of EVMs.

RMS EVM is one measure of a receiver's quality. RMS EVM can account for as much or as little distortion and noise as you like. The trick is to figure out where the ideal received symbol lies. You can do this using the *I_ref* and *Q_ref* inputs to the *offset_comms_instr* instrumentation block.

To calculate RMS EVM, you

- Create a duplicate copy of the receiver chain from the *BB_driver* to the *IQ_demod_BB* including these two blocks
- Place the duplicate copy below the original receiver chain in the Schematic window
- Modify the duplicate receiver chain to make it as *ideal* as you like by changing parameter values for the individual function blocks.

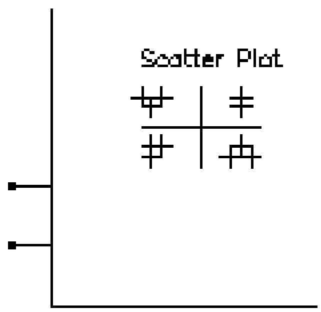
  For example, to see the effect of just the LNA's IP3 value on RMS EVM, in the duplicate receiver chain make the LNA's IP3 absurdly large.

Constructing the Ideal Receiver Chain

The *ideal* receiver chain (the duplicated and modified receiver chain) is driven from the same input, the *CDMA_reverse_xmit* block*,* as the original receiver chain. The output of the ideal receiver chain drives the instrumentation block's *I_ref* and *Q_ref* inputs.

In the ideal receiver chain, you copy the first receiver chain and make every block ideal.

1. Remove the *gnd* from the *I_ref* and *Q_ref* pins on the instrumentation block.
   - **a.** In the Schematic window, choose *Edit - Delete*.
   - **b.** In the Schematic window, click on each wire and the *Gnd* symbol attached to the *I_ref* and *Q_ref* pins.

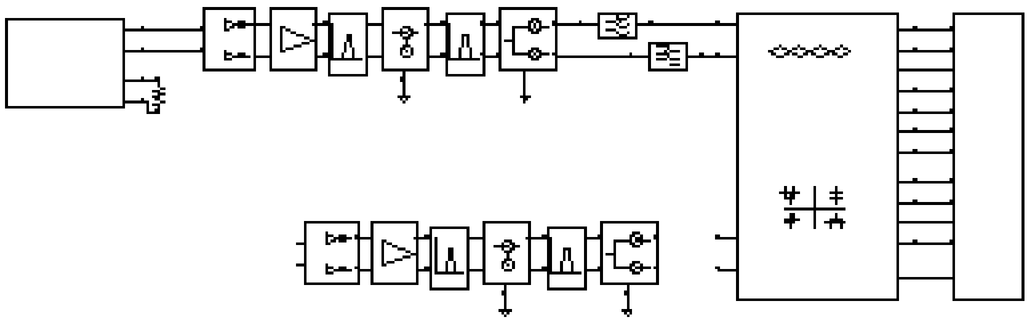


   - **c.** Click *esc* to stop deleting.
2. Duplicate the receiver chain from the *BB_driver* to the *IQ_demod_BB* inclusive. Do not duplicate the filters. Follow the prompts at the bottom of the Schematic window.
   - **a.** In the Schematic window, choose *Edit—Copy*.
   - **b.** In the Schematic window, draw a box around the blocks to copy by clicking to the left of and above the *BB_driver* block and dragging the cursor to a point below and to the right of the *IQ_demod_BB* block. Click again to complete the box.

     The blocks within the box are highlighted.
   - **c.** Click within the highlighted area.

     A copy of the highlighted blocks in the receiver chain now moves with the cursor.

d. Place the duplicate receiver chain so that the output pins on the *IQ_demod_BB* are in line with the *I_ref* and *Q_ref* pins on the instrumentation block.



3. Wire the duplicate receiver chain to the CDMA signal source (*CDMA_reverse_xmit)* and the instrumentation block (*offset_comms_instr*).

   a. In the Schematic window, choose *Add - Wire (narrow).*

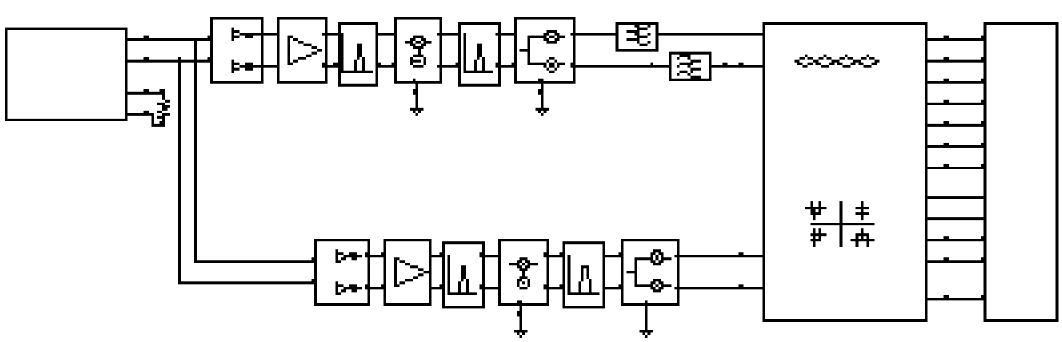
   b. Connect the *IQ_demod_BB* outputs on the duplicate receiver to the *I_ref* and *Q_ref* pins on the instrumentation block.

   c. Connect the output pins on *CDMA_reverse_xmit* to the input pins of the duplicate *BB_driver*. This drives the duplicate receiver from the CDMA signal source.

   d. Click *esc* to stop wiring.

The schematic with the duplicate receiver chain wired up looks like Figure 12-11.

Figure 12-11 Receiver Model with Duplicated Receiver Chain



Modifying Parameter Values to Make the Blocks Ideal

Now modify the parameter values for each block in the duplicate receiver chain to create ideal blocks. Block names, parameter names, and parameter values are given in Table 12-10.

Table 12-10 Parameter Values to Create an Ideal Receiver

Block Names	Parameter Names	New Parameter Values
`LNA_BB`	*Input referred IP3 [dBm]* *{1,0,-1} for {cw,none,ccw}*	`100` `0`
`BB_butterworth_bp`	*Cell Name*	`BB_loss`
`dwn_cnvrt`	*Input referred IP3 [dBm]*	`100`
`BB_butterworth_bp`	*Cell Name*	`BB_loss`
`IQ_demod_BB`	*I-[dBm] Input referred IP3* *Q-[dBm] Input referred IP3*	`100` `100`

1. In the Schematic window, choose *Edit—Properties—Objects* to open the Edit Object Properties form.
2. In the Schematic window, select the *LNA_BB* block.

   The Edit Object Properties form changes to display properties for the *LNA_BB* block.

   a. Set *Input referred IP3 [dBm]* to 100.

   b. Set *{1, 0,-1} for {cw, none, ccw}* to 0. (This eliminates AM/PM conversion.)

   c. Click *Apply*.
3. In the Schematic window, select the first RF *BB_butterworth_bp* block.

   The Edit Object Properties form changes to display properties for the *BB_butterworth_bp* block.

   a. Change the *Cell Name* to *BB_loss*.

   b. Click *Apply*.

   The properties and symbol change to those for the *BB_loss* block. The sole purpose of the *BB_loss* model is to replace a filter in an RMS EVM analysis.

The *Reference impedance* for the *BB_loss* block should equal the *Output impedance* of the *BB_butterworth_bp* bandpass filter block it replaces. The value should be 50 ohms for both blocks and you should not have to change it.

The *BB_loss* model retains the filter's loss but eliminates the filter's dynamics so you can see what, if any, affect the filter has on EVM through inter-symbol interference. To eliminate the loss as well as the dynamics, you might even replace the filter with straight wires. This example uses the *BB_loss* block instead.

4. In the Schematic window, select the *dwn_cnvrt* block.

   The Edit Object Properties form changes to display properties for the *dwn_cnvrt* block

   a. Set *Input referred IP3 [dBm]* to 100.

   b. Click *Apply*.

5. In the Schematic window, select the second *BB_butterworth_bp* block.

   The Edit Object Properties form changes to display properties for the *BB_butterworth_bp* block

   a. Change the *Cell Name* to *BB_loss*.

   b. Click *Apply*.

6. In the Schematic window, select the *IQ_demod_BB* block.

   The Edit Object Properties form changes to display properties for the *IQ_demod_BB* block

   a. Set *I-[dBm] input referred IP3* to 100.

   b. Set *Q-[dBm] input referred IP3* to 100.

   c. Click *Apply*.

7. In the Edit Object Properties form, click *OK* to close the form.

8. In the Schematic window, select *Design—Check and Save* to check and save your modifications to the circuit.

Set Up and Run a Transient Analysis

Set up and run a transient analysis as described in "Setting Up and Running a Transient Analysis" on page 970. Set the *Stop Time* to `130u` and the *outputstart* option to `30u`. Click *OK* in both the Transient Options and Choosing Analyses forms. Choose *Simulation—Netlist and Run* to run the transient analysis.

Look for messages in the CIW stating that the simulation is starting. Watch the simulation log window for messages that the simulation has completed successfully.

Plot the RMS EVM Output

After the simulation, plot the RMS_EVM output of the instrumentation block.

1. In the Simulation window, choose *Results—Direct Plot—Transient Signal*.

   This displays the Waveform window.

2. Following the prompts at the bottom of the Waveform window.

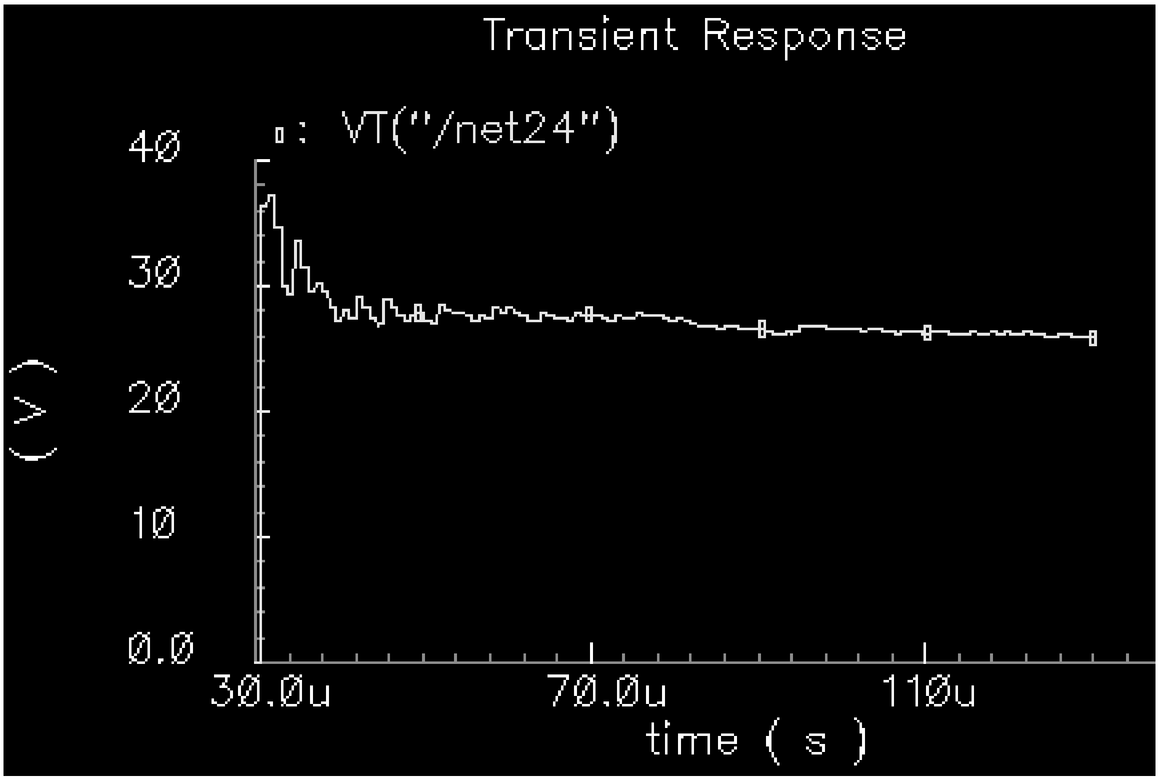
   In the Schematic window,

   a. Click the *rms_EVM* output net from the instrumentation (*offset_comms_instr)* block.

   b. Click *Esc* to indicate that you have finished selecting outputs.

      This creates the RMS EVM plot in the Waveform window as shown in Figure 12-12 on page 984.

Figure 12-12 RMS EVM



The RMS EVM trace starts at 30us, which is the `statistics start time` parameter of the instrumentation block. The `statistics start time` parameter keeps start-up transients out of the statistics.

The trace settles out at 25.84 Volts. This means that after 130us of data is collected, and ignoring the first 30us, the RMS EVM is 25.84%. The EVM measurement is normalized to the RMS magnitude of the ideal symbol then multiplied by 100 to express the measurement as a percentage.

3. In the Waveform window, choose *Window — Close*.

Computing Minimized RMS Noise Using the Optimizer

There is one more construction step before proceeding to the Circuit Optimizer application. You will set up the Circuit Optimizer to minimize RMS noise subject to performance constraints. This step replicates the receiver chain yet one more time to generate the noise measurement.

1. Duplicate the original receiver chain from the *BB_driver* up to and including both low pass filters (*butterworth_lp*). Follow the prompts at the bottom of the Schematic window.

   a. In the Schematic window, choose *Edit—Copy*.

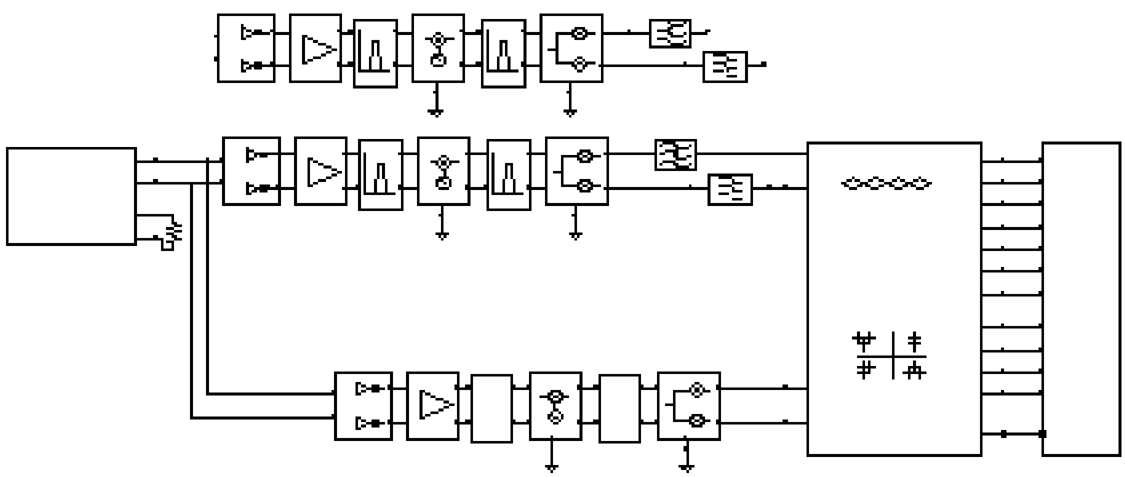
   b. In the Schematic window, draw a box around the blocks to copy by clicking to the left of and above the *BB_driver* block and dragging the cursor to a point below and to the right of the *butterworth_lp* filter blocks. Click again to complete the box.

      The blocks within the box are highlighted.

   c. Click within the highlighted area.

      A copy of the highlighted blocks in the receiver chain now moves with the cursor.

   d. Place the duplicate receiver chain above the original receiver chain.



2. In the duplicate receiver chain, ground the *Q_in* pin on the *BB_driver* block.

   a. In the Schematic window, choose *Add—Wire.*

   b. Click the *Q_in* pin and run the wire to the *Gnd* symbol below the *dwn_cnvrt* block.

   c. Click the *Gnd* symbol below the *dwn_cnvrt* block.

   d. Click *esc* to stop wiring.

3. Add a 50mV DC voltage source to the left of the *BB_driver* block to drive the *I_in* pin on the *BB_driver* block. At the same time add a *gnd* symbol below the *port* in the schematic.

a. In the Schematic window, choose *Add—Instance* to display the Add Instance form.

b. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

c. In the Library Browser - Add Instance form, click *analogLib*.

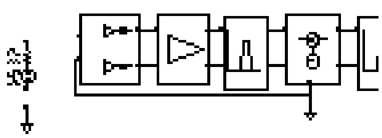
d. Scroll the elements in the *Cell* column and click *port*.

e. The outline for the *port* symbol is attached to the cursor. Move the *port* symbol to the left of the *BB_driver* block and click to place the *port* symbol.

f. Return to the Library Browser - Add Instance form and scroll the elements in the *Cell* column and click *gnd*.

g. The outline for the *gnd* symbol is attached to the cursor. Move the *gnd* symbol below the *port* symbol and click to place it there.

h. Click *Esc* to remove the *gnd* symbol from the cursor.



4. In the Schematic window, choose *Edit—Object—Properties* to modify the *port* symbol using the Edit Object Properties form.

a. In the Schematic window, click on the *port* symbol.

The Edit Object Properties form changes to display information for the *port* symbol.

b. In the *Source Type* cyclic field, select *dc*.

c. In the *DC Voltage* field, enter 50m.

d. Highlight *Display small signal params* to display small signal parameters.

e. In the *AC Magnitude* field type `1V`.

f. In the *AC Phase* field type `0`.

g. Click *OK* in the Edit Object Properties form.

5. Load the low pass filters with a *res_BB* model from the *top_dwnBB* category of *rfLib*. Use the default parameters and ground the output pins.

a. In the Schematic window, choose *Add—Instance* to display the Add Instance form.

b. In the Add Instance form, click *Browse* to display the Library Browser - Add Instance form.

c. In the Library Browser - Add Instance form, click *rfLib*.

d. Scroll the elements in the *Cell* column and click *res_BB*.

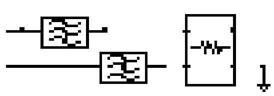
e. The outline for the *res_BB* symbol is attached to the cursor. Move the *res_BB* symbol to the right of the two low pass filters (*butterworth_lp*) and click to place the *res_BB* symbol.

f. Return to the Library Browser - Add Instance form and click *analogLib.*

g. Scroll the elements in the *Cell* column and click *gnd*.

h. The outline for the *gnd* symbol is attached to the cursor. Move the *gnd* symbol to the right of the *res_BB* symbol and click to place it there.

i. Click *Esc* to remove the symbol from the cursor.



6. In the duplicate receiver chain, wire the *port*, the *res_BB* block, and their *gnd* blocks.

a. In the Schematic window, choose *Add—Wire.*

b. Click the *I_in* pin on the *BB_driver*, then click the top pin on the *port*.

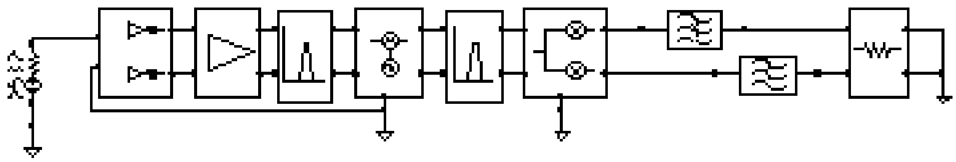
c. Click the bottom pin on the *port*, then click the *gnd* pin just below it.

d. Click the *out* pin on the top *butterworth_lp* filter, then click the *I_in* pin on the *res_BB* block.

e. Click the *out* pin on the lower *butterworth_lp* filter, then click the Q_*in* pin on the *res_BB* block.

f. Click the *I_out* pin on the *res_BB* block. Then click the top pin on the *gnd* located to it's right.

g. click the Q_*out* pin on the *res_BB* block. Then click the top pin on the same *gnd*.

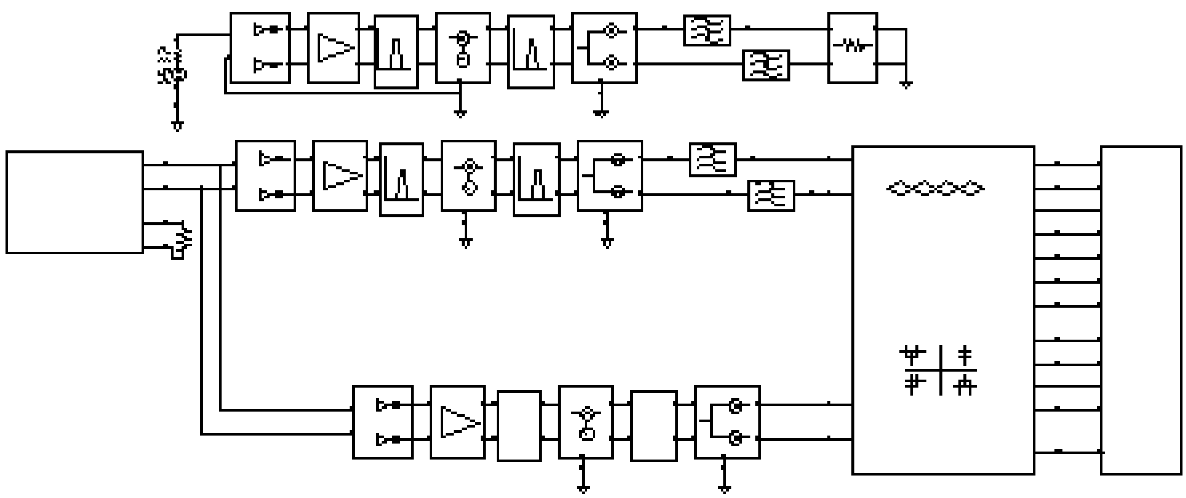


h. Click *esc* to stop wiring.

7. In the Schematic window, choose *Design—Check and Save* to check and save the schematic.

The schematic with the third receiver chain is shown in Figure 12-13.

Figure 12-13 Schematic with Noise Generating Receiver



Set Up and Run Transient and Noise Analyses

Set up a transient analysis as described in “Setting Up and Running a Transient Analysis” on page 970. Set the *Stop Time* to `130u` and the *outputstart* option to `30u` and make sure that the transient analysis is enabled.

Set up a noise analysis as follows:

1. In the Choosing Analysis form, click *noise* to select a noise analysis.

2. For *Sweep Variable*, click *Frequency*.

3. For *Sweep Range*, click *Start-Stop*.

4. Set up the analysis to sweep frequency from 0 to100 MHz.

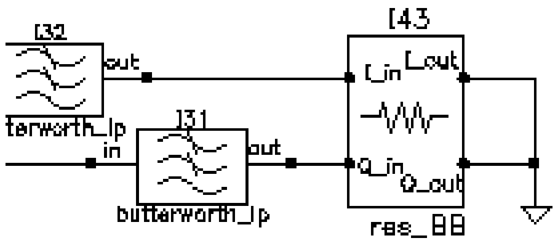
   a. For the starting frequency, in the *Start* field enter `0`.

   b. For the stop frequency, in the *Stop* field enter `100M`.

5. Set up the *Output Noise* source.

   a. In the *Output Noise* cyclic field, select *voltage.*

   b. To select the *Positive Output Node*, click *Select* next to the *Positive Output Node* field. Then, in the Schematic window, click the net next to the *I_in* pin on the *res_BB* block.

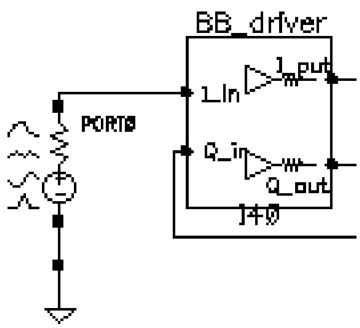


   c. To select the *Negative Output Node*, click *Select* next to the *Negative Output Node* field. Then, in the Schematic window, click the net next to the *I_out* pin on the *res_BB* block.

6. Set up the *Input Noise* source.

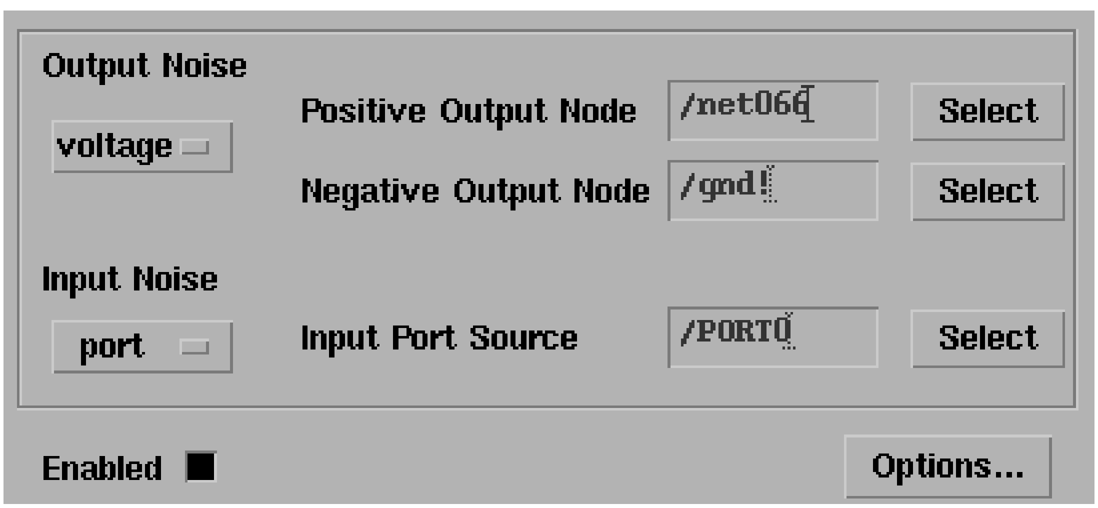
a. In the *Input Noise* cyclic field, select *port.*

b. To select the *Input Port Source*, click *Select* next to the *Input Port Source* field. Then, in the Schematic window, click the *DC input port* model.



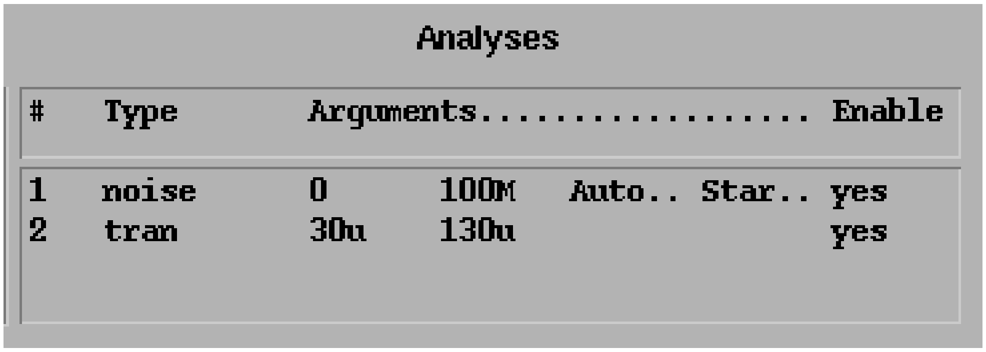
7. Click Enabled.

The *Output Noise* and *Input Noise* sections of the Noise analysis form look as follows.



8. Verify that the noise analysis is *Enabled* and click *OK* in the Choosing Analysis form.

9. In the Simulation window, check the Analysis area to verify that both the transient and noise analyses are set up properly and that they are both enabled.



Analyses

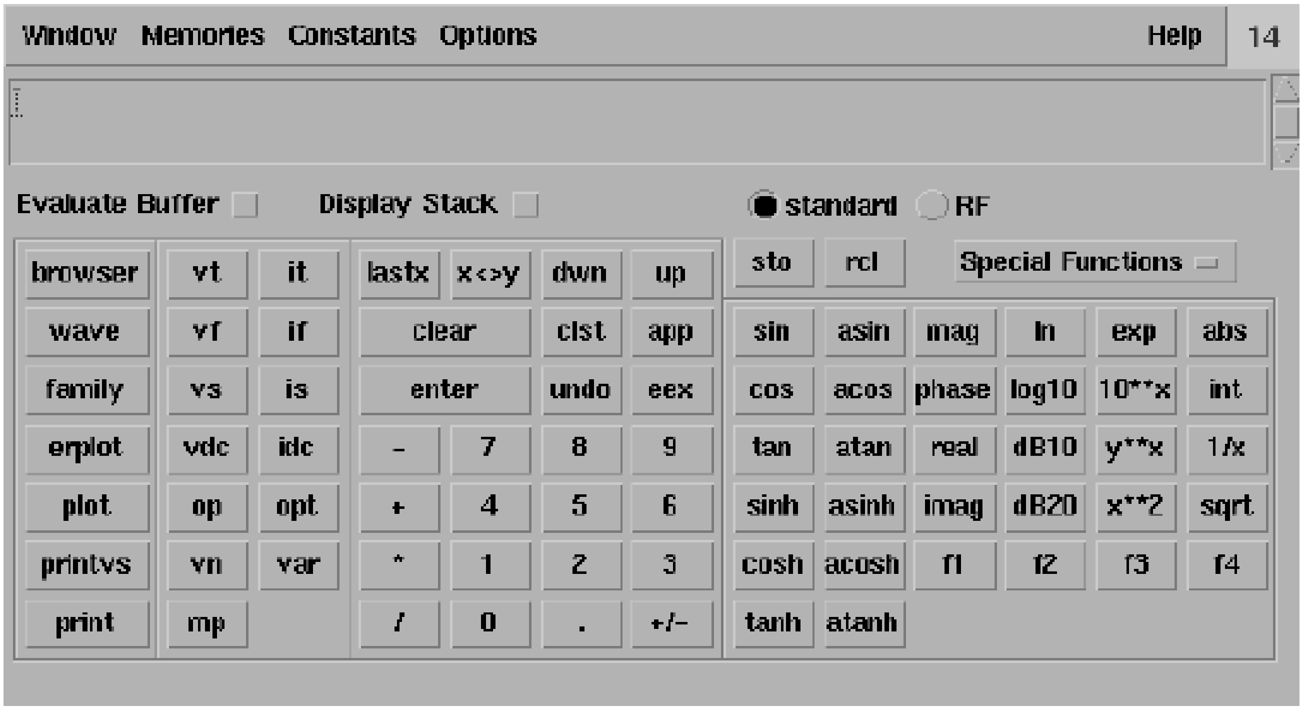
#	Type	Arguments..................				Enable
1	noise	0	100M	Auto..	Star..	yes
2	tran	30u	130u			yes

10. In the Simulation window, choose *Simulation—Netlist and Run* to start the simulations.

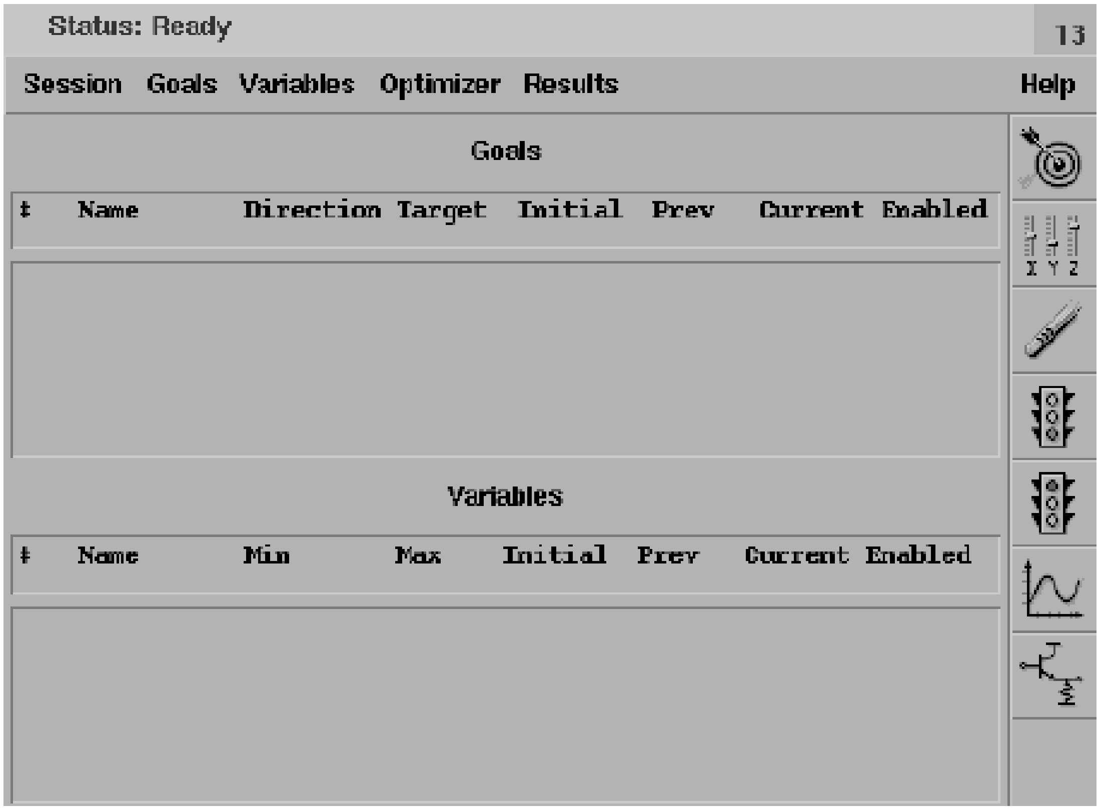
    Watch the messages in the CIW to verify that everything is set up properly and that the simulations start. Check the simulation log window to see that the simulations run and complete properly.

Set Up to Run the Circuit Optimizer

1. In the Simulation window, choose *Tools—Calculator* to open the Waveform Calculator window.



2. In the Simulation window, choose *Tools—Optimization* to open the Circuit Optimizer window.



Add the First Goal

1. In the Calculator window's *Special Function* cyclic field, select *rmsNoise*.

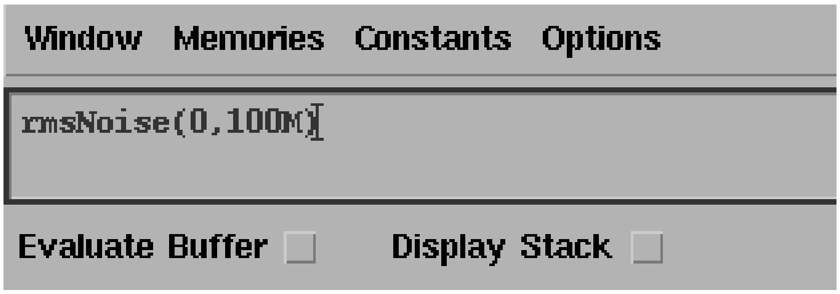
   When the RMS Noise form opens.

   a. In the *From* field, enter `0`.

   b. In the *To* field, enter `100M`.

   c. Click *OK*.

The expression `rmsNoise(0,100M)` displays in the Calculator window.



2. In the Circuit Optimizer, choose *Goals—Add* to add the first goal.

When the Adding Goals form opens

a. In the *Name* field, enter `rmsNoise`.

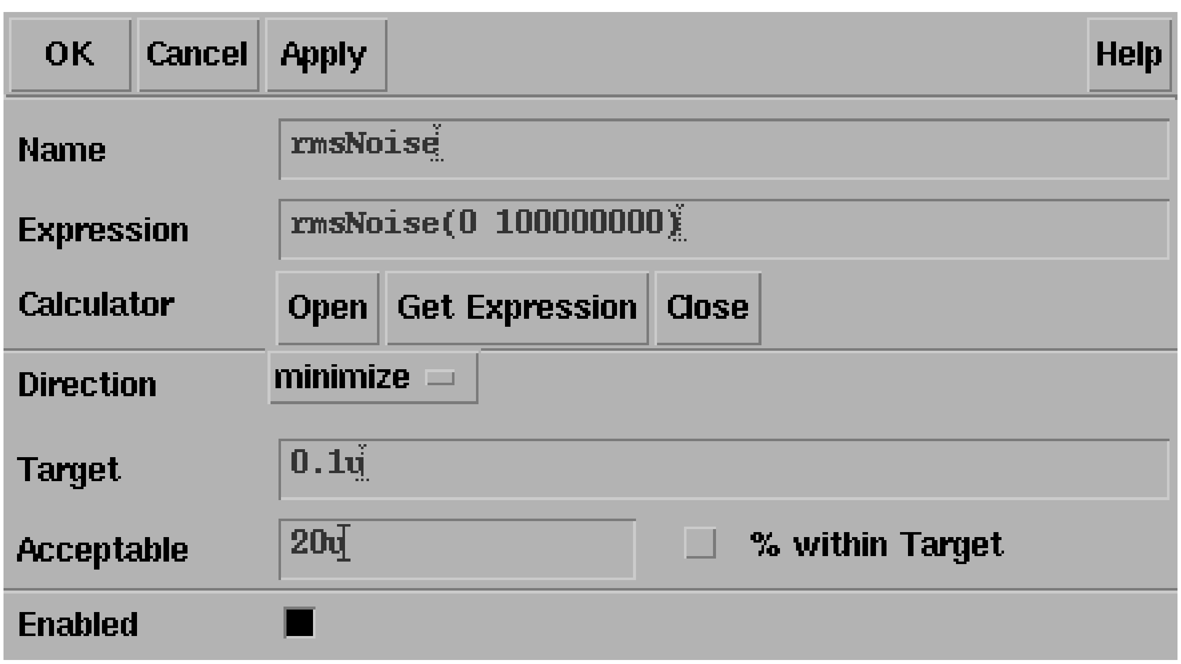
a. Click the *Get Expression* button to the right of the *Calculator* label.

The expression `rmsNoise(0,100000000)` displays in the *Expression* field.

b. In the *Target* field, enter `0.1u`.

c. In the *Acceptable* field, enter `20u`.

The Adding Goals form looks as follows.



d. Verify that the form is enabled and click *OK*.

The first goal is added to the Circuit Optimizer’s Goals window.

Note: By default the Circuit Optimizer minimizes goals.

Goals

#	Name	Direction	Target	Initial	Prev	Current	Enabled
1	rmsNoise	minimize	100n				yes

Add the Second Goal

1. In the Calculator window, click the *vt* button.
2. Then, in the Schematic window, click on the *rms_EVM* output net of the instrumentation block.

   The expression `VT(“/net0101”)` displays in the Calculator window.
3. In the Calculator’s *Special Functions* cyclic field, select *value*.

   When the Value form appears

   a. Enter `130u` in the *Interpolate At* field.

   b. Click *OK*.

   The expanded expression `value(VT(“/net0101”),130u)` displays in the Calculator window.
4. In the Circuit Optimizer, choose *Goals—Add* to add the second goal.

   When the Adding Goals form opens

   a. In the *Name* field, enter `evm`.

   b. Click the *Get Expression* button to the right of the *Calculator* label.

   The expression `value(VT(“/net0101”) 0.00013)` displays in the Expression field.

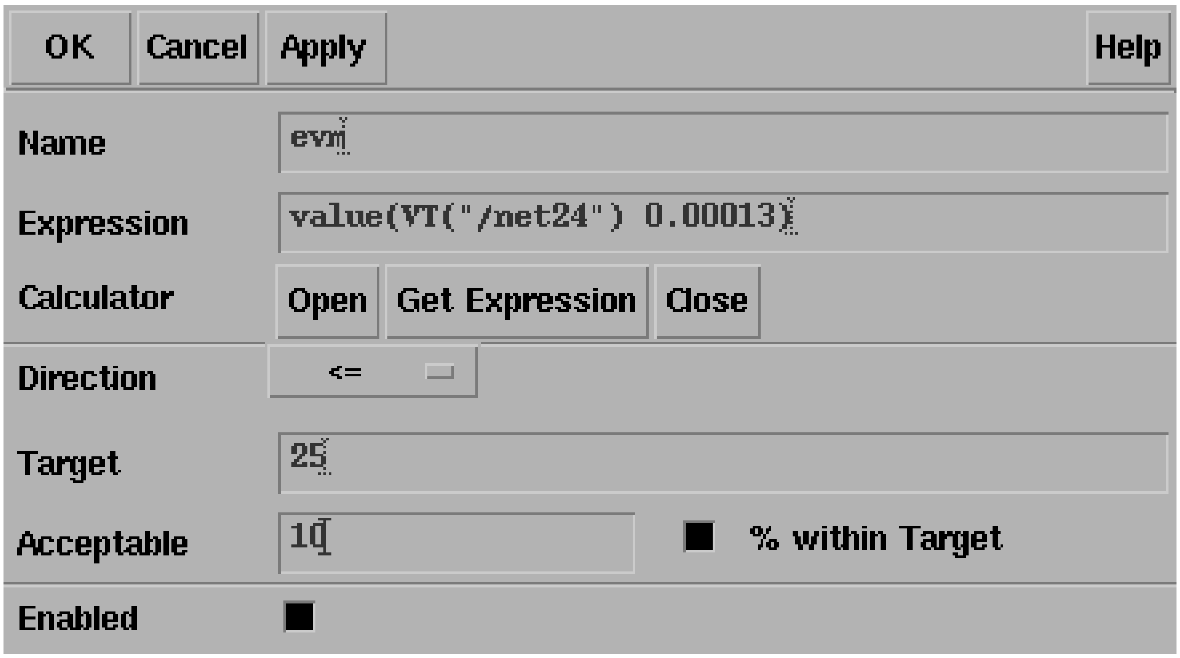
   c. in the *Direction* cyclic field, select <=

   d. In the *Target* field, enter `25`.

e. In the *Acceptable* field, enter `10`.

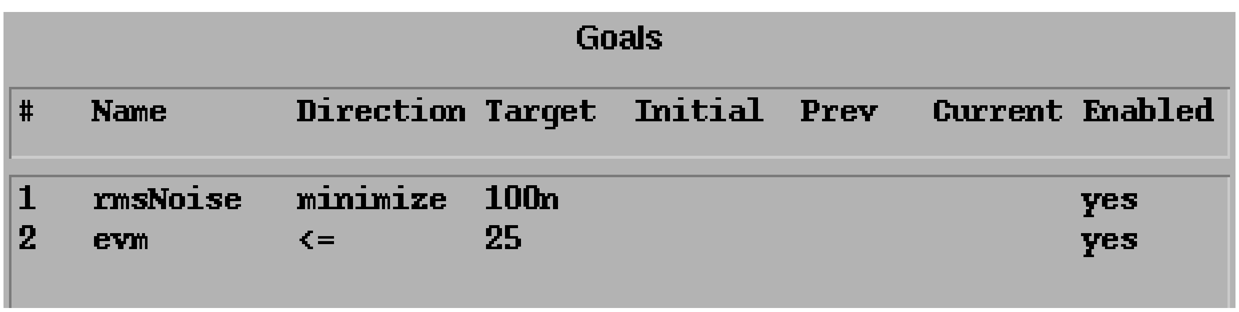
f. Click the *% within Target* button.

The Adding Goals form looks as follows.



g. Verify that *Enabled* is active and click *OK*.

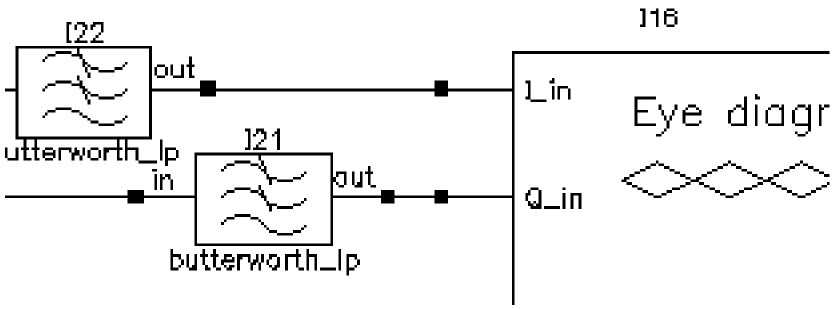
The second goal is added to the Circuit Optimizer's *Goals* window.


Goals

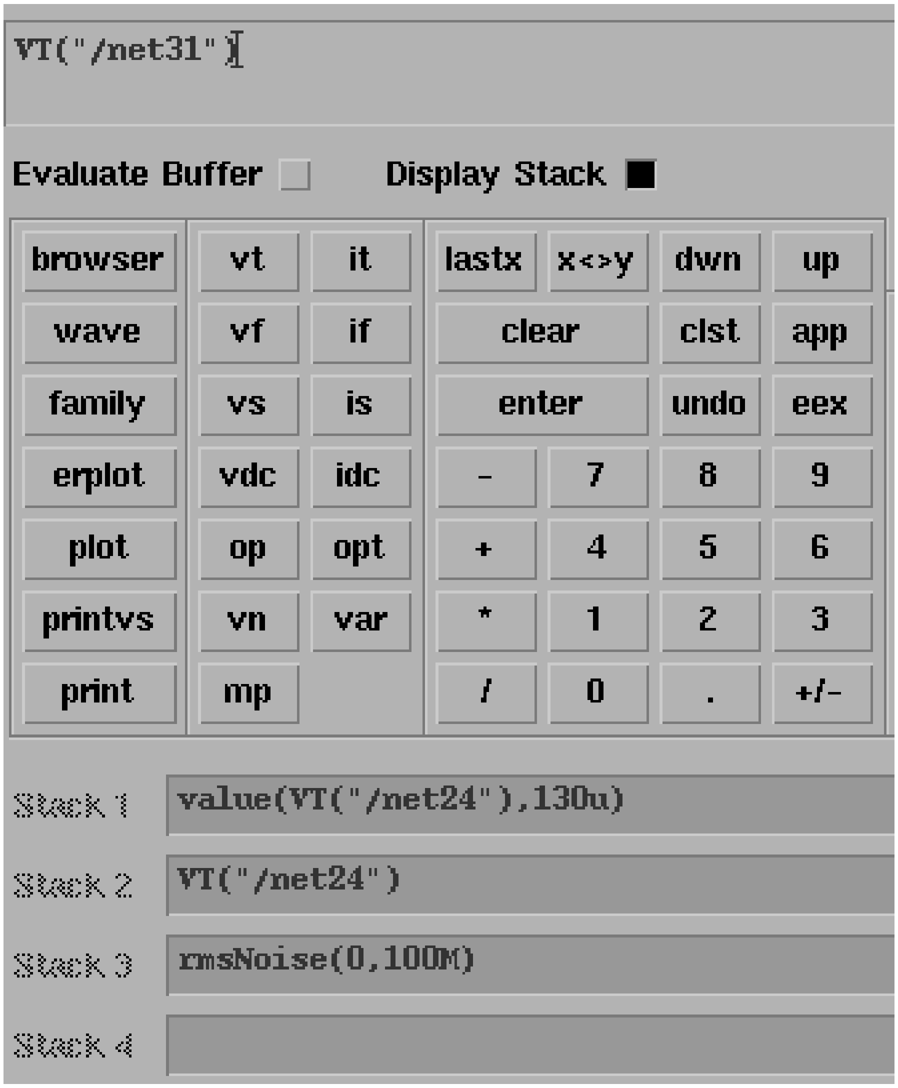
#	Name	Direction	Target	Initial	Prev	Current	Enabled
1	rmsNoise	minimize	100m				yes
2	evm	<=	25				yes

Add the Third Goal

1. In the Calculator window, click the *vt* button. Then, in the Schematic window, click on the *I_in* pin of the instrumentation block.



The expression `VT("/net043")` displays in the Calculator window. Notice that as new expressions are added to the calculator, the existing expressions move down the calculator's stack.



2. In the Calculator's *Special Functions* cyclic field, select *rms*.

   The expanded expression `rms(VT("/net043"))` displays in the Calculator window.

   We will try to keep the *rms* value of this signal level above 300 mV. Note that all goals must be scalars.

3. In the Circuit Optimizer, choose *Goals - Add* to add the third goal.

   When the Adding Goals form opens

a. In the *Name* field, enter `sig_level`.

b. Click the *Get Expression* button to the right of the *Calculator* label.

The expression `rms(VT(“/net043”))` displays in the *Expression* field.

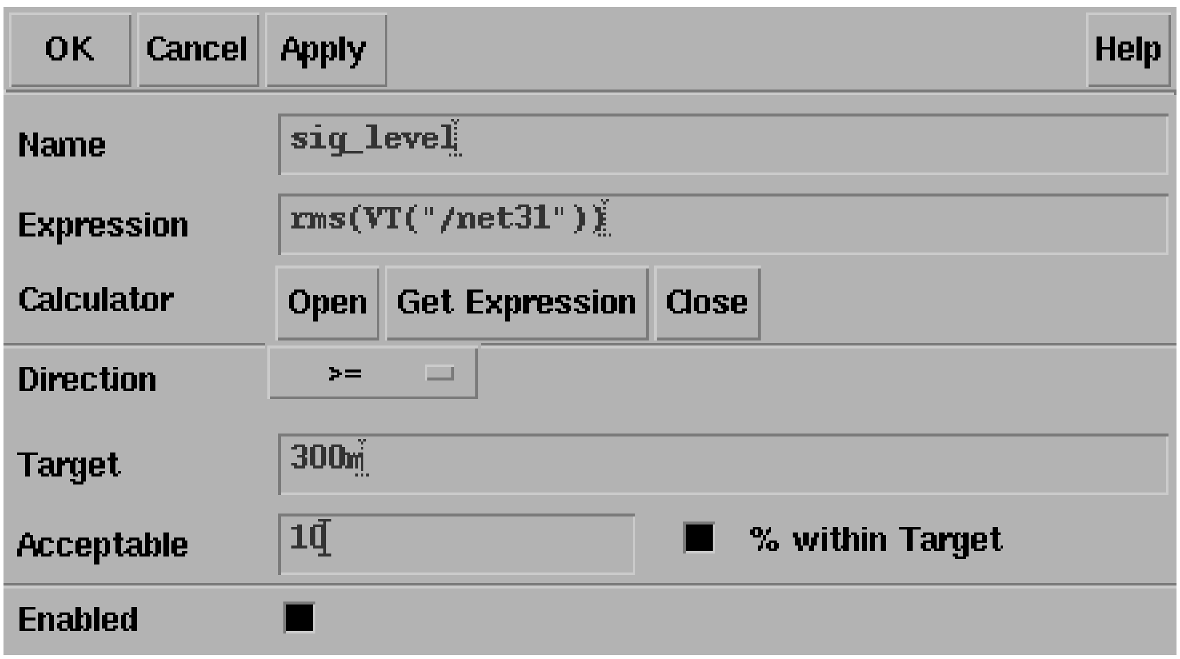
c. in the *Direction* cyclic field, select >=

d. In the *Target* field, enter `300m`.

e. In the *Acceptable* field, enter `10`.

f. Click the *% within Target* button.

The Adding Goals form looks as follows.



g. Click *OK*.

The third goal is added to the Circuit Optimizer's *Goals* window.

Goals

#	Name	Direction	Target	Initial	Prev	Current	Enabled
1	rmsNoise	minimize	100m				yes
2	evm	<=	25				yes
3	sig_level	>=	300m				yes

Add the Circuit Variables to the Optimizer

Add the variables to the Circuit Optimizer window.

1. In the Circuit Optimizer window, choose *Variables—Add/Edit*.

When the Editing Variables form opens

a. In the *Name* list box, click on the *lna_ip3* variable.

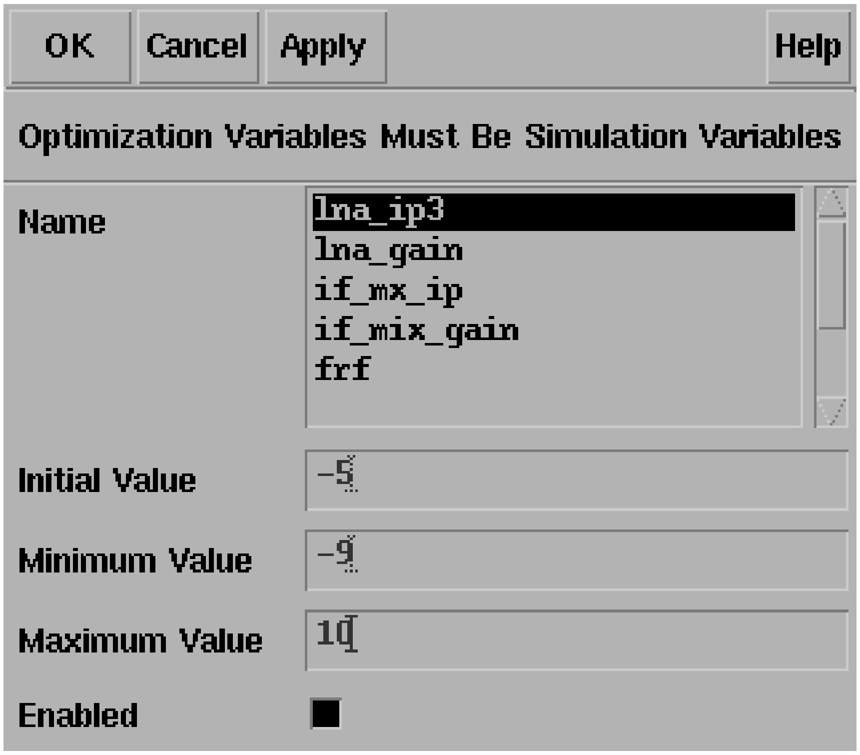
The *lna_ip3* variable is highlighted in the list box and it's current value `-5` displays in the *Initial Value* field.

b. In the *Minimum Value* field, enter `-9`.

c. In the *Maximum Value* field, enter `10`.

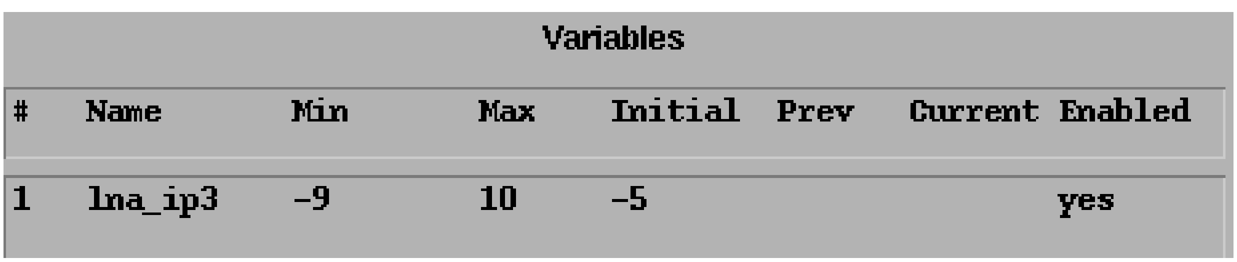
d. If necessary, click *Enabled*.

The Editing Variables form appears as follows.



e. Click *Apply*.

Information for the *lna_ip3* variable displays in the *Variables* section of the Circuit Optimizer.



Variables

#	Name	Min	Max	Initial	Prev	Current	Enabled
1	lna_ip3	-9	10	-5			yes

2. Repeat this procedure to add all the variables and values listed in Table 12-11

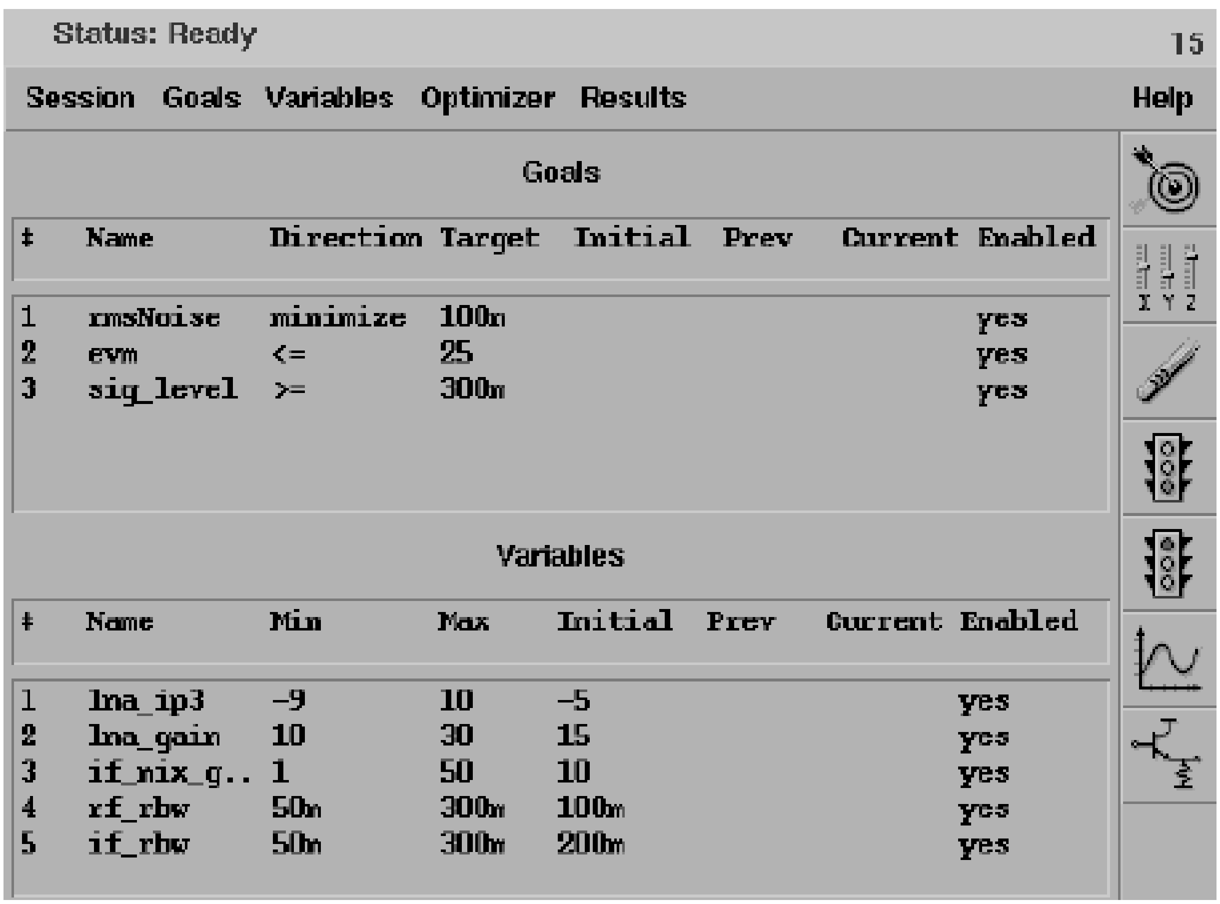
Table 12-11 rVariables and Values for the Optimize

Variable Name	Initial Value	Minimum Value	Maximum Value
lna_ip3	-5	-9	10

Variable Name	Initial Value	Minimum Value	Maximum Value
lna_gain	15	10	30
if_mx_gain	10	1	50
if_rbw	200m	50m	300m
rf_rbw	100m	50m	300m

3. Click *OK* to close the Editing Variables form.
4. In the Circuit Optimizer window, the variables and optimization goals appear as shown in Figure 12-14 on page 1002.

Figure 12-14 Circuit Optimizer Setup



Run the Circuit Optimizer

1. In the Circuit Optimizer, choose *Optimizer—run n* to display the *Run for Fixed Number of Iterations* form.

   In the *Run for Fixed Number of Iterations* form

   a. In the *Number of Iterations* field, move the slider to the right until *12* displays.

   b. Click *OK.*

   The *Run for Fixed Number of Iterations* form closes and the Circuit Optimizer starts running.

   Watch for simulator startup messages in the CIW. Monitor the progress of the analysis in the log window.

   The Waveform window opens when the first simulation completes. As simulations complete, the results are added to the plots in the Waveform window

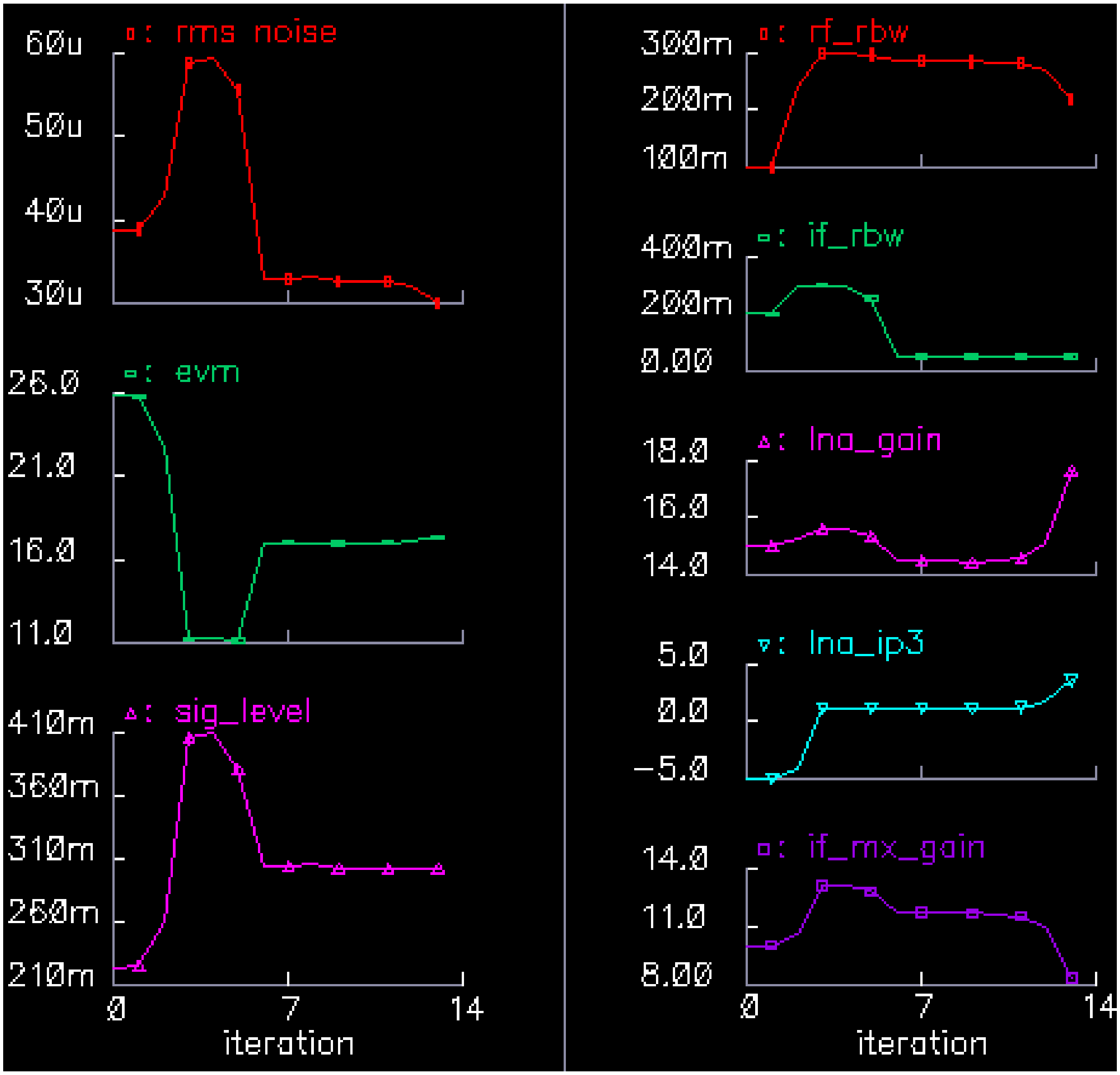
Note: This Circuit Optimizer analysis might take up to several hours to complete.

Viewing the Circuit Optimizer Output

The Circuit Optimizer results displayed in the Waveform window are shown in Figure 12-15 on page 1004.

- The traces on the left show the EVM and RMS output signal level, and the RMS noise at each Circuit Optimizer iteration.
- The traces on the right show the variable function block specifications at each Circuit Optimizer iteration.

Figure 12-15 Circuit Optimizer Results After 12 Iterations

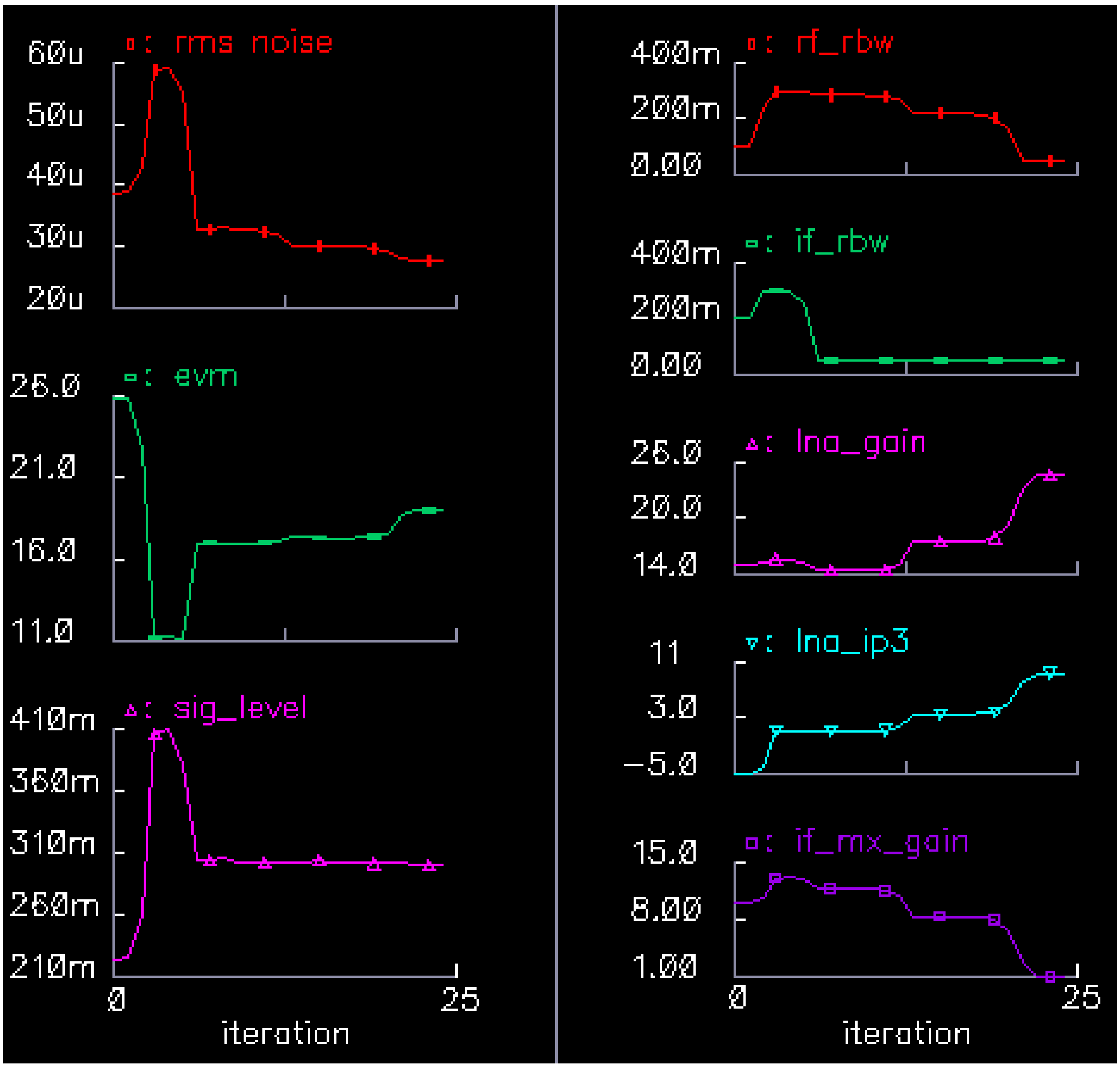


Although the example is contrived, it illustrates the use model. After the Circuit Optimizer met the constraints it tried to minimize RMS noise.

1. Save the initial state of the Analog Design Environment in case you want to start over.
2. Then in the Circuit Optimizer window, click *Results—Update Design*. The last click updates the variables in the Analog Design Environment window with the last set of variables found by the Circuit Optimizer. You will use these states in the passband view.

3. If you have time to run the optimization for another 12 iterations you should see the results in Figure 12-16 on page 1005.

Figure 12-16 Circuit Optimizer results after 24 iterations



Summarizing the Design Procedure

To summarize, the semi-automated design procedure consists of

- Setting up the measurements
- Placing tolerances on the block parameters
- Constraining the system performance
- Identifying a quantity to minimize (or maximize)
- Running the Optimizer
- Evaluating the results

This is why the process is called *semi*-automated. Upon evaluating results for the first or second time you will probably have to

- Refine tolerances
- Refine goals
- Add or delete constraints
- Add or delete variables

Each simulation covers 100s, or about 80 CDMA symbols. The suppressed carriers are an RF carrier at 2.14 GHz, an LO carrier at 2.354GHz, and an IF carrier at 214 MHz. The symbol rate is 1.2288 Mega-symbols per second.

Each combined transient and noise analysis runs in less than 3 minutes on an Ultra 1. Six Circuit Optimizer iterations take between 1 and 2 hours depending on the machine and the options used. Based on run times from the *BB_test_bench* circuit, with Envelope Following analysis the same six Circuit Optimizer iterations would have taken over 6 days. With transient analysis they would have taken over 2 months. There is no way either passband approach can optimize specifications and tolerances in any reasonable amount of time.

Creating a Passband View of the Architectural Model

Once you have designed an architecture, you can quickly create a passband view of the architectural model. (Currently, the passband behavioral models in the *top_dwnPB* category and in the passband view do not introduce any specifications that are not in the baseband models. In subsequent software releases of the library, the passband models may have the ability to introduce new specifications such as IIP2 and DC offsets.)

The passband view checks for problems that might have escaped detection in the baseband view. For example, although the baseband view quickly assesses what filters do to the baseband signal, baseband models do not indicate whether the filters are indeed removing undesired carrier harmonics.

Baseband modeling is also not the best way to evaluate image rejection. Although the baseband model accurately simulates how the desired signal propagates through an image rejection receiver, it does not accurately simulate how much of the image signal propagates to the receiver output.

The passband view also creates a system testbench as mentioned in "Top-Down Design of RF Systems" on page 904.

Procedures for Creating the Passband Model of the Receiver

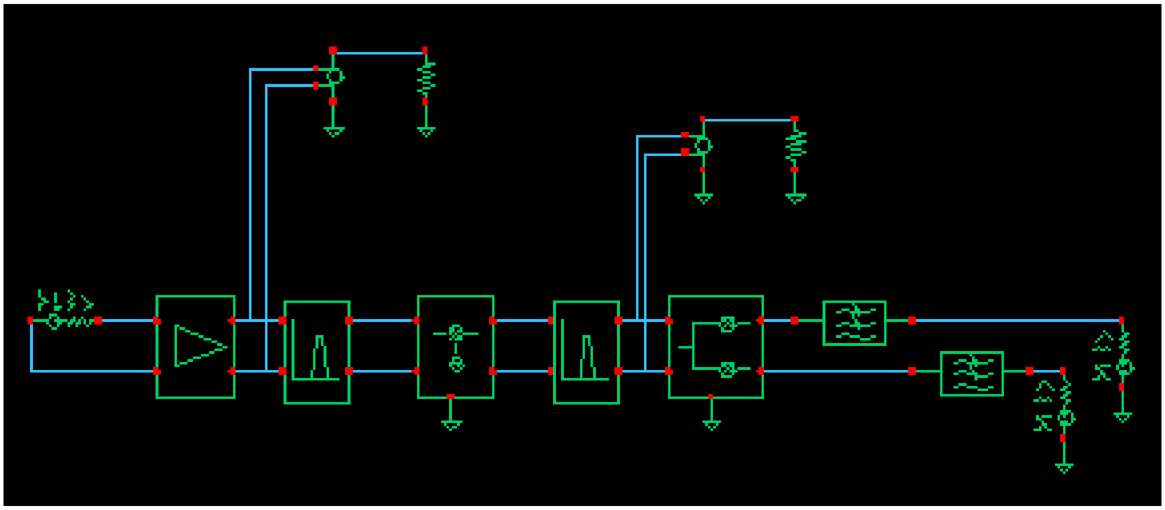
The procedures described in this section illustrate how to

- Switch from a baseband to a passband view
- Make an end-to-end RF measurement
- Measure the one dB compression point

  The one dB compression point is usually a transmitter specification but it is used to demonstrate this flow because it is easier to set up.

1. Copy the original receiver model from the Circuit Optimizer analysis to a new schematic window. Copy everything from the LNA to the low pass filters.
2. Edit the properties of the IQ_demodulator (*IQ_demod_BB*) to set the last parameter, *flo*, to *-frf+flo1*. The baseband view does not need the local oscillator frequency but the passband view does.
3. Load the low pass filters with ports.
4. Connect a port across the LNA_BB inputs. Set the Frequency name to "fin", the frequency to frf, and the amplitude to "power". (Don't abbreviate power to "pwr". "pwr" is a reserved variable and you will not get any warning. Spectre may complain about a mysterious indexed undefined variable that increments from run to run.)
5. Add loaded voltage-controlled-voltage sources as shown in Figure 12-17 on page 1008 to observe intermediate differential voltages. Alternately, you could use the 4.4.6 engineering release Main Results form. It lets you display differential voltages without using the Waveform Calculator or inserting voltage-controlled-voltage sources.

Figure 12-17 Passband View



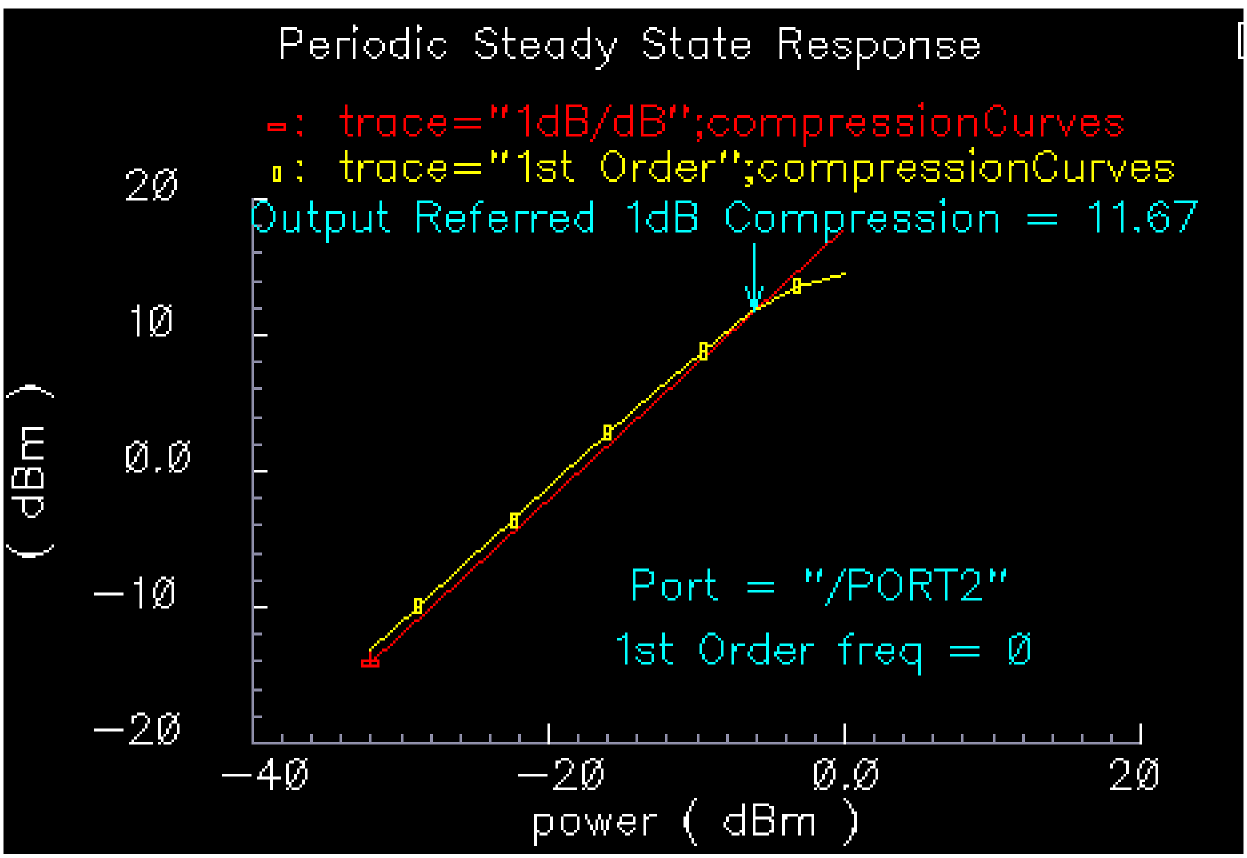
6. Check and save the schematic
7. Close the schematic window.
8. Bring up a Library Manager and select but do not open the schematic you just created.
9. In the Library Manager window click File->New->Cell View.
10. In the window that comes up, select Hierarchy Editor for the Tool. Click OK.
11. Type in “schematic” for the View then click on “Use Template.”
12. Set the Name to Spectre
13. Enter “veriloga_PB” as the first item in the View List then click OK.
14. In the HE (Hierarchy Editor) window, click File->Save. This is important.
15. Click the Open button to bring up the schematic.
16. Bring up an Analog Environment tool and use the states from the last Circuit Optimizer iteration.
17. Recall the states you saved from the last Circuit Optimizer iteration. Add the “power” variable and set it to -16.
18. Delete the previous Analyses and set up a PSS analysis. Enter a new Fundamental Tone named “LO” and make it 2.354GHz. Auto Calculate the Beat Frequency and enter “1” for the Number of harmonics.

19. Run the analysis and observe the intermediate differential voltages. The model is indeed now a passband model. At the higher power levels the LNA output contains odd harmonics of the RF carrier. The filter reflects the odd harmonics back to the LNA and does not let them propagate forward. The baseband model does not simulate the odd harmonics but it does simulate the intermodulation term between the second harmonic and fundamental that falls at the fundamental. One reason to simulate the passband view is to check for peak voltage levels that might exceed voltage ratings. The baseband models only simulate peak voltage at the carrier fundamental, not the absolute peak.

20. Set up swept PSS analysis. Sweep “power” from -32 to 0 in 10 steps.

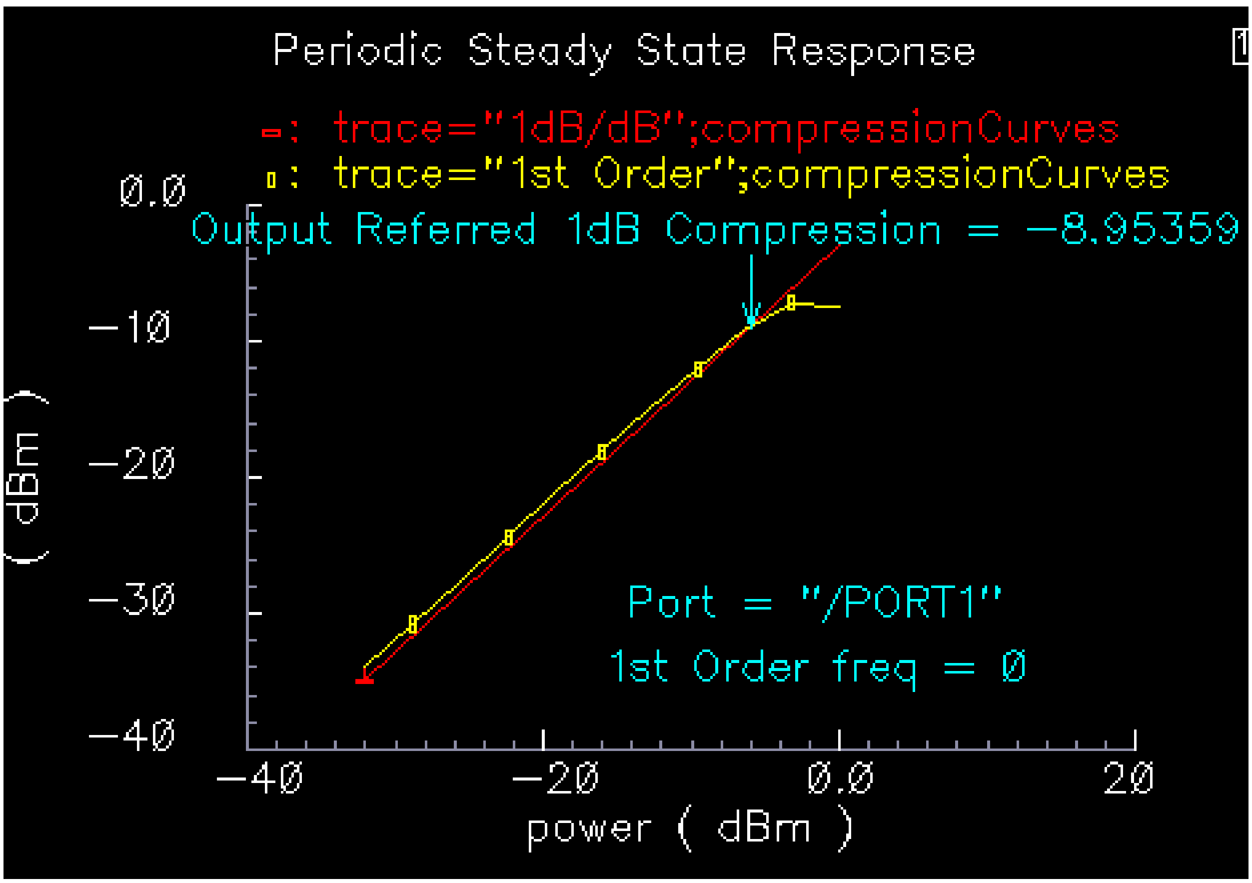
21. After the sweep finishes, click on Results->Direct Plot in the Environment window, select Compression point, 1dB, Output Referred. Select the 0 harmonic because the end-to-end system produces a baseband output. Then click on the port loading the top output low pass filter. You should see the compression point plot show in Figure 12-18 on page 1009.

Figure 12-18 End-to-end RF measurement, one dB compression point at the I-baseband output.



22. Repeat the last step but this time click on the lower output port. You should see the compression plot in Figure 12-19 on page 1010.

Figure 12-19 End-to-end RF measurement, one dB compression point at the Q-baseband output.



Comparing Baseband and Passband Models

This section illustrates how to compare baseband and passband models by:

- Setting up a Transient analysis with the passband view
- Setting up a Transient analysis with the baseband view f
- Directly comparing the baseband and passband models.

You will run one analysis of the baseband view and two analyses of the passband view. You will perform the second passband analysis with tightened tolerances.

1. Save the passband schematic under a different name. You will use the new copy.
2. Repeat steps 9 through 17 from the last recipe for the new copy but do not enter the "veriloga_PB" view in the View List yet. You will do a baseband analysis first.
3. Delete the port driving the LNA.
4. You can also delete the loaded voltage-controlled-voltage sources.
5. You need to synthesize an antenna signal. Add an IQ_mod_BB from the top_dwnBB category. Set the I and Q gains to 0 dB. Set the 1dB compression points to 1000 so that the modulator is ideal. Instantiate it in front of the LNA with the pins aligned then wire the pins straight across. Ground the "phase_err" pin.
6. Drive the "I_in" pin of the IQ modulator with a port. Set the port frequency to 2MHz and name the frequency BB1. Set the amplitude to -16dBm.
7. Do the same for the "Q_in" modulator input.
8. Load or duplicate the states from the 12-iteration Circuit Optimizer analysis but delete the Noise and Transient analyses.
9. Remove AM/PM conversion from the LNA by setting the last parameter in the properties list to zero. It is not fair to compare passband and baseband views with AM/PM conversion because the passband view does not capture it.
10. Set up a 1us Transient analysis with default options.
11. Run the analysis and plot the filtered baseband outputs, the outputs of the low pass filters. Note how fast the simulation runs. Save the results so you can plot them again later. [1]
12. Switch to the passband view by entering "veriloga_PB" in the View List in the Hierarchy Editor. Click the update button in the HE.
13. After you switch to the veriloga_PB view, edit the IQ modulator properties to set "flo" to frf. Edit the demodulator properties and set its "flo" to flo1-frf.
14. Click Results->Printing Plotting Options then click the "Overlay Plots" button.
15. Overlay the passband results with the baseband results. You should see the waveforms in Figure 12-20 on page 1012. The comparison is not very good.

---

1. Note that the baseband outputs are out of phase with each other, even though the baseband inputs are in phase. In the baseband model, changing the RF-IF mixer LO from "flo1" to "-flo1" fixes the sign problem. In the passband model, the IQ_demodulator flo should be frf-flo1. To maintain the convention, in the baseband model the IF filter's carrier frequency should be frf-flo1.

16. Rerun the analysis with conservative options and set reltol to 1e-6. This run will take a while.

17. Plot the results.

18. Recall the saved baseband results and overlay them with those from the last simulation. You should see the waveforms in Figure 12-21 on page 1013. The passband results now lay right on top of the baseband results but took much longer to compute! It was not obvious without the baseband results that the first passband simulation did not run with tight enough numerical tolerances.

Figure 12-20 Passband and baseband results with default options in the passband analysis

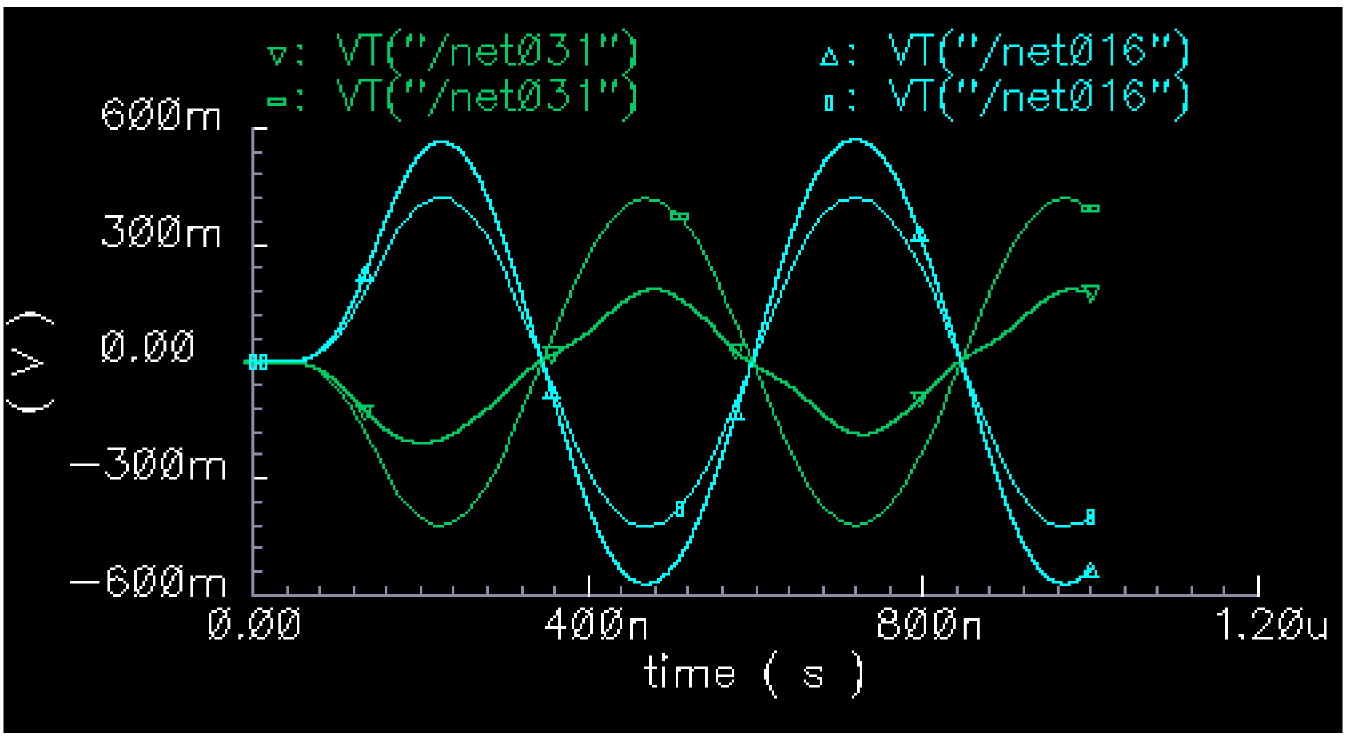
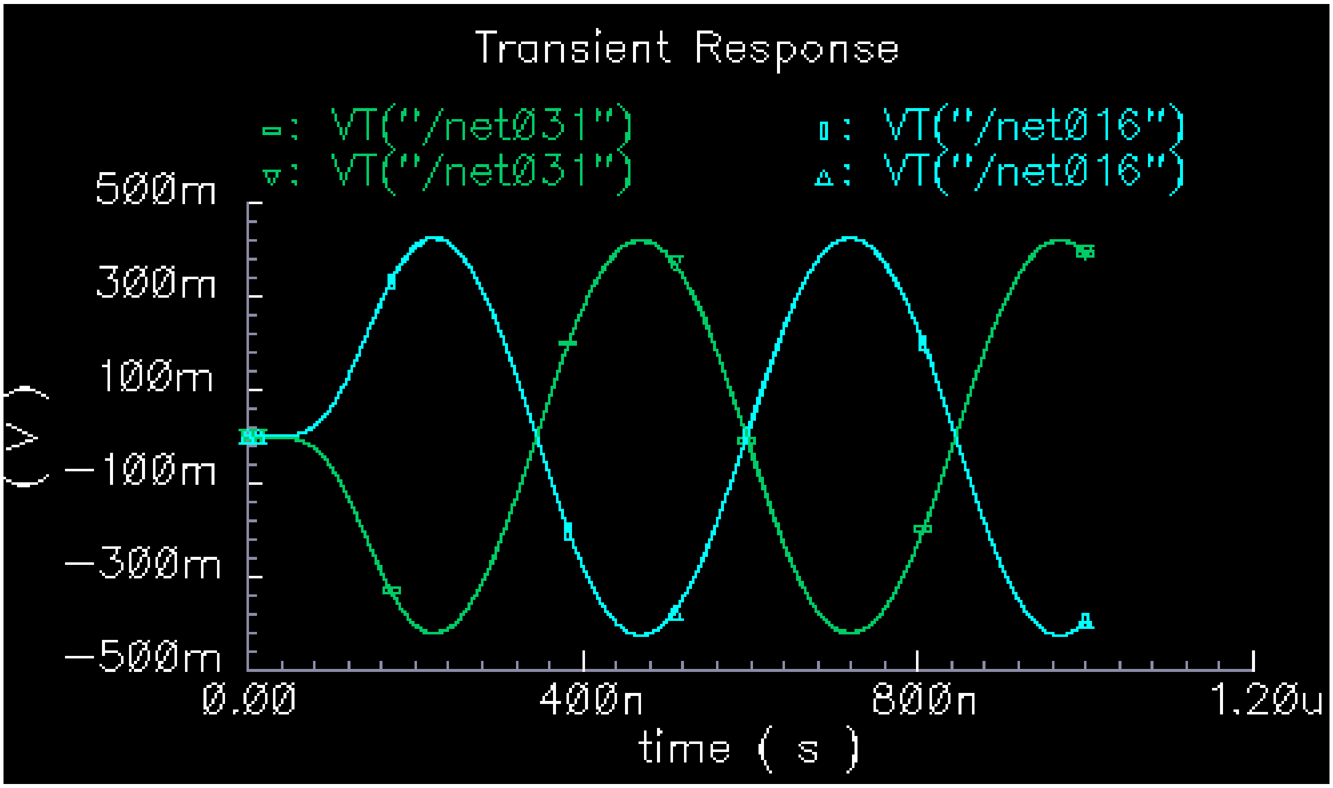


Figure 12-21 Passband and baseband results with tighter options in the passband analysis



Relationship Between Baseband and Passband Noise

Noise analysis at baseband can be confusing because factors of two appear in a number of places throughout the calculations. For example:

- Each passband node becomes two equivalent baseband nodes.
  - Does noise injected at a passband node split between the two equivalent baseband nodes?
  - If so, does the noise split evenly?
- As shown in the BB_test_bench example, baseband models simulate peak in-phase and peak quadrature components of the carrier.

  When analyzing signal-to-noise ratios, does that mean you have to use half the square of the baseband signals?

- The analog design environment displays single sided power spectral densities.

  Since the baseband power spectral density is the two-sided passband density shifted down, is there another factor of two because we can only see the baseband density for positive frequencies?

- White noise at the input of a mixer mixes up to the carrier from DC but there is also noise at twice the carrier that mixers down to the carrier.

  Does the baseband model account for this?

Before sorting out the factors of two, please note that baseband noise analysis is valid only for small signals. If any element in the architecture operates in a non-linear fashion, the noise analysis might be inaccurate. This is due to the fact that a baseband noise analysis follows a DC operating point analysis, rather than a PSS analysis.

Instantaneous incremental gain in a passband static non-linear model dithers at the carrier frequency.

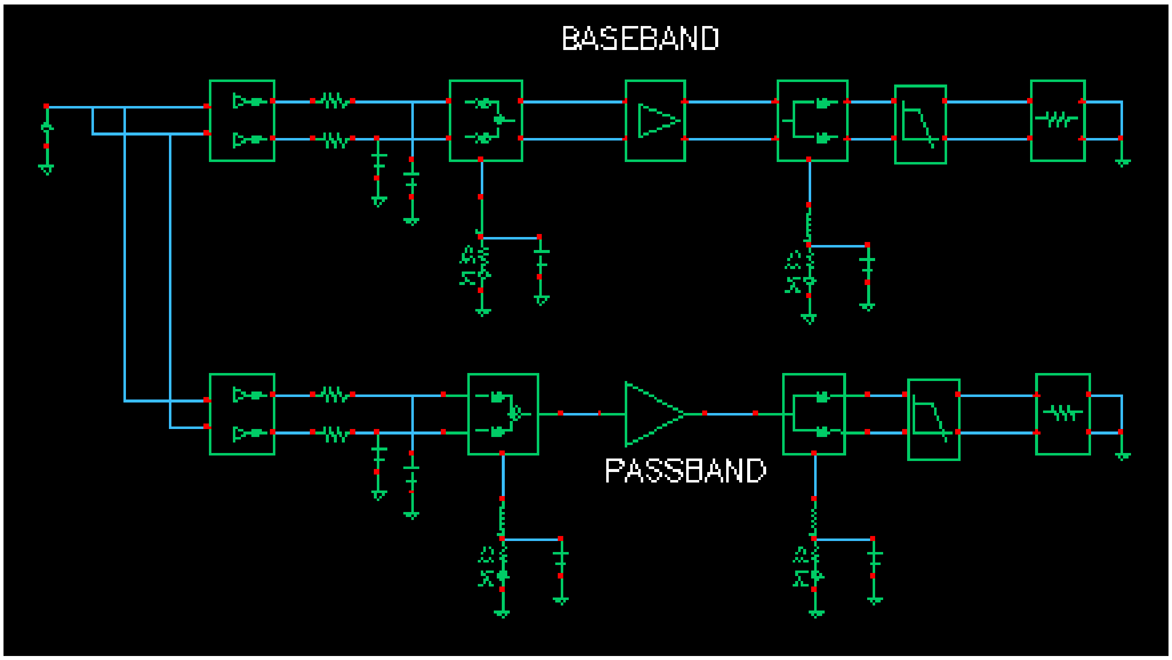
- When the carrier swings through zero, the incremental gain is large and noise at the input is amplified.
- When the carrier reaches its peak and drives the circuit into saturation, the incremental gain is smaller.

The average gain is greater than the minimum gain. The baseband model remains in the non-linear region because it only simulates peak voltages. Consequently, the incremental gain is always a minimum and the baseband model under-estimates the amount of noise propagating to the output. If the peak input signal drives the model into saturation, be sure to scale the baseband noise results accordingly.

Intro to Analysis

The circuit discussed here is called *noise_test_circuit* and you can find it in the *rfExamples* library. The circuit looks as shown in Figure 12-22 on page 1015.

Figure 12-22 The noise_test_circuit in the rfExamples Library



The *noise_test_circuit* shows the relationship between baseband and passband noise. One branch consists of passband models. The other branch is a baseband equivalent of the first. You can assess noise at each of three observation points located in each branch of the circuit. At each observation point, you can examine both the noise and pnoise summaries.

The I and Q inputs are both driven by the same DC source so that you only have to view one baseband output, the other baseband output is identical by symmetry. Noise parameters in the passband and baseband models are identical. Aside from the behavioral blocks at the end of each branch, each behavioral block has noise injected at its input.

Prep Steps for Analyses

Setup a PSS analysis. Since the local oscillator is inside the passband mixer models you will have to manually enter the frequency (1GHz) into the PSS analysis form. Let the beat frequency be *autocalculated* and use 1 harmonic.

Setup a pnoise analysis with the start frequency equal to 0 Hz, the stop frequency equal to 100MHz, use a linear sweep with 100 steps. Set the Maximum sideband to 1. For the input source select *none* and for the output use *voltage*. For the positive output node, select the I-input of the IQ_modulator in the passband branch. Use ground for the negative node.

Setup a noise analysis that sweeps frequency from 0 to 100MHz linearly in 100 steps. Use voltage for the output noise and select the I-input of the *IQ_mod_BB* component in the baseband branch for the positive node. Select ground for the negative node. Set the input noise port to any one of the ports in the circuit. Noise from that port will not affect either passband nor baseband branches.

Run the analysis.

When the analysis finishes, go to the analog design environment simulation window and click on **results-print-pss noise summary.**

In the form that appears, include All types, set *Type* to *integrated noise*, look at noise from 0 to 100MHz, select “truncate by number", and view the top 2 noise contributors. There are only 2 noise contributors at this point, noise at the I/Q inputs and noise from the low pass resistors.

Now print the noise summary. (This is different from the pnoise summary.) Again, print integrated noise (from 0 to 100MHz). Select All types and print noise from the top 2 noise contributors.

Figure 12-23 on page 1017 shows the noise summaries. The two summaries agree because at this point in the circuit, both nodes are really baseband nodes.

Figure 12-23 Noise Summaries for the First PSS PNoise and Noise Analyses

```
Device    Param              Noise Contribution    % Of Total

/I2       IQ_modulator_i     1.56676e-12           70.06
/R20      rn                 6.69678e-13           29.94

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.23644e-12
No input referred noise available


Device    Param           Noise Contribution    % Of Total
/I22      IQ_mod_BB_i     1.56676e-12           70.06
/R22      rn              6.69679e-13           29.94

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.23644e-12
Total Input Referred Noise = Inf
```

Repeat the analyses but this time select the “outi” pins on the low pass filters as outputs in the appropriate noise analyses. Again print the same noise summaries but this time look at the top 12 noise contributors in each summary.

Figure 12-24 on page 1018 shows the noise summaries for the second analysis. The top 7 noise contributors in the baseband and passband branches agree. The remaining noise contributors are negligible, and should be negligible for the circuit since it has no AM/PM conversion.

Figure 12-24 Noise Summaries for the Second PSS PNoise Analyses

Device	Param	Noise Contribution	% Of Total
/I77	PA_BB	7.49879e-10	82.29
/I2	IQ_modulator_i	9.69448e-11	10.64
/R20	rn	4.14371e-11	4.55
/I87	butterworth_lp_i_noise	1.52928e-11	1.68
/I84	IQ_demodulator	7.64648e-12	0.84
/PORT13	rn	3.77571e-14	0.00
/PORT14	rn	3.77571e-14	0.00
/I2	IQ_modulator_q	2.33139e-15	0.00
/R21	rn	9.96506e-16	0.00
/R22	rn	0	0.00
/R23	fn	0	0.00
/R23	rn	0	0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 9.11279e-10
No input referred noise available

Device	Param	Noise Contribution	% Of Total
/I76	PA_BB_i	7.49861e-10	82.29
/I22	IQ_mod_BB_i	9.69448e-11	10.64
/R22	rn	4.14371e-11	4.55
/I86	butterworth_lp_i_noise	1.52928e-11	1.68
/I82	IQ_demod_BB_i	7.64642e-12	0.84
/PORT15	rn	3.77571e-14	0.00
/PORT9	rn	3.77571e-14	0.00
/I76	PA_BB_q	1.80329e-14	0.00
/I22	IQ_mod_BB_q	2.33136e-15	0.00
/R23	rn	9.96493e-16	0.00
/I82	IQ_demod_BB_q	1.54184e-31	0.00
/R23	fn	0	0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 9.11279e-10
Total Input Referred Noise = 0.0151541

Now, repeat the analysis but this time change the Pnoise sweep to run from 900MHz to 1.1GHz in 200 linear steps and select the power amplifier output as the noise output node. For the noise analysis, leave the sweep at zero to 100MHz, use 100 steps as before, and change the output node to be the I-output of the power amplifier.

Look at the top 7 noise contributors in each analysis. This time, integrate noise from 900MHz to 1.1GHz for the pnoise run and integrate noise from 0 to 100MHz for the noise run. Figure 12-25 on page 1019 shows the new summaries. Although noise analyses agree at either ends of the branches, noise analyses seem to disagree at a point where the baseband node is only a baseband equivalent, not a true baseband node.

Figure 12-25 Noise Summaries for the Third PSS PNoise Analyses

Device	Param	Noise Contribution	% Of Total
/I77	PA_PB	2.3769e-09	83.70
/I2	IQ_modulator_i	1.53663e-10	5.41
/I2	IQ_modulator_q	1.53663e-10	5.41
/R20	rn	6.568e-11	2.31
/R21	rn	6.568e-11	2.31
/I84	IQ_demodulator	2.42376e-11	0.85
/PORT13	rn	1.19682e-13	0.00

```
Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.83995e-09
No input referred noise available
```

Device	Param	Noise Contribution	% Of Total
/I76	PA_BB_i	2.3769e-09	83.69
/I22	IQ_mod_BB_i	3.07294e-10	10.82
/R22	rn	1.31347e-10	4.62
/I82	IQ_demod_BB_i	2.42376e-11	0.85
/PORT9	rn	1.19682e-13	0.00
/I76	PA_BB_q	5.71606e-14	0.00
/I22	IQ_mod_BB_q	7.38993e-15	0.00

```
Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 2.83997e-09
Total Input Referred Noise = 0.0148587
```

The *apparent* disagreement shown in Figure 12-25 on page 1019 requires an explanation. Let us examine the noise contributors and try to answer some the questions we posed earlier.

Noise at the power amplifier output due to noise injected at a passband node:

- Passband model contributors: *PA_PB* and *IQ_demodulator*, port 9.
- Baseband model contributors: *PA_BB_i*, *IQ_demod_BB_i*, port 13.

Passband and baseband counterparts contribute the same amount of noise. However, in the baseband model, from symmetry you see the same numbers if you look at the Q-node. This means the baseband model predicts twice as much total noise due to noise injected between the modulators.

This factor of two is intentionally introduced to maintain the correct signal-to-noise ratio. The baseband model simulates peak signals; the carrier is suppressed. Without the carrier, signal power equals the square of the peak rather than one half of the square. This factor of two is not as arbitrary as it seems. The baseband model predicts the correct noise after demodulation because the passband demodulator model includes an extra factor of two to offset the factor of two inherent in the demodulation process.

Let the modulated carrier be `i(t)*cos(wc*t) -q(t)*sin(wc*t) + noise(t)`.

Now consider the I-output. To generate the I-output, the demodulator multiplies the signal by `cos(wc*t)`. The only part that propagates through the subsequent filter is `(1/2)i(t) + noise(t)*cos(wc*t)`. To recover `i(t)`, the passband demodulator model must scale this sum by two. (The baseband demodulator does not need to scale by two to extract the baseband signal because the carrier is suppressed.)

Thus, noise at passband demodulator model output equals `2*noise(t)*cos(wc*t)`. The filtered noise power density is then `4*(`*`input noise density`*`)/2`. The factor of 1/2 comes from the cosine. The filtered output noise density is twice the input noise density.

In the baseband model, doubling the noise injected at the passband nodes was not simply a matter of convenience.

So, **to answer questions 1 and 2**

- Yes, noise injected at a passband node splits evenly between the two equivalent baseband nodes

  but

- Since each split is doubled, the ratio of *peak* signal to total noise equals the true signal-to-noise ratio.

Question 3 is rendered moot by using the noise summary and integrating over the proper band. If the pnoise analysis only integrated from 1GHz to 1.1GHz, (instead of from 900MHz to 1.1GHz), there would be a mysterious factor of two error.

In the baseband model, phase noise entering on the phase error pin propagates to both the I and Q outputs. In the baseband model, the same noise power appears on just the one output. Again, the total noise in the baseband model is twice that of the passband model to maintain the correct signal-to-noise ratio.

Noise injected at the modulator input (resistors and modulator noise):

Total noise in the baseband model due to modulator and input resistor noise is twice what it is from just one phase. Thus, the total noise due to sources on the input sides of the modulators differ by a factor of two. This occurs because the passband model is a real multiplier, which modulates the noise. If the peak signal voltage agrees with the baseband model, the passband modulator model attenuates input noise most of the time. The important thing is that the signal-to-noise ratios in the passband and baseband models agree anywhere in the system.

Now, copy the circuit so you can **remove the capacitors at the modulator inputs** and repeat the last set of noise analyses. You will see that in the passband model, the input resistors, R20 and R21, together contribute twice as much as the baseband counterpart, R22. With the capacitors, R20 and R21 together contribute just as much as R22. Without the capacitors, the input noise is truly white over the frequencies of interest. The same thing happens to the modulator noise itself. Figure 12-26 on page 1022 shows the results.

In particular, the modulator now also has noise at twice the carrier frequency and that noise mixes down to the carrier frequency. The baseband model is just that, a baseband model. The answer to question 4 is *no*. The baseband models do not account for noise, or signals, at carrier harmonics. The baseband equivalent noise analysis is valid only if noise injected into the modulators has no power beyond the local oscillator frequency. Phase noise injected at the phase noise pins should also be band limited.

Figure 12-26 Noise Summaries with the input capacitors removed.

Device	Param	Noise Contribution	% Of Total
/I2	IQ_modulator_q	4.22571e-09	29.21
/I2	IQ_modulator_i	4.22571e-09	29.21
/I77	PA_PB	2.3769e-09	16.43
/R21	rn	1.80619e-09	12.49
/R20	rn	1.80619e-09	12.49
/I84	IQ_demodulator	2.42376e-11	0.17

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 1.44651e-08
No input referred noise available

Device	Param	Noise Contribution	% Of Total
/I22	IQ_mod_BB_i	4.22561e-09	50.11
/I76	PA_BB_i	2.3769e-09	28.19
/R22	rn	1.80615e-09	21.42
/I82	IQ_demod_BB_i	2.42376e-11	0.29
/PORT9	rn	1.19682e-13	0.00
/I22	IQ_mod_BB_q	1.01619e-13	0.00

Integrated Noise Summary (in V^2) Sorted By Noise Contributors
Total Output Noise = 8.43322e-09
Total Input Referred Noise = 9.76997e-11

A

Oscillator Noise Analysis

Sergei and Yu,

I turned on red change bars so I can see what you add, or mark for deletion.

Please use the <xxcomment> paragraphtag to insert a comment or note to me. It's at the end of the paragrah menu.

Thanks for looking at this. I REALLY appreciate it.

Belinda

In RF systems, local oscillator phase noise can limit the final system performance. The SpectreRF circuit simulator lets you rigorously characterize the noise performance of oscillator elements. This appendix explains phase noise, tells how it occurs, and shows how to calculate phase noise using the SpectreRF circuit simulator.

"Phase Noise Primer" on page 1024 discusses how phase noise occurs and provides a simple illustrative example.

"Models for Phase Noise" on page 1027 contains mathematical details about how the SpectreRF circuit simulator calculates noise and how these calculations are related to other possible phase noise models. You can skip this section without any loss of continuity, but this section can help you better understand how the Spectre circuit simulator calculates phase noise and better appreciate the drawbacks and pitfalls of other simple phase noise models. This section can also help in debugging difficult circuit simulations.

"Calculating Phase Noise" on page 1038 provides some suggestions for successful and efficient analysis of oscillators and discusses the limitations of the simulator.

"Troubleshooting Phase Noise Calculations" on page 1041 explains troubleshooting methods for difficult simulations.

"Frequently Asked Questions" on page 1045 answers some commonly asked questions about phase noise and the SpectreRF circuit simulator.

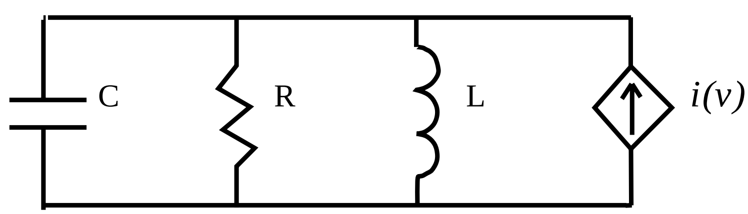
“Further Reading” on page 1051and “References” on page 1051 list additional sources of information on oscillator noise analysis.

The procedures included in this appendix are intended for SpectreRF users who analyze oscillator noise. You must have a working familiarity with SpectreRF simulation and its operating principles. In particular, you must understand the SpectreRF PSS and Pnoise analyses. For information about performing these analyses, consult Chapter 5, “Simulating Oscillators.” You might also read *SpectreRF Theory*.

Phase Noise Primer

Consider the simple resonant circuit with a feedback amplifier shown in Figure A-1, a parallel LC circuit with nonlinear transconductance. At small capacitor voltages, the transconductance is negative, and the amplifier is an active device that creates positive feedback to increase the voltage on the capacitor. At larger voltages, where the transconductance term goes into compression, the amplifier effectively acts as a positive resistor (with negative feedback) and limits the capacitor voltage.

Figure A-1 A Simple Resonant Oscillator



A simple model for the nonlinear transconductance is a cubic polynomial. We hypothesize a nonlinear resistor with a current-voltage relation given by

$$i(v) = -\left(\frac{v}{R}\right)(1-\alpha v^2)$$

The effect of the resistor in parallel with the inductor and the capacitor can be lumped into this transconductance term. The parameter is a measure of the strength of the nonlinearity in the transconductance relative to the linear part of the total transconductance. Because the signal amplitude grows until the nonlinearity becomes significant, the value of this parameter does not affect the qualitative operation of the circuit.

Note: For simplicity, for the remainder of this appendix

$$\alpha = 1/3$$

After some renormalization of variables, where time is scaled by

$$1/\omega_0$$

with

$$\omega_0 = \frac{1}{\sqrt{LC}}$$

and current is scaled by

$$\sqrt{C/L}$$

You can write the differential equations describing the oscillator in the following form

$$\frac{dv}{dt} = -i + \frac{1}{Q}(1 - \alpha v^2)v + \xi(t)$$

and

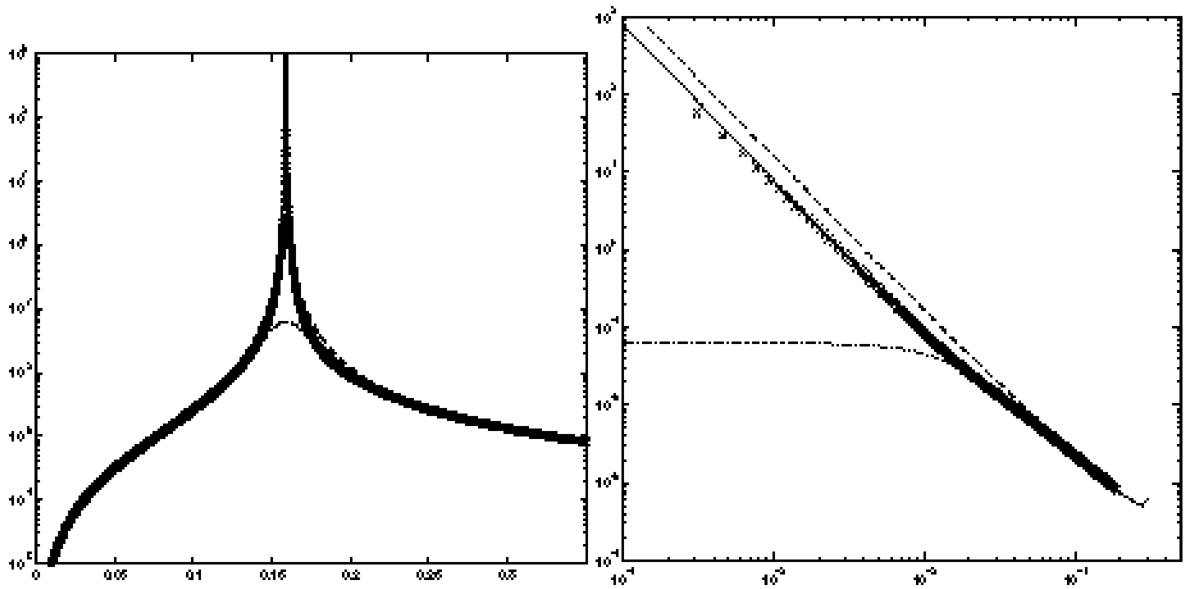
$$\frac{di}{dt} = v$$

In these equations, v and i represent the normalized capacitor voltage and inductor current, respectively, and $\xi(t)$ is a small-signal excitation such as white Gaussian noise, $Q = R/\omega_0 L$ is the quality factor of an RLC circuit made by replacing the nonlinear transconductance by a positive resistance R.

The equations just discussed describe the familiar Van der Pol oscillator system. This model includes many of the qualitative aspects of oscillator dynamics, yet it is simple enough to analyze in detail. Many more complicated oscillators that operate in a weakly nonlinear mode can be approximated with this model by using the first few terms in the Taylor series expansion of the relevant transconductances.

As a brute-force method of calculating the noise properties of this circuit, the nonlinear stochastic differential equations that describe the current and voltage processes were numerically integrated [1], and the noise power was obtained using a standard FFT-periodiagram technique. This technique requires several hundred simulations of the oscillator over many thousands of periods. Consequently, it is not a feasible approach for practical circuits, but it is rigorously correct in its statistical description even though it requires no knowledge of the properties of oscillators, noise, periodicity, or signal amplitudes. Figure A-2 shows the total time-averaged noise in the voltage variable.

Figure A-2 Noise in a Simple Van der Pol System



By plotting *Power Spectral Density* against *Normalized Noise Frequency Offset* for a Q = 5 system, Figure A-2 shows noise in a simple Van der Pol system.

- The left half of Figure A-2 shows noise as a function of absolute frequency.
- The right half of Figure A-2 shows noise as a function of frequency offset from the oscillator fundamental frequency.

  The dashed line is LC-filtered white noise, the dash-dot line is RLC-filtered white noise, the solid line is SpectreRF phase noise, and (x) marks are noise power from a full nonlinear stochastic differential equation solution.

The resulting noise power spectral density looks much like the voltage versus current response of a parallel LC circuit. The oscillator in steady-state, however, does not look like an

LC circuit. As you will see in the following paragraphs, this noise characteristic similarity occurs because both systems have an infinite number of steady-state solutions.

The characteristic shape of the small-signal response of an LC circuit results because an excitation at the precise resonant frequency can introduce a drift in the amplitude or phase of the oscillation. The magnitude of this drift grows with time and is potentially unbounded. In the frequency domain, this drift appears as a pole on the imaginary axis at the resonant frequency. The response is unbounded because no restoring force acts to return the amplitude or phase of the oscillation to any previous value, and perturbations can therefore accumulate indefinitely.

Similarly, phase noise exists in a nonlinear oscillator because an autonomous oscillator has no time reference. A solution to the oscillator equations that is shifted in time is still a solution. Noise can induce a time shift in the solution, and this time shift looks like a phase change in the signal (hence the term *phase noise*). Because there is no *resistance* to change in phase, applying a constant white noise source to the signal causes the phase to become increasingly uncertain relative to the original phase. In the frequency domain, this corresponds to the increase of the noise power around the fundamental frequency.

If the noise perturbs the signal in a direction that does not correspond to a time shift, the nonlinear transconductance works to put the oscillator back on the original trajectory. This is similar to AM noise. The signal uncertainty created by the amplitude noise remains bounded and small because of the action of the nonlinear amplifier that created the oscillation. The LC circuit operates differently. It lacks both a time (or phase) reference and an amplitude reference and therefore can exhibit large AM noise.

Another explanation of the similarity between the oscillator and the LC circuit is that both are linear systems that have poles on the imaginary axis at the fundamental frequency, ω_0. That is, at the complex frequencies $s = i\omega_0$. However, the associated transfer functions are not the same. In fact, because of the time-varying nature of the oscillator circuit, multiple transfer functions must be considered in the linear time-varying analysis.

Understanding the qualitative behavior of linear and nonlinear oscillators is the first step towards a complete understanding of oscillator noise behavior. Further understanding requires more quantitative comparisons that are presented in Models for Phase Noise. If you are not interested in these mathematical details, you might skip ahead to "Calculating Phase Noise" on page 1038.

Models for Phase Noise

This section considers several possible models for noise in oscillators. In the engineering literature, the most widespread model for phase noise is the Leeson model [2]. This heuristic model is based on qualitative arguments about the nature of noise processes in oscillators.

It shares some properties with the LC circuit models presented in the previous section. These models fit well with an intuitive understanding of oscillators as resonant RLC circuits with a feedback amplifier. In the simplest treatment, the amplifier is considered to be a negative conductance whose value is chosen to cancel any positive real impedance in the resonant tank circuit. The resulting linear time-invariant noise model is easy to analyze.

Linear Time-Invariant (LTI) Models

To calculate the noise in a parallel RLC configuration, the noise of the resistor is modeled as a parallel current source of power density

$$S(\omega) = \frac{4k_BT}{R}$$

In general, if current noise excites a linear time-invariant system, then the noise power density produced in a voltage variable is given by [3] as follows

$$S_v(\omega) = |H(\omega)|^2 S_i(\omega)$$

where $H(\omega)$ is the transfer function of the LTI transformation from the noise current source *input* to the voltage *output*. The transfer function is defined in the standard way to be

$$H(\omega) = \frac{\nu_0(\omega)}{i_s(\omega)}$$

where i_s is a (deterministic) current source and v_0 is the measured voltage between the nodes of interest.

It follows that the noise power spectral density of the capacitor voltage in the RLC circuit is, at noise frequency $\omega = \omega_0 + \omega'$ with $\omega' << \omega_0$.

$$S_v(\omega') = \frac{4k_BTR}{1 + 4(\omega'/\omega_0)^2 Q^2}$$

where the quality factor of the circuit is

$$Q = \frac{R}{\omega_0 L}$$

The parallel resistance is R (the source of the thermal noise), and ω_0 is the resonant frequency.

If a noiseless negative conductance is added to precisely cancel the resistor loss, the noise power for small ω' / ω_0 becomes

$$S_v(\omega') = \frac{k_B T R}{(\omega'/\omega_0)^2 Q^2}$$

This linear time-invariant viewpoint explains some qualitative aspects of phase noise, especially the $(\omega_0 / Q\omega')^2$ dependencies. However, even for this simple system, a set of complicating arguments is needed to extract approximately correct noise from the LTI model. In particular, we must explain the 3 dB of excess amplitude noise inside the resonant bandwidth generated by an LC model but not by an oscillator (see “Amplitude Noise and Phase Noise in the Linear Model” on page 1033). Furthermore, many oscillators, such as relaxation and ring oscillators, do not naturally fit this linear time-invariant model. Most oscillators are better described as time-varying (LTV) circuits because many phenomena, such as upconversion of $1/f$ noise, can only be explained by time-varying models.

Linear Time-Varying (LTV) Models

For linear time-invariant systems, the noise at a frequency ω is directly due to noise sources at that frequency. The relative amplitudes of the noise at the system outputs and the source noise are given by the transfer functions from noise sources to the observation point. Time-varying systems exhibit frequency conversion, however, and each harmonic $k\omega_0$ in the oscillation can transfer noise from a frequency $\omega \pm k\omega_0$ to the observation frequency ω. In general, for a stationary noise source $\xi(t)$, the total observed noise voltage will be [3]

$$S_v(\omega) = \sum_k |H(\omega)|^2 S_\xi(\omega + k\omega_0)$$

Each term in the series represents conversion of current power density at frequency $\omega + k\omega_0$ to voltage power density at frequency ω with gain $|Hk(\omega)|^2$. As an example, return again to the Van der Pol oscillator with $\alpha = 1/3$ and notice how a simple time-varying linear analysis of noise proceeds.

The first analysis step for the Van der Pol oscillator is to obtain a large-signal solution, so you set $\xi(t) = 0$. In the large-Q limit, the oscillation is nearly sinusoidal and so it is a good approximation to assume the following

$$v(t) = a \sin \omega_0 t$$

The amplitude, a, and oscillation frequency can be determined from the differential equations that describe the oscillator. Recognizing that

$$i(t) = \left(\frac{a}{\omega_0}\right)\cos(\omega_0 t)$$

and substituting into the equation for dv/dt, a and ω_0 are determined by the following

$$a\omega_0 \cos(\omega_0 t) - \left(\frac{1}{Q}\right)\left(a \sin(\omega_0 t) - \frac{a^3}{3}\sin^3(\omega_0 t)\right) - \left(\frac{a}{\omega_0}\right)\cos(\omega_0 t) = 0$$

Substituting

$$\sin^3(\omega_0 t) = \frac{3\sin(\omega_0 t) - \sin(3\omega_0 t)}{4}$$

and using the orthogonality of the sine and cosine functions, it follows that

$$a - \left(\frac{a^3}{4}\right) = 0$$

and

$$\omega_0 - \left(\frac{1}{\omega_0}\right) = 0$$

(The $\sin(3\omega_0 t)$ term is relevant only when we consider higher-order harmonics of the oscillation.) Therefore, to the lowest order of approximation, $a = 2$ and $\omega_0 = 1$.

The only nonlinear term in the Van der Pol equations is the current-voltage term, $v^3/3$. This term differentiates the Van der Pol oscillator from the LC circuit. The small-signal conductance is the derivative with respect to voltage of the nonlinear current

$$\frac{-(1-v^2)}{Q}$$

With

$$v(t) = 2\sin t$$

the small-signal conductance as a function of time is

$$(1/Q)(1-\cos 2t)$$

Because there is a nonzero, time-varying, small-signal conductance, the PTVL model is different from the LTI LC circuit model. In fact, the time-average conductance is not even zero. However, the time-average power dissipated by the nonlinear current source is zero, a necessary condition for stable, sustained oscillation.

Oscillators are intrinsically time-varying elements because they trade off excessive gain during the low-amplitude part of the cycle with compressive effects during the remainder of the cycle. This effect is therefore a generic property not unique to this example.

To complete the noise analysis, write the differential equations that the small-signal solution $i_{s(t)}$, $v_{s(t)}$ must satisfy,

$$\frac{dv_s}{dt} = -i_s + \frac{1}{Q}(1-3\alpha v^2(t))v_s + \xi(t)$$

and

$$\frac{di_s}{dt} = v_s$$

From the large signal analysis, $v(t) = 2\sin t$, and so

$$\frac{dv_s}{dt} = -i_s + \frac{1}{Q}(2\cos 2t - 1)v_s + \xi(t)$$

and

$$\frac{di_s}{dt} = v_s$$

The time-varying conductance can mix voltages from a frequency ω to $\omega\ 2$. For small ω', if an excitation is applied at a frequency $\omega = 1 + \omega'$, i_s and v_s are expected to have components at $1 + \omega'$ and $-1 + \omega'$ for the equations to balance. (Higher-order terms are again presumed to be small.) Writing

$$i_{s(t)} = i_{e^{(i(1+\omega')t)}} + i_{-e^{(i(-1+\omega')t)}}$$

and substituting into the small-signal equations with

$$\xi(t) = c_{e^{i(1+\omega')t}}$$

leads to the following system of equations for i_+ and i_-

$$\begin{bmatrix} 1 - (1+\omega')^2 + \left(\frac{i}{Q}\right)(1+\omega') & -\left(\frac{i}{Q}\right)(-1+\omega') \\ -\left(\frac{i}{Q}\right)(1+\omega') & 1 - (-1+\omega')^2 + \left(\frac{i}{Q}\right)(-1+\omega') \end{bmatrix} \begin{bmatrix} i_+ \\ i_- \end{bmatrix} = \begin{bmatrix} c_+ \\ 0 \end{bmatrix}$$

Solving these equations gives the transfer function from an excitation at frequency $1 + \omega'$ to the small-signal at frequency $1 + \omega'$ that we call $H_0(\omega')$. A similar analysis gives the other significant transfer function, from noise at frequency $-1 + \omega'$ of amplitude C_ to the small-signal response at frequency $1 + \omega'$, that we call $H_{-2}(\omega')$. In the present case, for small ω',

$$H_0^2 \cong H_{-2}^2 \cong \frac{R^2}{16Q^2\left(\frac{\omega'}{\omega_0}\right)^2}$$

For a general Van der Pol circuit with a parallel resistor R that generates white current noise, $\xi(t)$, with $S_\xi(\omega) = 4\ k_B\ T/R$,

$$S_v(\omega') = \frac{k_B TR}{2\left(\frac{\omega'}{\omega}\right)^2 Q^2}$$

Note that this is precisely one-half the noise predicted by the LC model.

You can gain additional insight about phase noise by analyzing the time-domain small-signal response. The small-signal current response is.

$$i_s(t) = \frac{ie^{i\omega' t}(c_- + c_+)}{2\omega'}\sin t$$

Notice that c_+ and c_- are complex random variables that represent the relative contribution of white noise at separate frequencies. As white noise has no frequency correlations, they have uncorrelated random phase, and thus zero amplitude expectation, and unit variance in amplitude. Because the large-signal current is $i(t) = 2\cos t$, and the sine and cosine functions are orthogonal, the total noise for small ω' that we computed is essentially all phase noise.

Amplitude Noise and Phase Noise in the Linear Model

Occasional claims are made that in oscillators, “Half the noise is phase noise and half the noise is amplitude noise.” However, as the simple time-varying analysis in the previous section shows, in a physical oscillator the noise process is mostly phase noise for frequencies near the fundamental. It is true that in an LC-circuit half the total noise power corresponds to AM-like modulation and the other half to phase modulation. In the literature, the AM part of the noise is sometimes disregarded when quoting the oscillator noise although this is not always the case. (The SpectreRF circuit simulator computes the total noise generated by the circuit; see “Details of the SpectreRF Calculation” on page 1034).

However, a *linear* oscillator does not really exist. Physical oscillators operate with a tradeoff of gain that causes growing signal strength and nonlinear compressive effects that act to limit the signal amplitude. For noise calculation, the oscillator cannot be considered a linear time-invariant system because there are intrinsic nonlinear effects that produce large phase noise but limited amplitude noise. Oscillators are time-varying, and they therefore require a time-varying small-signal analysis.

Arguments that start with stationary white noise and pass it through a linear model in a forward-analysis fashion produce incorrect answers. This is true because they neglect the time-variation of the conductances (and possibly the capacitances) in the circuit. In the simple cases considered here, the conductances vary in time in a special way so as to produce no amplitude noise, only phase noise.

They have that special variation because they result from linearization about an oscillator limit cycle. An oscillator in a limit cycle has a large response to phase perturbations, but not to amplitude perturbations. The amplitude perturbations are limited by the properties of the nonlinear amplifier, but the phase perturbations can persist. The Spectre circuit simulator calculates the correct phase noise because it *knows* about the oscillator properties.

Similarly, arguments [13] that start with noise power and derive phase noise in a backwards fashion also usually produce incorrect results because they cannot correctly account for frequency correlations in the noise of the oscillator. These frequency correlations are introduced by the time-varying nature of the circuit.

Occasionally, a netlist appears in which a negative resistance precisely cancels a positive resistance to create a pure LC circuit. Because such a circuit has an infinite number of oscillation modes, the SpectreRF circuit simulator cannot correctly calculate the noise because it assumes a unique oscillation. Such a circuit is not physically realizable because adding or subtracting a microscopically small amount of conductance makes the circuit either go into nonlinear operation (amplifier saturation) or become a damped LC circuit that has a unique final equilibrium point. This equilibrium point is the zero-state solution. Trying to create the negative resistance oscillator is like trying to bias a circuit on a metastable point. Any amplitude oscillation can exist, depending on the initial conditions, as long as the amplitude is less than the amplifier saturation point.

Details of the SpectreRF Calculation

This section contains the mathematical details of how the SpectreRF circuit simulator computes noise in oscillators. Understanding the material in this section can help you troubleshoot and understand difficult oscillator problems.

The analysis the SpectreRF circuit simulator performs is similar to the simple analysis in the section "Linear Time-Varying (LTV) Models" on page 1029. During analysis, the SpectreRF circuit simulator

1. First finds the periodic steady state of the oscillator using the PSS analysis.
2. Then linearizes around this trajectory.

The resulting time-varying linear system is used to calculate the noise power density. The primary difference between the SpectreRF calculation and the previous analysis is that the basis functions used for the SpectreRF calculation are not just a few sinusoids, but rather a collection of many piecewise polynomials. The use of piecewise polynomials allows the Spectre circuit simulator to solve circuits with arbitrary waveforms, including circuits with highly nonlinear behavior.

Noise computations are usually performed with a small-signal assumption, but a rigorous small-signal characterization of phase noise is complicated because the variance in the phase of the oscillation grows unbounded over time. From a mathematical viewpoint, an oscillator is an autonomous system of differential equations with a stable limit cycle. An oscillator has phase noise because it is neutrally stable with respect to noise perturbations that move the oscillator in the direction of the limit cycle. Such *phase* perturbations persist with time, whereas transverse fluctuations are damped with a characteristic time inversely proportional to the quality factor of the oscillator.

Further care is necessary because, in general, the two types of excitations (those that create phase slippage and those responsible for time-damped fluctuations) are not strictly those that are parallel or perpendicular, respectively, to the oscillator trajectory, as is sometimes claimed (for example, in [4]).

However, one must realize that the noise powers at frequencies near the fundamental frequency correspond to correlations between points that are widely separated on the oscillator envelope. In other words, they are long-time signal effects. In fact, asymptotically (at long times), the ratio of the variance of any state variable to its power at the fundamental frequency is unity for any magnitude of the noise excitation. Therefore, in practical cases, you can consider only small deviations in the state variables when describing the phase noise.

The first step in the noise analysis is to determine the oscillator steady-state solution. This is done in the time domain using shooting methods [5]. Once the periodic steady-state is obtained, the circuit equations are linearized around that waveform in order to perform the small-signal analysis.

The time-varying linear system describing the small-signal response $\mathrm{vs}(t)$ of the oscillator to a signal $w(t)$ can be written in general form as [6, 7]

$$\left[C(t)\frac{d}{dt} + G(t)\right]v_s \equiv L(t)v_s(t) = w(t)$$

where $C(t)$ and $G(t)$ represent the linear, small-signal, time-varying capacitance and conductance matrixes, respectively. These matrixes are obtained by linearization about the periodic steady-state solution (the limit cycle). To understand the nature of time-varying linear analysis, the concept of Floquet multipliers is introduced.

Suppose $x(t)$ is a solution to the oscillator circuit equations that is periodic with period T. If $x(0)$ is a point on the periodic solution $x_L(t)$, then $x(T) = x(0)$. If $x(0)$ is perturbed slightly off the periodic trajectory, $x(0) = x_L(0) + \delta x$, then $x(T)$ is also perturbed, and in general for small δx,

$$x(T) - x_L(T) \approx \frac{\partial x(T)}{\partial x(0)}\delta x$$

The Jacobian matrix

$$\frac{\partial x(T)}{\partial x(0)}$$

is called the sensitivity matrix. The SpectreRF circuit simulator uses an implicit representation of this matrix both in the shooting method that calculates the steady-state and in the small-signal analyses. To see how the sensitivity matrix relates to oscillator noise analysis, consider the effect of a perturbation at time $t = 0$ several periods later, at $t = nT$. From the above equation,

$$x(nT) - x_L(nT) \approx \left[\frac{\partial x(T)}{\partial x(0)}\right]^n \delta x$$

so

$$x(nT) - x_L(nT) \approx \sum_i C_i \lambda_i^n \phi_i$$

where ϕ_i is an eigenvector of the sensitivity matrix. The C_i are the expansion coefficients of δx in the basis of ϕ_i. If ψ_i is a left eigenvector (an eigenvector of its transpose) of the sensitivity matrix, then

$$C_i = \psi_i^T \delta x$$

Let λ be an eigenvalue of the sensitivity matrix. In the context of linear time-varying systems, the eigenvalues λ are called *Floquet multipliers*. If all the λ have magnitude less than one (corresponding to left-half-plane poles), the perturbation decays with time and the periodic trajectory is stable. If any λ has a magnitude greater than one, the oscillation cannot be linearly stable because small perturbations soon force the system away from the periodic trajectory $x_L(t)$.

A stable nonlinear physical oscillator, however, must be neutrally stable with respect to perturbations that move it in the direction of the orbit. These are not necessarily perturbation *in* the direction of the orbit because, in general,

$$\Psi \neq \phi_\iota$$

This is true because a time-shifted version of the oscillator periodic trajectory still satisfies the oscillator equations. In other words, one of the Floquet multipliers must be equal to unity. This Floquet multiplier is responsible for phase noise in the oscillator. The associated eigenvector determines the nature of the noise.

If $\lambda = e^{\eta}$ is a Floquet multiplier, then $\eta + ik\omega_0$ is a pole of the time-varying linear system for any integer k. Therefore, because of the unity Floquet multiplier, the time-varying linear system has poles on the imaginary axis at $k\omega_0$. This is very similar to what occurs in a pure LC resonator, and it explains the identical shape of the noise profiles.

Because operator $L(t)$ has poles at the harmonics of the oscillation frequency, numerical calculations of the noise at nearby frequencies become inaccurate if treated in a naive manner [8, 9]. To correctly account for the phase noise, the SpectreRF circuit simulator finds and extracts the eigenvector that corresponds to the unity Floquet multiplier. To correctly extract the phase noise component, both the right and left eigenvectors are required. Once these vectors are obtained, the singular (phase noise) contribution to the noise can be extracted. The remaining part of the noise can be obtained using the usual iterative solution techniques [6] in a numerically well-conditioned operation.

In Figure A-2 on page 1026, you can see that the SpectreRF PTVL analysis correctly predicts the total noise, including the onset of 3 dB amplitude noise outside the bandwidth of the resonator. Note that this simulation was conducted at

$$E\left\{\xi^2(t)\right\} = 10^{-3}$$

which represents a very high noise level that is several orders of magnitude higher than in actual circuits. The good match of the PTVL models to the full nonlinear simulation shows the validity of the PTVL approximation.

Calculating Phase Noise

The following sections suggest simulation parameters, give you tips for using these parameters, and advise you about checking for accuracy.

Setting Simulator Options

The SpectreRF time-varying small-signal analyses are more powerful than the standard large-signal analyses (DC, TRAN) but, like any precision instrument, they also have greater sensitivity to numerical errors. For many circuits, particularly oscillators, more simulator precision is needed to get good results from the PAC, PXF, and Pnoise calculations than is needed to get good DC or TRAN results.

The small-signal analyses operate by linearizing around the periodic steady state solution. Consequently, the oscillator noise analysis, and the periodic small-signal analyses in general, inherit most of their accuracy properties from the previous PSS simulation. You must be sure the PSS simulation generates a sufficiently accurate linearization. See “What Can Go Wrong” on page 1041 for a discussion. See also “Tips for Getting a PSS Analysis to Converge” on page 1039.

Table A-1 recommends simulator options for various classes of circuits.

Table A-1 Recommended SpectreRF Parameter Values

Circuit	reltol	vabstol	iabstol
Easy	$1.0e^{-4}$	default	default
Hard-I	$1.0e^{-5}$	10n	1p
Hard-II	$1.0e^{-6}$	1n	0.1p
Hard-III	$1.0e^{-7}$	0.1n	0.1p

- *Easy* circuits are low-Q (about $Q < 10$) resonant oscillators, ring oscillators, and weakly-nonlinear relaxation oscillators. Most *textbook* circuits are in this category.
- *Hard-I* circuits are most other resonant oscillators; circuits with complicated AGC, load, or bias circuitry; and relaxation or ring oscillators that exhibit moderate to strong nonlinear or *stiff* effects. This is the best general-purpose set of options.
- *Hard-II* and *Hard-III* circuits include a few particularly difficult circuits.
    - Usually these options are used only in a convergence study (see “How to Tell if the Answer Is Correct” on page 1040) or for circuits that previously failed a conversion study using less strict options.
    - Circuits in this category often exhibit some form of unusual behavior (see “What Can Go Wrong” on page 1041).
    - Sometimes this behavior results from circuit properties, for example, some very high-Q crystal oscillators and some very stiff relaxation oscillator circuits. Occasionally, the behavior reflects a design flaw.

Usually setting `method=gear2only` is recommended for the PSS simulation (but see “What Can Go Wrong” on page 1041).

Tips for Getting a PSS Analysis to Converge

You can get most circuits to converge by manipulating the `tstab` and `steadyratio` parameters. Set `tstab` large enough so that the oscillation amplitude increases to near its steady-state value and most other transients have died out. You can estimate the value of `tstab` either by performing a TRAN analysis, or by performing a PSS analysis itself with the setting `saveinit = yes`. At tighter simulation tolerances, if `steadyratio` is too small, the PSS simulation often will not converge. Setting `steadyratio = 0.1` usually fixes this problem.

For Large or Particularly Difficult circuits

For particularly difficult circuits, or for large circuits that make the above procedure excessively time-consuming, you can use a slightly different procedure.

Try the options settings `reltol = 1e`$^{-3}$ and `steadyratio = 1`, and run the PSS analysis with a very long `tstab` parameter setting. You might also need to relax `iabstol` and `vabstol`. Save this solution to a file using the `writefinal` option. This step can usually obtain a low-accuracy PSS solution with an acceptable simulation time. Using a very long `tstab` increases the probability that the simulation converges, and relaxing `reltol` ensures a reasonable simulation time and increases the probability of PSS convergence.

For some circuits, the oscillation might die out before the oscillator builds up a final level, or the circuit might oscillate for a while before returning to a zero-state. Setting `saveinit = yes` lets you view the initial transient waveforms to determine if this problem is occurring. This problem might be caused by difficulty starting the oscillator, or it might be the result of artificial numerical losses introduced by the very large timesteps. This last cause is likely if the PSS options `method` parameter was set to `gear2only`, `gear2`, or `euler`. In these cases, decrease `reltol` or set the `maxstep` parameter to make the simulator use smaller step sizes.

Once the initial PSS simulation has completed, reset the accuracy parameters `reltol`, `vabstol`, and `iabstol` to their preferred final values. Then rerun the PSS simulation using the `readic` option to read in the initial conditions saved from the first, low-accuracy, PSS analysis. You might need to leave the `steadyratio` setting in the 0.1 to 1 range to achieve convergence. Any value of `steadyratio` less than 1 or so should be acceptable.

If the circuit contains independent sources used to start the oscillator, set the PSS start time to a large enough value to be sure these sources are all inactive at the start of the simulation.

You need not use a large `tstab` value in this second step. However, varying `tstab` slightly in this second analysis can sometimes help secure convergence.

Some users report that decreasing the maximum allowed timestep sometimes helps convergence. To do this, either decrease the `maxstep` parameter or increase the `maxacfreq` parameter.

How to Tell if the Answer Is Correct

To be sure that only small numerical errors are introduced into the phase noise calculation, you can simulate the oscillator with progressively more stringent accuracy parameters until the change in the calculated noise is less than the desired simulation precision. Such a set of simulations typically starts from the *Easy* parameter set given in Table A-1 on page 1038 and proceeds downward through the table until the calculated noise no longer changes. Remember that generally you must lower *all three* tolerances (`reltol`, `vabstol`, and `iabstol`) to ensure that the discrete approximations used by the Spectre circuit simulator are converging to the continuous solutions of the physical circuit.

Using the final solutions from the previous simulations as an initial estimate for the next PSS simulations can help minimize the total PSS simulation time. Use the `writefinal` parameter in PSS to write out each final PSS solution and the `readic` parameter to read it back in.

You can sometimes identify problem circuits or simulations by changing the `tstab` parameter. See "The tstab Parameter" on page 1045 for details.

Troubleshooting Phase Noise Calculations

The SpectreRF circuit simulator calculates noise effectively for most oscillators. However, circuits that are very stiff, very nonlinear, or just poorly designed can occasionally cause problems for the simulator. Stiff circuits exhibit dynamics with two or more very different time scales; for example, a relaxation oscillator with a square-wave-like periodic oscillation. Over most of the cycle, the voltages change very slowly, but occasional rapid transitions are present. This section describes some of the reasons for the problems, what goes wrong, how to identify problems, and how to fix them.

See "Details of the SpectreRF Calculation" on page 1034 for help troubleshooting particularly difficult circuits.

Known Limitations of the Simulator

Any circuit that does not have a stable periodic steady-state cannot be analyzed by the SpectreRF circuit simulator because oscillator noise analysis is performed by linearizing around a waveform that is assumed to be strictly periodic.

For example, oscillators based on IMPATT diodes generate strong subharmonic responses and cannot be properly analyzed with the SpectreRF circuit simulator. As another example, Colpitts oscillators, properly constructed, can be made to exhibit chaotic as well as subharmonic behavior.

Similarly, any circuit with significant large-signal response at tones other than the fundamental and its harmonics might create problems for the simulator. Some types of varactor-diode circuits might fit this category. In addition, some types of AGC circuitry and, on occasion, bias circuitry can create these effects.

The SpectreRF circuit simulator cannot simulate these circuits because simulation of an autonomous circuit with subharmonic or other aperiodic components in the large signal response essentially requires foreknowledge of which frequency components are important. Such foreknowledge requires Fourier analysis of very long transient simulations and cannot be easily automated. Such simulations can be very expensive.

What Can Go Wrong

The SpectreRF circuit simulator can have problems in the following situations.

Generic PSS Simulation Problems

Any difficulties in the underlying PSS analysis affect the phase noise computation. For example, underestimating the oscillator period or failing to start the oscillator properly can cause PSS convergence problems that make running a subsequent Pnoise analysis impossible.

Hypersensitive Circuits

Occasionally, you might see circuits that are extremely sensitive to small parameter changes. Such a circuit was a varactor-tuned VCO that had the varactor bias current, and therefore the oscillation frequency, set by a $1\ \mathrm{T}\Omega$ resistor. Changing to a $2\ \mathrm{T}\Omega$ resistor, which is a $1e^{-12}$ relative perturbation in the circuit matrixes, changed the oscillation frequency from 125 MHz to 101 Mhz. Such extreme circuit sensitivity results in very imprecise PSS simulations. In particular, the calculated periods have relatively large variations. If precise PSS simulations are impossible, precise noise calculations are also impossible. In such a case, you must fix the circuit.

Subharmonics or Parametric Oscillator Modulation

Sometimes bias and AGC circuitry might create small-amplitude parasitic oscillations in the large signal waveform. You can identify these oscillations by performing a transient simulation to steady-state and then looking for modulation of the envelope of the oscillation waveform. For high-Q circuits and/or low-frequency parasitics, this transient simulation might be very long.

In this case, because the oscillator waveform is not actually periodic, the PSS simulation can only converge to within approximately the amplitude of the parasitic oscillation. If the waveform possesses a parasitic oscillation that changes amplitude, over one period, around 10^{-5} relative to the oscillator envelope, then convergence with `reltol` $< 10^{-5}$ is probably not possible (assuming `steadyratio` is one or less).

These effects might also appear as a parametric sideband amplification phenomenon. See “Frequently Asked Questions” on page 1045 for more information.

Small-Signal Frequency is Much Higher than the Fundamental Frequency

The same timesteps are used for both the small-signal analysis and the PSS analysis. If the small-signal frequency is much higher than the fundamental frequency, much smaller timesteps might be required to accurately resolve the small-signal than are needed for the large signal. To force the Spectre circuit simulator to take sufficiently small timesteps in the PSS simulation, be sure the `maxacfreq` parameter is set correctly.

Wide Timestep Variation

Occasionally, in simulations that generate PSS waveforms with timesteps that vary over several orders of magnitude, the linear systems of equations that determine the small-signal response become ill-conditioned. As a result, the noise analysis is inaccurate. Usually this occurs because you have requested excessive simulator precision; for example, nine-digit precision. You can sometimes eliminate this problem using `method = traponly` in the PSS solution. You might also set `maxstep` to a very small value in the PSS analysis or you might specify a very large `maxacfreq` value.

Problems with Device Models

When the device models leave their physically meaningful operating range during the large-signal PSS solution, the noise calculations are usually inaccurate. Similarly, when the models are discontinuous, or have discontinuous derivatives, the small-signal analysis might be inaccurate.

Problems Resolving Floquet Multipliers in Stiff Relaxation Oscillators

Sometimes in very stiff relaxation oscillators, the PSS solution rapidly and easily converges; but the numerically calculated Floquet multiplier associated with the PSS solution is far from unity. Typically, this multiplier is real and has a magnitude much larger than unity. The SpectreRF circuit simulator prints a warning (see “Message III” on page 1044). It is interesting that sometimes the phase noise is quite accurate even with low simulation tolerances. If you have this problem, perform a convergence study (see “How to Tell if the Answer Is Correct” on page 1040).

Problems Resolving Floquet Multipliers in High-Q Resonant Circuits

In a physical oscillator, there is one Floquet multiplier equal to unity. In an infinite-Q linear resonator, however, the multipliers occur in complex conjugate pairs. A very high-Q nonlinear oscillator has another Floquet multiplier on the real axis nearly equal to, but slightly less than, one. In this presence of numerical error, however, these two real Floquet multipliers can appear to the simulator as a complex-conjugate pair. The phase noise is computed using the Floquet vector associated with the unity Floquet multiplier. When the two multipliers appear as a complex pair, the relevant vector is undefined. When the Spectre circuit simulator correctly identifies this situation, it prints a warning (see “Message III” on page 1044). The solution is usually to simulate using the next higher accuracy step (see Table A-1 on page 1038). Sometimes varying `tstab` can also help with this problem.

If the circuit is really an infinite-Q resonator (for example, a pure parallel LC circuit), the multipliers always appear as complex conjugate pairs and the noise computations are not accurate close to the fundamental frequency. Such circuits are not physical oscillators, and the Spectre circuit simulator is not designed to deal with them; see “Amplitude Noise and Phase Noise in the Linear Model” on page 1033 and “Frequently Asked Questions” on page 1045.

Phase Noise Error Messages

SpectreRF displays error messages when it encounters several types of known numerical difficulty. To interpret the error messages produced by the phase noise analysis, you must know the material in “Details of the SpectreRF Calculation” on page 1034.

Message I

The Floquet eigenspace computed by spectre PSS analysis appears to be inaccurate. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol and method=gear2only.

The eigenvector responsible for phase noise was inaccurately computed and the PSS simulation tolerances might be too loose. Try simulating the circuit at the next higher accuracy setting (see Table A-1 on page 1038) and then compare the calculated noise in the two simulations.

Message II

The Floquet eigenspace computed by spectre PSS analysis appears to be ill-defined. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol, different tstab(s), and method=gear2only. Check the circuit for unusual components.

This can be an accuracy problem, or it can result from an unusual circuit topology or sensitivity. Tighten the accuracy requirements as much as possible (see Table A-1 on page 1038). If this message appears in all simulations, the noise might be incorrect even if the simulations agree.

Message III

The Floquet eigenspace computed by spectre PSS analysis appears to be inaccurate and/or the oscillator possesses more than one stable mode of oscillation. PNOISE computations may be inaccurate. Consider re-running the simulation with smaller reltol, different tstab(s), and method=gear2only.

All the real Floquet multipliers were well-separated from unity, suggesting that the PSS simulation tolerances might be too loose. Simulate the circuit at the next higher accuracy setting (see Table A-1 on page 1038) and then compare the calculated noise in the two simulations. If the calculated noise does not change, it is probably correct even if this message appears in both simulations.

The tstab Parameter

Because SpectreRF performs the PSS calculation in the time domain by using a *shooting method*, an infinite number of possible PSS solutions exist, depending on where the first timepoint of the PSS solution is placed relative to the oscillator phase.

The placement of the first timepoint is determined by the length of the initial transient simulation, which you can control using the `tstab` parameter. If the `tstab` value causes the edges of the periodic window to fall on a point where the periodic oscillator waveform is making very rapid transitions, it will probably be very difficult for PSS to converge. Similarly, the results of the small-signal analyses probably will not be very accurate. Avoid such situations. If the start of the PSS waveform falls on a very fast signal transition, you usually need to view the results of further small-signal analyses with some skepticism.

Although a poor choice of the `tstab` parameter value can degrade convergence and accuracy, appropriate use of `tstab` can help to identify problem circuits and to estimate the reliability of their noise computations.

If you perform several PSS and Pnoise computations that differ only in their `tstab` parameter values, the results should be fairly similar, within a relative deviation of the same order of magnitude as the simulator parameter `reltol`. If this is not the case, you might not have set the simulator accuracy parameters sufficiently tight to achieve an accurate solution; and you need to reset one or more of the parameters `reltol`, `vabstol`, or `iabstol`. The circuit might also be poorly designed and very sensitive to perturbations in its parameters.

If the calculated fundamental period of the oscillator varies with `tstab` even when you set `reltol`, `iabstol`, and `vabstol` to very small (but not vanishingly small) values, the circuit is probably poorly designed, exhibiting anomalous behavior, or both. (see “Known Limitations of the Simulator” on page 1041).

Frequently Asked Questions

The following questions are similar to those commonly asked about oscillator noise analysis with the SpectreRF circuit simulator.

Does SpectreRF simulation calculate phase noise, amplitude noise, or both?

SpectreRF simulation computes the total noise of the circuit, both amplitude and phase noise. What the analog circuit design environment plots as *phase noise* is really the total noise scaled by the power in the fundamental oscillation mode. Close enough to the fundamental frequency, the noise is all phase noise, so what the analog circuit design environment plots of *phase noise* is really the phase noise as long as it is a good ways above the noise floor.

Some discussions of oscillator noise based on a simple resonator/amplifier description describe the total noise, at small frequency offsets from the fundamental, as being half amplitude noise and half phase noise. In reality, for physical oscillators, near the fundamental nearly all the noise is phase noise. Therefore, these simple models overestimate the total noise by 3 dB. For a detailed explanation, see the phase noise theory described in "Details of the SpectreRF Calculation" on page 1034 and the detailed discussion of the Van der Pol oscillator "Linear Time-Varying (LTV) Models" on page 1029.

I have a circuit that contains an oscillator. Can I simulate the oscillator separately and use the phase noise Spectre calculates as input for a second PSS/PNOISE simulation?

No. Oscillators generate noise with correlated spectral sidebands. Currently, SpectreRF simulation output represents only the time-average noise power, not the correlation information, so the noise cannot be input to a simulation that contains time-varying elements that might mix together noise from separate frequencies.

If the second circuit is a linear filter (purely lumped linear time-invariant elements, such as resistors, capacitors, inductors, or a linearization of a nonlinear circuit around a DC operating point) that generates no frequency mixing, then you can use the output of the SpectreRF Pnoise analysis as a *noisefile* for a subsequent NOISE (not Pnoise) analysis.

How accurate are the phase noise calculations? What affects the errors?

Initially, it is important to distinguish between modeling error and simulation (numerical) error. If the device models are only good to 10% the simulation is only good to 10% (or worse). So, for the rest of this appendix, we discuss numerical error introduced by the approximations in the algorithms.

You must also distinguish between absolute and relative signal frequencies in the noise analysis. When the noise frequency is plotted on an absolute scale, the error is primarily a function of the variance in the calculated fundamental period. This is true because of the singular behavior, in these regions, of the phase noise near a harmonic of the fundamental. To see this behavior, note that for the simple oscillator driven by white noise, the noise power is proportional to the offset from the fundamental frequency,

$$S_v(\omega) \propto \frac{1}{(\omega - \omega_0)^2}$$

If you make a small error in the calculation of ω_0, the error $\Delta\, S_v$ in the noise is proportional to S / ω_0

$$\Delta S_v(\omega) \propto \frac{\Delta\omega_0}{(\omega - \omega_0)^3}$$

This error can be very large even if $\Delta\omega_0$, the error in ω_0, is small. However, because of the way SpectreRF simulation extracts out the phase noise, the calculated phase noise, as a function of offset from the fundamental frequency, can be quite accurate even for very small offsets.

Now consider how much error is present in the calculated fundamental frequency. Because the numerical error is related to many simulation variables, it is difficult to quantify, without examination, how much is present. However, as a rough approximation, if we define the quantity

$$r = min\left\{reltol, \frac{iabstol}{max(i)}, \frac{vabstol}{max(v)}\right\}$$

where $max(i)$ and $max(v)$ are the maximum values of current and voltage over the PSS period, then, under some assumptions, $\Delta\,\omega_0$, the error in the fundamental ω_0, probably satisfies

$$r\omega_0 < \delta\omega_0 < Mr\omega_0$$

where M is the number of timesteps taken for the PSS solution. This analysis assumes that `steadyratio` is sufficiently tight, not much more than one, and also that `iabstol` and `vabstol` are sufficiently small.

If you require a good estimate of the accuracy in the fundamental, run the PSS simulation with many different accuracy settings, initial conditions and/or `tstab` values (see “How to Tell if the Answer Is Correct” on page 1040 and “The tstab Parameter” on page 1045). For example, to estimate how much numerical error remains in the calculated fundamental frequency for a given simulation, run the simulation; reduce `reltol`, `iabstol`, and `vabstol` by a factor of 10 to100; rerun the simulation; and then compare the calculated fundamental frequencies. For the sorts of parameters we recommend for oscillator simulations, four to five digits of precision seems typical. Past that point, round off error and anomalous effects introduced by vastly varying timesteps offset any gains from tightening the various accuracy parameters.

For phase noise calculations, again it is unrealistic to expect relative precision of better than the order of `reltol`. That is, if `reltol` is 10^{-5} and the oscillator fundamental is about 1 GHz, the SpectreRF numerical fuzz for the calculated period is probably about 10 KHz. Therefore, when plotted on an absolute frequency scale, the phase noise calculation exhibits substantial variance within about 10 KHz of the fundamental.

However, when plotted on a frequency scale *relative* to the fundamental, the phase noise calculation might be more precise for many oscillators. If the circuit is strongly dissipative (that is, low-Q, such as ring oscillators and relaxation oscillators), the phase noise calculation is probably fairly accurate up to very close to the fundamental frequency even with loose simulation tolerance settings. High-Q circuits are more demanding of the simulator and require more stringent simulation tolerances to produce good results. In particular, circuits that use varactor diodes as tuning elements in a high-Q tank circuit appear to cause occasional problems. Small modifications to the netlist (runs with different `tstab` values and minor topology changes) can usually tell you whether (and where) the simulator results are reliable.

Simulation accuracy is determined by how precisely SpectreRF simulation can solve the augmented nonlinear boundary value problem that determines the periodic steady-state. The accuracy of the BVP solution is controlled primarily by the simulation variables `reltol`, `iabstol`, `vabstol`, `steadyratio`, and `lteratio`. Typically, `steadyratio` and `lteratio` are fixed, so `reltol` is usually the variable of interest.

Occasionally accuracy might be somewhat affected by other variables such as `relref`, `method`, the number of timesteps, and `tstab`. Again, the physical properties of the circuit might limit the accuracy.

I have a circuit with an oscillator and a sinusoidal source. Can I simulate this circuit with SpectreRF simulation?

In general, SpectreRF simulation is not intended to analyze circuits that contain autonomous oscillators and independent periodic sources.

If the circuit contains components that could potentially oscillate autonomously and also independent large-signal sinusoidal sources, SpectreRF simulation works properly only if two conditions are fulfilled. The system must be treated as a driven system, and the coupling from the sinusoidal sources to the oscillator components must be strong enough to lock the oscillator to the independent source frequency. (In different contexts, this is known as *oscillator entrainment* or *phase-locking*) The normal (nonautonomous) PSS and small-signal analyses function normally in these conditions.

If the autonomous and driven portions of the circuit are weakly coupled, the circuit waveform might be more complicated; for example, a two-tone (quasi-periodic) signal with incommensurate frequencies. Even if PSS converges, further small-signal analyses (PAC, PXF, Pnoise) almost certainly give the wrong answers.

What is the significance of total noise power?

First, you must understand that SpectreRF simulation calculates and measures noise in voltages and currents. The total power in the phase process is unbounded, but the power in the actual state variables is bounded.

Oscillator phase noise is usually characterized by the quantity

$$d(f) = \frac{Sv(f)}{P_1}$$

where P_1 is the power in the fundamental component of the steady state solution and $Sv(f)$ is the power spectral density of a state variable V.

For an oscillator with only white-noise sources, $L(f)$ has a Lorentzian line shape,

$$L(f) = \frac{1}{\pi}\frac{a}{a^2 + f^2}$$

where a is dependent on the circuit and noise sources, and thus the total phase noise power

$$\int L(f)df = 1$$

Because

$$varv(t) = R_v(t,t) = \int_{-\infty}^{\infty} S_v(f)df$$

we are led to the uncomfortable, but correct, conclusion that the variance in any variable is 100 percent of the RMS value of the variable, *irrespective of circuit properties or the amplitude of the noise sources*.

Physically, this means that if a noise source has been active, since $t = -\infty$, then the voltage variable in question is randomly distributed over its whole trajectory. Therefore, the relative variance is one. Clearly, the variance is not a physically useful characterization of the noise, and the total noise power must be interpreted carefully. What is actually needed is the variance as a function of time, given a fixed reference for the signal in question; or, more often, the rate at which the variance increases from a zero point; or, sometimes, the increment in the variance from cycle to cycle. That is, we want to specify the phase of the oscillator signal at a given time point and to find a statistical characterization of the variances relative to that time. But because of the non-causal nature of the Fourier integral, quantities like the total noise power give us information about the statistical properties of the signal over all time.

What's the story with pure linear oscillators (LC circuits)?

Oddly enough, SpectreRF simulation is not set up to do Pnoise analysis on pure LC circuits.

Pure LC circuits are not physically realizable oscillators, and the mathematics that describes them is different from the mathematics that describes physical oscillators. A special option must be added to the code in order for Pnoise to handle *linear oscillators*. See "Models for Phase Noise" on page 1027, and, in particular, "Amplitude Noise and Phase Noise in the Linear Model" on page 1033. Because the normal NOISE analysis is satisfactory for these circuits and also much faster, it is unlikely that Pnoise will be modified.

Why doesn't the SpectreRF model match my linear model?

As is discussed in "Amplitude Noise and Phase Noise in the Linear Model" on page 1033, the difference between the SpectreRF model (the correct answer) and the linear oscillator model is that in the linear oscillator, both the amplitude and the phase fluctuations can become large. However, in a nonlinear oscillator, the amplitude fluctuations are always bounded, so the noise is half as much, asymptotically.

We emphasize that computing the correct total noise power requires using the time-varying small signal analysis. An oscillator is, after all, a time-varying circuit by definition. Time-invariant analyses, like the *linear oscillator model*, can sometimes be useful, but they can also be misleading and should be avoided.

There are funny sidebands/spikes in the oscillator noise analysis. Is this a bug?

Very possibly this is parametric small-signal amplification, a real effect. This sometimes occurs when there is an AGC circuit with a very long time constant modulating the parameters of circuit elements in the oscillator loop. Sidebands in the noise power appear at frequencies offset from the oscillator fundamental by the AGC characteristic frequency.

Similarly, any elements that can create a low-frequency parasitic oscillation, such as a bias inductor resonating with a capacitor in the oscillator loop, can create these sorts of sidebands.

Further Reading

The best references on the subject of phase noise are by Alper Demir and Franz Kaertner. Alper Demir's thesis [10], now a Kluwer book, is a collection of useful thinking about noise. Kaertner's papers [11, 12, 9] contain a reasonably rigorous and fairly mathematical treatment of phase noise calculations.

The book by W. P. Robins [13] has a lot of engineering-oriented thinking. However, it makes heavy use of LTI models, and much of the discussion about noise cannot be strictly applied to oscillators. As a consequence, you must interpret the results in this book with care.

Hajimiri and Lee's paper [4] is worth reading, but their analysis is superseded by Kaertner's.

Other references include [8, 14, 15, 16].

References

[1] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1995.

[2] D. Leeson, "A simple model of feedback oscillator noise spectrum," *Proc. IEEE*, vol. 54, pp. 329–330, 1966.

[3] W. A. Gardner, *Introduction to random processes*. McGraw Hill, 1990.

[4] A. Hajimiri and T. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of Solid. State Circuits*, vol. 33, pp. 179–193, 1998.

[5] R. Telichevesky, J. White, and K. Kundert, "Efficient steady-state analysis based on matrix-free krylov-subspace methods," in *Proceedings of 32rd Design Automation Conference*, June 1995.

[6] R. Telichevesky, J. White, and K. Kundert, “Efficient AC and noise analysis of two-tone RF circuits,” in *Proceedings of 33rd Design Automation Conference*, June 1996.

[7] M. Okumura, T. Sugaware, and H. Tanimoto, “An efficient small-signal frequency analysis method of nonlinear circuits with two frequency excitations,” *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 225–235, 1990.

[8] W. Anzill and P. Russer, “A general method to simulate noise in oscillators based on frequency domain techniques,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 41, pp. 2256–2263, 1993.

[9] F. X. Kärtner, “Noise in oscillating systems,” in *Proceedings of the Integrated Nonlinear Microwave and Millimeter Wave Circuits Conference*, 1992.

[10] A. Demir, *Analysis and simulation of noise in nonlinear electronic circuits and systems*. PhD thesis, University of California, Berkeley, 1997.

[11] F. X. Kaertner, “Determination of the correlation spectrum of oscillators with low noise,” *IEEE Trans. Microwave Theory and Techniques*, vol. 37, pp. 90–101, 1989.

[12] F. X. Kaertner, “Analysis of white and f-a noise in oscillators,” *Int. J. Circuit Theory and Applications*, vol. 18, pp. 485–519, 1990.

[13] W. P. Robins, *Phase Noise in Signal Sources*. Institution of Electrical Engineers, 1982.

[14] A. A. Abidi and R. G. Meyer, “Noise in relaxation oscillators,” *IEEE J. Sol. State Circuits*, vol. 18, pp. 794–802, 1983.

[15] B. Razavi, “A study of phase noise in cmos oscillators,” *IEEE J. Sol. State Circuits*, vol. 31, pp. 331–343, 1996.

[16] K. Kurokawa, “Noise in synchronized oscillators,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 16, pp. 234–240, 1968.

B

Using PSS Analysis Effectively

Periodic steady-state (PSS) analysis is a prerequisite for all periodic small-signal analyses such as the Periodic AC (PAC), Periodic Transfer Function (PXF), Periodic S-Parameter (PSP) and Periodic Noise (Pnoise) analyses provided by the SpectreRF circuit simulator. PSS provides a rich set of parameters to help you adapt it to your own applications. For most circuits, PSS converges with the default parameter values. However, for some difficult circuits, changing the values of some parameters is necessary to achieve convergence.

This appendix describes methods you can use to remedy nonconvergence. This appendix also tells you how to improve convergence and efficiency using hierarchical PSS runs.

This appendix is divided into the following three main sections:

“General Convergence Aids” on page 1053 describes techniques you can use to resolve PSS nonconvergence with both driven and autonomous circuits

“Convergence Aids for Oscillators” on page 1055 describes techniques you use only with autonomous circuits such as oscillators

“Running PSS Analysis Hierarchically” on page 1056 describes how to run a sequence of PSS analyses to improve the convergence, efficiency, and quality of the PSS solution used in subsequent periodic small-signal analyses

General Convergence Aids

You can use the convergence aids described in this section to remedy PSS nonconvergence with driven as well as autonomous circuits. Autonomous circuits are usually harder to converge than driven circuits.

Adjusting the steadyratio and tstab Parameters

You can converge most difficult circuits by manipulating the `steadyratio` and `tstab` parameters.

The `steadyratio` parameter guards against false convergence. However, in unusual situations, the default value for `steadyratio` might be too conservative (1.0e^{-3} is the default).

The PSS convergence criteria (for voltage-valued variables) is roughly

$$|\Delta v| < (reltol \times |vsig| + vabstol) \times steadyratio \times lteratio$$

To solve the periodic steady-state problem, SpectreRF simulation replaces the time derivatives in the one-period time interval with discrete differences. This turns the nonlinear, continuous differential equations into a set of discrete nonlinear equations that the SpectreRF circuit simulator can solve.

The `steadyratio` parameter specifies the accuracy requirements for the discrete system. The discrete system might have its own solution (a steady state of the discrete difference equations) independent of what is happening in the continuous limit. In other words, the discrete system might be solved to zero tolerance. This happens frequently, particularly in driven circuits which is why setting a conservative `steadyratio` value generally works quite well.

In some cases, however, the solution to zero tolerance might not occur oscillators seem to be especially problematic. Avoid setting the convergence tolerance too tightly. For example, if `reltol` **=** 1.0e^{-6} and `steadyratio` **=** 1.0e^{-3}, the relative tolerance for solving the discrete system is approximately `reltol` × `steadyratio` **=** 1.0e^{-9}. This tolerance level approaches the limit of precision the simulator can provide. In this situation, loosen `steadyratio` to 1.0 or 0.1 and reduce the precautions against false convergence.

Providing a larger value for `tstab` usually improves convergence. Occasionally, you must set `tstab` to a value equal to or greater than the time needed for the circuit to reach approximate steady state.

Additional Convergence Aids

Below is a list of additional suggestions that sometimes help convergence.

- Carefully evaluate and resolve any notice, warning, or error messages.
- While trapezoidal rule ringing is simply annoying in transient analysis, in PSS analysis it can cause the shooting iteration to stall before convergence is achieved. You can remedy this problem by changing the PSS options `method` parameter from `traponly` to either `trap`, `gear2`, or `gear2only`.

- Help convergence by increasing the `maxperiods` parameter to increase the maximum iterations for shooting method to use. Sometimes a PSS analysis might simply need more than the default number of iterations to converge. However, in some situations convergence does not occur regardless of the number of iterations. In this case, increasing the iteration limit simply causes the simulation to take longer to fail.
- Decrease the maximum allowed time step to help convergence. To adjust the time step, either decrease the `maxstep` parameter or increase the `maxacfreq` parameter.
- Use the `errpreset` parameter properly. Use `liberal` for digital circuits; use `moderate` for typical analog circuits; use `conservative` for sensitive analog circuits (for example, charge storage circuits).

Convergence Aids for Oscillators

Oscillator circuits are usually harder to converge than their driven counterparts. In addition to manipulating the `steadyratio` parameter, as discussed in "General Convergence Aids" on page 1053, set the `tstab` parameter large enough so that the oscillation amplitude increases almost to its steady-state value and most other transients die out. You can estimate the required value of `tstab` by performing a transient analysis, or in the PSS analysis itself, set `saveinit = yes`.

For some circuits, the oscillation might die out before the oscillator builds up a final value, or the circuit might oscillate temporarily but then return to a zero state. Setting `saveinit = yes` lets you view the initial transient waveforms to identify the problem. This problem might be due to difficulty in starting the oscillator, or it might be caused by artificial numerical losses introduced by very large time steps. The latter is particularly likely if you set the `method` parameter to `gear2only`, `gear2`, or `euler`. In this case, you might try using `method = traponly`. If the problem persists, force the simulator to use smaller step sizes by decreasing `reltol` or by setting the `maxstep` parameter.

With autonomous PSS analysis, exclusive use of the trapezoidal rule can lead to ringing that spans the length of the oscillation period and causes convergence problems. When you set `method = trap`, the SpectreRF circuit simulator occasionally takes a backward Euler step, which acts to damp the ringing. The `gear2` and `gear2only` methods use Gear's backward difference method, which is not subject to ringing. Each of these alternatives to `traponly` avoids trapezoidal rule ringing and the attendant convergence problems at the expense of adding a small amount of artificial numerical damping. This damping slightly reduces the computed Q of the oscillator.

Be sure that the method you choose to start your oscillator is effective. It must *kick* the oscillator hard enough to start the oscillation and make the oscillator respond with a signal level that is between 25 and 100 percent of the expected final level. Avoid kicking the oscillator so hard that it responds in an unnecessarily nonlinear fashion. Also try to avoid exciting response modes in the circuit that are unrelated to the oscillation, especially those associated with long time constants.

Try to improve your estimate of the period. The simulator uses your estimate of the period to determine the length of the initial transient analysis interval. This interval is used to measure the oscillation period. If you specify a period that is too short, the estimate of the oscillation period is not accurate, and the PSS analysis might fail. Overestimation of the period is not a serious problem because the only disadvantage is a longer initial transient interval. However, significant overestimation can result in an excessively long simulation time.

Sometimes the analysis might need more than the default number of iterations (`maxperiods = 50`) to converge. This is more likely to occur with high-Q circuits. You can increase the maximum iterations for shooting methods using the `maxperiods` parameter.

Running PSS Analysis Hierarchically

For most circuits, a single PSS analysis run is sufficient to find the periodic steady-state solution. However, for some difficult circuits it is preferable, or even necessary, to run PSS analysis multiple times to find the steady-state solution with the parameter settings you want (for example, with a tight `reltol`).

The biggest obstacle to PSS convergence is poorly chosen initial conditions. The backbone of PSS analysis is Newton's method. Theoretically, when the initial guess is close enough to the solution and the problem is not ill-conditioned, Newton's method is guaranteed to converge because of its contraction property. Consequently, it is very important to provide the best initial conditions that you can to ensure rapid convergence.

To run PSS hierarchically, you

1. Start by running a minimally accurate PSS analysis to obtain, with high likelihood, a coarse-grid solution that converges in an acceptably short simulation time.
2. Use the coarse-grid solution as the initial condition for another PSS run to achieve a fine-grid solution.

This practice significantly increases the chance that the fine-grid PSS analysis will converge promptly. As you might expect, it is also more efficient. Running PSS hierarchically often reduces the total simulation time needed to find a periodic steady-state solution because it reduces the number of time-consuming fine-grid PSS iterations.

The `writefinal` and `readic` parameters serve as threads between hierarchical PSS runs.

- When you set `writefinal` to *SomeFileName*, both the associated time and period information (for autonomous PSS) and the final transient solution at PSS steady-state are saved to the file, *SomeFileName*.
- When you run another similar PSS analysis and you set `readic` to *SomeFileName* and `skipdc` to `yes`, setting `skipdc` to `yes` forces the simulator to use the initial conditions in the file *SomeFileName* as the initial transient solution for the first PSS iteration.

For example, if you want to find the periodic steady-state solution with a tight `reltol` such as $1.0e^{-5}$, you might

- Run an initial PSS analysis with a looser tolerance; for example, `reltol = ` $1.0e^{-3}$
- Use the `writefinal` parameter in the initial PSS analysis to write out the final results to a file. A PSS analysis runs faster with the looser tolerance because fewer time points are generated during each transient integration performed during each PSS iteration.
- Run a second PSS analysis with the tighter tolerance, `reltol =` $1.0e^{-5}$
- Use the *readic* and *skipdc* in the second PSS analysis to read in the final results of the first PSS analysis as the initial conditions.

After the second PSS analysis, you can run small-signal analyses such as PAC.

```
set1 set reltol=1.0E-3
pss1 pss ... writefinal="SomeFile"
set2 set reltol=1.0E-5
pss2 pss ... readic="SomeFile" skipdc=yes
...  pac ... ...
```

Always use a sequence of PSS runs when a you need a tight tolerance PSS solution. If necessary, you can run more than two PSS analyses in the hierarchical process. You choose the tolerance sequence for the continuation. The multiple PSS approach usually produces a better periodic steady-state solution for subsequent small-signal analyses.

The above procedure is often called a continuation on the simulation parameter `reltol`. However, you can use continuation with many other simulation parameters. For example, in order to achieve PSS convergence at a high input power that causes nonconvergence, you might gradually increase the input power at an RF port.

C

Using the psin Component

The procedures described in this appendix are deliberately broad and generic. Requirements for your specific design might dictate procedures slightly different from those described here.

This appendix describes how to use the *psin* component from the `analogLib` library in SpectreRF simulations within the analog design environment.

Independent Resistive Source (psin)

The *psin* component, located in the `analogLib` library, is used in all RF circuits for SpectreRF and Spectre S-parameter simulations.

When you netlist *psin* in the analog design environment using the SpectreS simulator, you can see that *psin* is the *port* component in Spectre simulation. A port is a resistive source that is tied between positive and negative terminals. It is equivalent to a voltage source in series with a resistor, and the reference resistance of the port is the value of the resistor.

psin Capabilities

While the *psin* is generally useful as a stimulus in high-frequency circuits, it also has the following three unique capabilities:

- Defines the ports of the circuit to the S-parameter analysis
- Has an intrinsic noise source that lets the noise analysis directly compute the noise figure of the circuit
- Is the only source for which the amplitude can be specified in terms of power

Terminating the psin

Be aware that when you specify the voltage on a *psin*, Spectre assumes that *the* `psin` *is properly terminated in it's reference resistance.* The specified voltage value is not the voltage on the internal voltage source, which is actually set to twice the value specified on the *psin*. So, if you use a *psin* source to drive an open circuit, the voltage (for *DC, transient*, *AC*, and *PAC* signals) is double its specified value. However, you can alternatively specify the amplitude of the sine wave in the *transient* and PAC analyses as the power in dBm delivered by the `psin` *when terminated with the reference resistance.*

The *psin* component Edit Properties form is shown in Figure C-1 on page 1061and in Figure C-2 on page 1062.

Figure C-1 psin Component Edit Properties Form (see next page also)

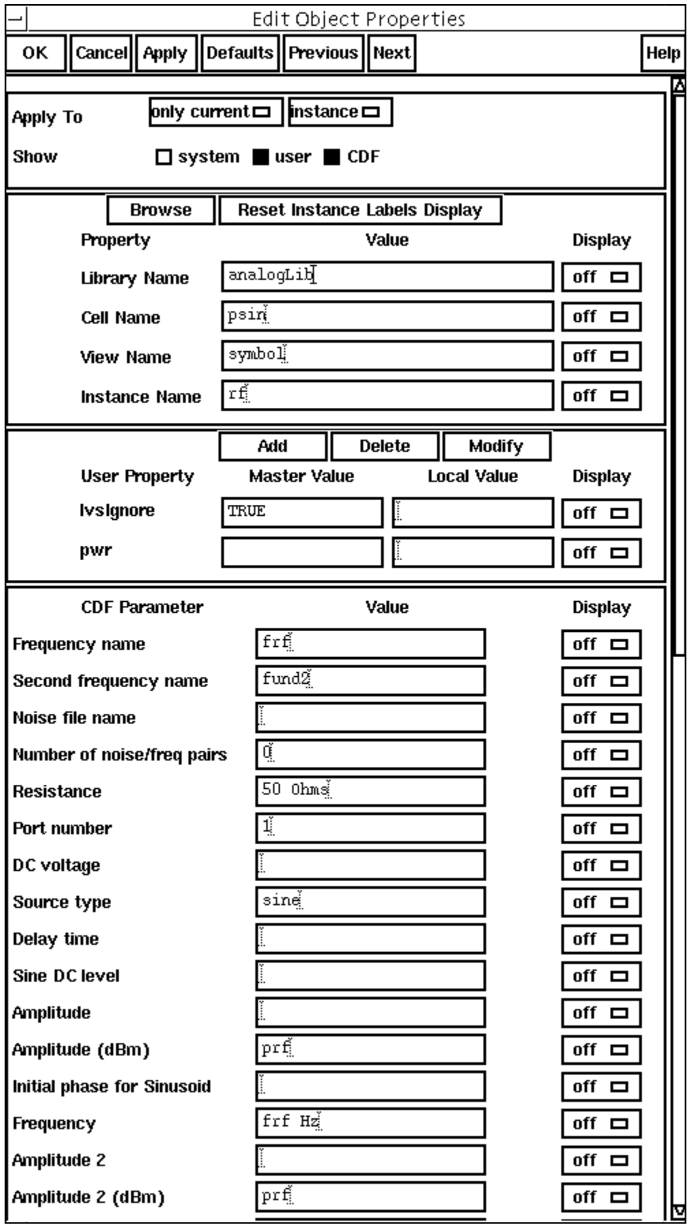
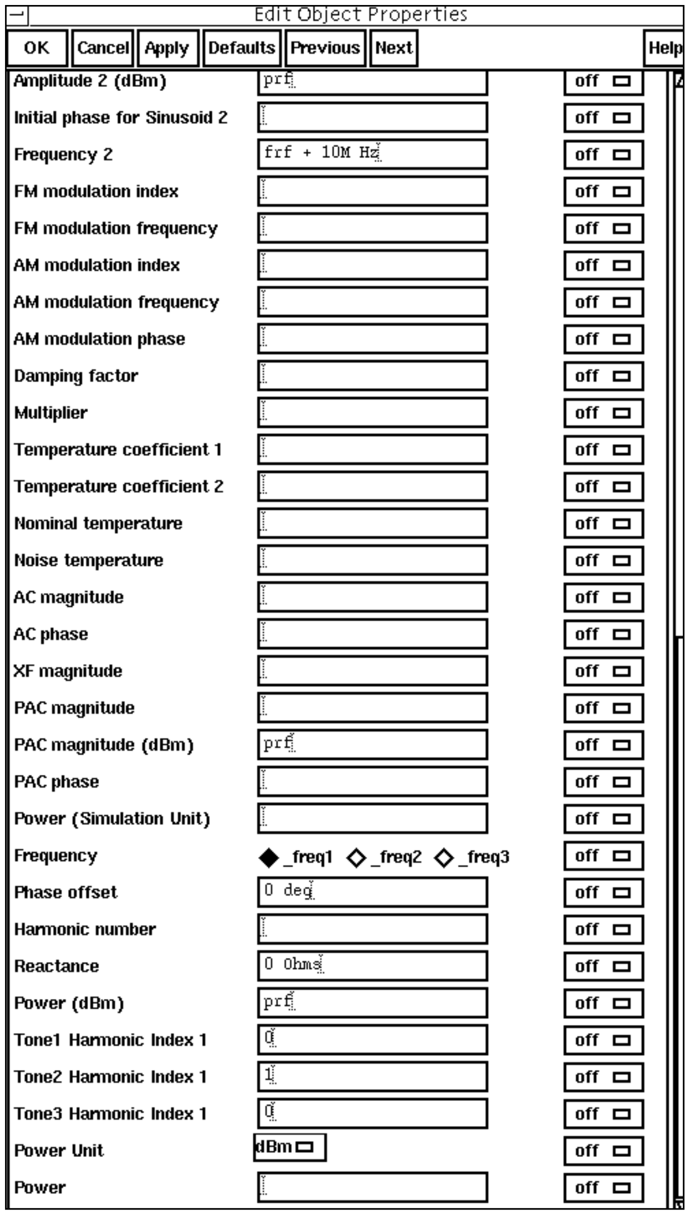


Figure C-2 psin Edit Properties Form, continued



Parameter Types for the psin Component

In this list, *psin* parameters are grouped by type rather than by the order in which they appear on the psin *Edit Properties* form. The *psin* parameter types and their respective parameters are shown below:

Name Parameters

Frequency name

Second frequency name

psin Instance Parameter

DC voltage

General Waveform Parameters

Source type

Delay time

Sinusoidal Waveform Parameters

Sine DC level

Amplitude

Amplitude (dBm)

Frequency

Initial phase for Sinusoid

Amplitude 2

Frequency 2

Phase for sinusoid 2

Amplitude Modulation Parameters

AM modulation frequency

AM modulation phase

AM modulation index

FM Modulation Parameters

FM modulation frequency

FM modulation index

Damping factor

Noise Parameters

Noise file name

Number of noise/freq pairs

Port Parameters

Resistance

Port number

Multiplier

Temperature Effect Parameters

Temperature coefficient 1

Temperature coefficient 2

Nominal temperature (See the following discussion about the effect of temperature parameters on the voltage level.)

Small-Signal Parameters

AC magnitude

AC phase

XF magnitude

PAC magnitude

PAC magnitude (dBm)

PAC phase

Note: The "Additional Notes" section on page 1076 contains information about active parameters in analyses and terminating the psin.

Name Parameters

Frequency Name and *Second frequency name* let you assign names to fundamental tones. After you save the schematic, the names you assign appear in the *Fundamental Tones* list box on the Choosing Analyses form.

psin Instance Parameter

DC voltage

Sets the DC level of the source for DC analysis. The value must be a real number. If you do not specify the DC value, it is assumed to be the *time* =0 value of the waveform. Default: `0` Units: V

Because all small signal analyses (*ac, xf,* and *noise*) use DC analysis results, the *DC voltage* level also affects small-signal analyses. Transient analysis is not affected unless you specify *type*=`dc` or use `dc` as a default for the other waveform types.

General Waveform Parameters

Source type

Lets you specify several different waveform shapes and quickly switch between them by changing the value in the *type* field. The typical source types used in SpectreRF analyses are *dc* and *sine*. Possible source type values are `dc`, `pulse`, or `sine`.

If you set the source to *type* = `dc`, you get only DC values even if you specify a sinusoid later. For *type*=`dc`, the *dc* and *tc* parameters are active and set the DC level for all analyses. In DC analysis, this setting also determines the DC level generated by the source regardless of what type you specify.

With the setting *type*=`sine`, you create a sinusoidal waveform whose parameters you can alter. These parameters are described in more detail in the *Sinusoidal Waveform Parameters* section below.

You cannot specify the parameters for type=`pulse` in the *psin Edit Properties* or *Add Component* forms. Source types *pwl* and *exp* are not supported for the `analogLib` *psin* component.

Delay time

The waveform delay time, or the time that the source stays at the DC level before it starts generating waveforms (assuming the source is set to *sine*). The value must be a real number. Default: `0` Units: `seconds`

Sinusoidal Waveform Parameters

The *psin* component can generate up to two sinusoids simultaneously. They are denoted 1 and 2. You can set the amplitude, frequency and phase for both individually. The amplitude can be set to either a voltage or a power level. You can also specify sinusoidal AM or FM modulation of sinusoid 1.

The first sinusoid is described by the parameters *Amplitude, Amplitude dBm, Frequency, Initial phase for sinusoid, AM or FM modulation terms,* and *Damping factor.*

The second sinusoid is described by the parameters *Amplitude2, Amplitude 2 (dBm), Frequency 2, and Initial phase for sinusoid 2*.

Note: The second sinusoid cannot be modulated.

Sine DC level

Sets the DC level for sinusoidal waveforms in transient analyses. This parameter is used when the sinusoid has a different average level than the one specified for the `DC` analyses. If not specified, the average value of the sinusoid is the same as that of the DC level of the source. The value must be a real number. Default: `dc` Units: `V`

Amplitude

Peak amplitude of the first sinusoidal waveform that you generate. The value must be a real number. Default: `1` Units: `V`

Remember, when you specify the voltage on a *psin*, you are specifying the voltage *when the psin is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *psin*.

Amplitude (dBm)

Amplitude of the first sinusoidal waveform when specified in dBm (alternative to Amplitude). The value must be a real number. Units: `dBm`

Caution

Set either Amplitude or Amplitude (dBm). Do not set them both! If you specify both Amplitude (in peak volts) and Amplitude (dBm) in the same source, SpectreRF simulation does not give you any errors or warning messages. It uses Amplitude (in peak volts) and ignores the Amplitude (dBm) field. If you specify Amplitude, verify that the Amplitude (dBm) field is empty, and vice-versa.

Frequency

The frequency of the first sinusoidal waveform (carrier frequency). You typically use unmodulated signals in SpectreRF analyses. The value must be a real number. Default: `0` Units: `Hz`

Phase for Sinusoid

The phase at the specified delay time. The sinusoidal waveform might start before the given delay time in order to achieve a specified phase and still remain continuous. For example, if you want to generate a cosine wave, set this parameter to 90^o. The value must be a real number. Default: `0` Units: `degrees`

Amplitude Modulation Parameters

The amplitude modulation (double sideband large carrier, or DSB-LC) is defined as

$$v_{AM}(t) = A (1 + m \sin(2\pi f_m t + \phi)) \sin(2\pi f_c t)$$

where

- A is the carrier amplitude (amplitude of sinusoid)
- f_m is the AM modulation frequency
- ϕ is the AM modulation phase
- m is the AM modulation index
- $\sin(2\pi f_c t)$ is the carrier signal

The amplitude modulation parameters affect *only* the first sinusoid generated by *psin*. They have no effect on the second sinusoid.

AM modulation frequency

AM modulation frequency for the sinusoidal waveform (f_m in the previous equation). The value must be a real number. Default: `0` Units: `Hz`

AM modulation phase

AM phase of modulation for the sinusoidal waveform (ϕ in the previous equation). The value must be a real number. Default: `0` Units: `degrees`

AM modulation index

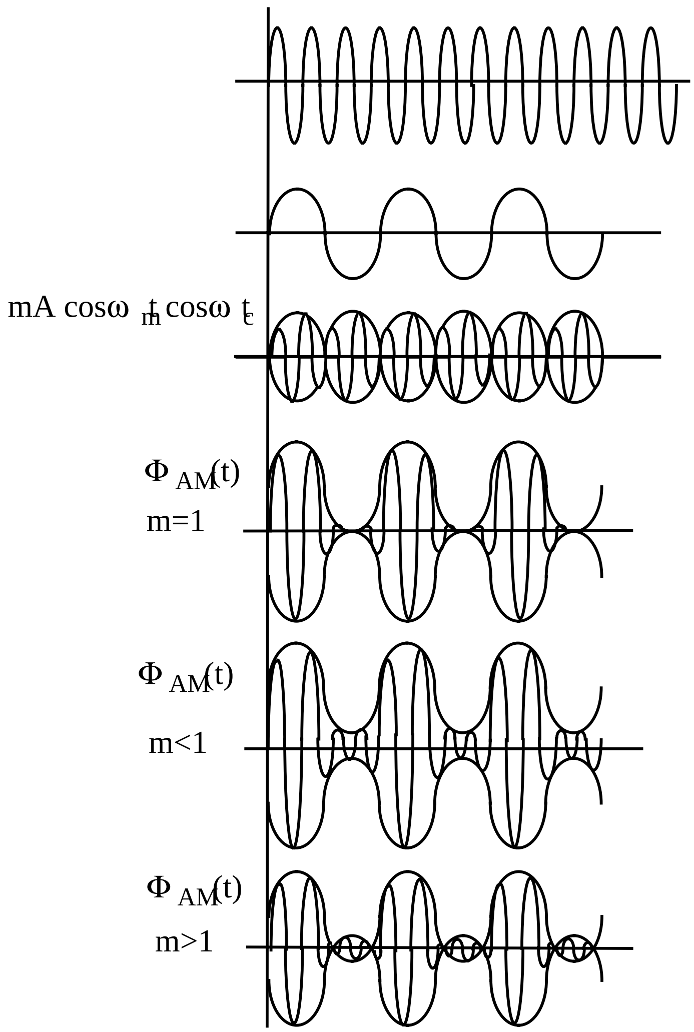
AM index of modulation for the sinusoidal waveform (*m* in the previous AM equation). The AM modulation index is a dimensionless scale factor used to control the ratio of the sidebands to the carrier.

```
m = (peak DSB-SC amplitude)/(peak carrier amplitude)
```

The value must be a real number. Default: `0`

The following figure shows the effect of varying modulation indexes for the following three cases: $m < 1$, $m = 1$, and $m > 1$. f_c is the carrier frequency, and f_m is the modulation frequency.

Figure C-3 Amplitude Modulation: Effects of Varying Modulation Indexes



FM Modulation Parameters

The frequency modulation for the sinusoidal case is defined as

$v_{FM}(t) = A \sin(2\pi f_c t + \beta \sin(2\pi f_m t) + \phi)$

where

- A is the amplitude of the sinusoid
- β is the FM modulation index

- $\sin(2\pi f_m t)$ is the modulation signal
- f_c is the carrier frequency
- ϕ is the phase of the sinusoid

The frequency modulation parameters affect *only* the first frequency generated by *psin*. They have no effect on the second frequency.

FM modulation frequency

FM modulation frequency for the sinusoidal waveform (f_m in the equation above). The value must be a real number. Default: `0` Units: `Hz`

FM modulation index

FM index of modulation for the sinusoidal waveform, the ratio of peak frequency deviation divided by the center frequency (β in the above equations).

$\beta = \Delta f / f_m$

The value must be a real number. Default: `0`

Damping factor

Damping factor for the sinusoidal waveform. *Damping factor* specifies the time it takes to go from the envelope (full amplitude) at *time*=0 to 63 percent of the full amplitude. For example, consider the following damped sinusoid:

$$v(t) = A\, e^{-\sigma t} \sin(2\pi f t + \phi)$$

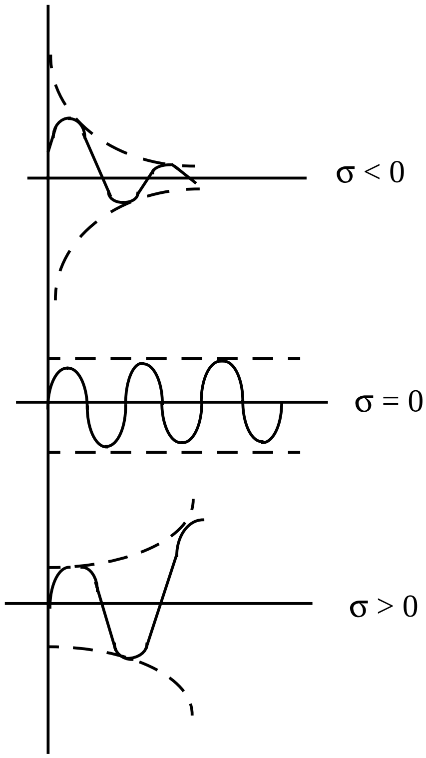
where σ= damping factor.

- If $\sigma = 0$, the waveform is a pure sinusoid (steady state).
- If $\sigma < 0$, the waveform exhibits decaying oscillations.
- If $\sigma > 0$, the waveform exhibits growing oscillations.

It takes 5s to diminish to 1 percent of the peak amplitude. The value must be a real number. Default: `0`. Units: `1/seconds`

The following figure shows the effect of *Damping factor* on the first sinusoid for three values of σ.

Figure C-4 Effect of *Damping Factor* on the First Sinusoid



Amplitude 2

Peak amplitude of the second sinusoidal waveform. The value must be a real number.
Default: `1` Units: `V`

Remember, when you specify the voltage on a *psin*, you are specifying the voltage *when the psin is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *psin*.

Amplitude 2 (dBm)

Amplitude of the second sinusoidal waveform in dBm (alternative to *Amplitude 2*). The value specified is the power delivered into a matched load. The value must be a real number. Units: `dBm`

Caution

Set either Amplitude2 or Amplitude2 (dBm). Do not set them both!

Frequency 2

Frequency of the second sinusoidal waveform. The value must be a real number. Default: `0` Units: `Hz`

Phase for Sinusoid 2

The phase at the specified *Delay time* for the second sinusoid. The sinusoidal waveform might start before the given *Delay time* in order to achieve specified phase while still remaining continuous. The value must be a real number. Default: `0` Units: `degrees`

Noise Parameters

Noise parameters include *Noise file name, Number of Noise Frequency Pairs, and Noise temperature.*

Noise file name

Name of the file containing the excess spot noise data in the form of frequency-noise pairs. The value must be a string. Default: no value

Number of noise/freq. pairs

You must specify the number of noise-frequency pairs that exist in your noise file in the form. In your file, list the noise-frequency pairs as one pair per line with a space or tab between the frequency and noise values. The values are given as a vector of real number pairs, where noise is given in V^2/Hz, and frequency is given in Hz.

Specific to Analog Design Environment netlisting: If you have more noise-frequency pairs than the number you specify in the *Number of noise frequency pairs* field, the number of noise-frequency pairs used is the number you specify in the field. Any additional noise-frequency pairs in the noise file are ignored. If the number of noise-frequency pairs in the noise file is smaller than the number you specified in the *number of noise frequency pairs* field, the analysis is stopped.

Noise temperature

The *Noise temperature* of the *psin*. If not specified, the *Noise temperature* is assumed to be the actual temperature of the *psin*. When you compute the noise figure of a circuit driven at its input by a *psin*, set the noise temperature of the *psin* (Spectre parameter *noisetemp*) to 16.85C (290K). This setting matches the standard IEEE definition of noise figure. In addition, disable all other sources of noise in the *psin*, such as the Spectre parameters *noisefile* and *noisevec.* If you want a noiseless *psin*, set the *noise temperature* to absolute zero or below, and do not specify a noise file or noise vector. Default: Units: ^{o}C.

Port Parameters

Port parameters include *Resistance, Port Number, and Multiplier*.

Resistance

The reference resistance of the system. The value must be a real number, but not 0. Default: `50` Units: Ω

Port number

The number of the port. The value must be a nonzero integer. Each *psin* in a schematic must have a different port number. The *Port number* is not automatically indexed when you place each *psin* on your schematic.

Multiplier

The multiplicity factor. The value must be an integer number greater than zero. This number lets you specify a number of *psin*s in parallel. For example, if you set *Resistance* to 50 and *Multiplier* to 2, you specify two *psin* ports in parallel, each with an effective reference resistance of 25 Ω. Default: `1`

Temperature Effect Parameters

How Temperature Parameters Affect the Voltage Level (Background Information)

The value of the DC voltage can vary as a function of the temperature if you specify *tc1* and *tc2*. The variation is given by

$$V_{DC}(T) = dc * [1 + tc1 * (T - tnom) + tc2 * (T - tnom)^2]$$

where *T* is the analysis temperature specified in the analysis options, *tnom* is the nominal temperature specified in the *Choosing Analyses* form, and *dc* is the DC voltage.

If the analysis temperature equals the nominal temperature, the result is the voltage amplitude that you specified, V_{DC}`(T)= dc`.

If the nominal and analysis temperatures differ, the voltage amplitude is given by

```
VDC(T) = dc * [1 + tc1 * (T – tnom) + tc2 * (T – tnom)2]
```

where *T* is the analysis temperature you specify in the analysis options and *tnom* is the nominal temperature. *tc1* and *tc2* are the linear and quadratic temperature coefficients.

For example, if the nominal temperature is 27^{o}C and the analysis temperature is 25^{o}C, there is a 2^{o} difference between the nominal and analysis temperature. The voltage amplitude is

$V_{DC}(T) = dc * [1 + tc1 * (-2) + tc2 * (-2)^2]$

Temperature effect parameters include *Temperature coefficient 1* (the linear temperature coefficient), *Temperature coefficient 2* (the quadratic temperature coefficient), and *Nominal temperature*.

Temperature coefficient 1

First order (linear) temperature coefficient of the *DC voltage* (`tc1`). The value must be a real number. Default: `0` Units: ^{o} C^{-1}

Temperature coefficient 2

Second order (quadratic) temperature coefficient of the *DC voltage* (`tc2`). The value must be a real number. Default: `0` Units: ^{o} C^{-2}

Nominal temperature

The nominal temperature for *DC voltage* (*tnom*). The value must be a real number. Default: Set by options specifications Units: `C`

Small-Signal Parameters

The small signal parameters are *AC magnitude, AC phase, XF magnitude, PAC magnitude, PAC magnitude (dBm),* and *PAC phase.*

Remember, when you specify the voltage on a *psin*, you are specifying the voltage *when the `psin` is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *psin*. The same is true for the values for the *transient*, *AC*, and *PAC* signals. However, the amplitude of the sine wave in the *PAC* and *transient* analysis can alternatively be specified as the power in dBm delivered by the *psin when terminated with the reference resistance*

AC magnitude

The peak small-signal voltage. The value must be a real number. Default: `0` Units: `V`

AC phase

The small-signal phase. The value must be a real number. Default: `0` Units: `degrees`

Typically, only one source in the circuit has *AC Magnitude* set to a value other than zero, and usually it has an *AC magnitude*=1 and *AC phase*=0. However, there are situations where more than one source has a nonzero *AC magnitude*. For example, applying a differential small-signal input could be done with two sources with the *AC magnitude*s set to 0.5 and the *AC phase*s set to 0 and 180.

XF magnitude

The transfer function analysis magnitude. Use *XF magnitude* to compensate for gain or loss in the test fixture. The value must be a real number. Default: `1` Units: `V/V`

PAC magnitude

The peak periodic *AC* analysis magnitude. Setting this value to unity is a convenient way of computing the transfer function from this source to the output. The value must be a real number. Default: `0` Units `V`

PAC magnitude (dBm)

The periodic *AC* analysis magnitude in dBm (alternative to *PAC magnitude*). The value must be a real number. Units: `dBm`

Caution

Set either PAC magnitude or PAC magnitude (dBm), but do not set them both!

PAC phase

The periodic AC analysis phase. The value must be a real number. Default: `0` Units: `degrees`

Typically, only one source in the circuit has a *PAC magnitude* set to a value other than zero, and usually it has a *PAC magnitude*=1 and *PAC phase*=0. However, there are situations where more than one source has a nonzero *PAC magnitude*. For example, applying a differential small-signal input could be done with two sources with the *PAC magnitudes* set to 0.5 and the *PAC phases* set to 0 and 180."

You do not specify the *PAC* frequency in the `psin` *Edit Object Properties* form. Instead, you set the *PAC frequency* in the *PAC Choosing Analysis* form. For example, when making an IP3 measurement, you set the PAC frequency to a variable value in the *Choosing Analysis* Form. Then, you enter the same variable in the *PAC Amplitude* (or *PAC Amplitude dBm*) field of the psin *Edit Object Properties* form.

Additional Notes

Active Parameters in Analyses

In DC analyses, the only active parameters are *dc, m*, and the temperature coefficient parameters.

In AC analyses, the only active parameters are *m, mag* and *phase*.

In Transient analyses, all parameters are active except the small-signal parameters and the noise parameters.

In PAC analyses, the only active parameters are *m*, *PAC magnitude* (amplitude or dBm), and *PAC phase*.

XF magnitude is active in *XF* and *PXF* analyses only.

D

The RF Library

The Contents of the rfLib

The elements contained in the RF Library, *rfLib,* are organized into the following categories:

Categories in *rfLib*	**Description**
Original_RFAHDL_lib	The original RFAHDL library
bot_upBB	Bottom-Up Baseband elements
measurement	Measurement elements
testbenches	Testbench elements
top_dwnBB	Top-Down Baseband elements
top_dwnPB	Top-Down Passband elements
Everything	List of all elements in *rfLib*
Uncategorized	Files created by testbenches elements

The *rfLib* contains elements to support the design of both RF circuits and RF systems.

Elements for Transistor-Level RF Circuit Design

Elements in the category *Original_RFAHDL_lib* exist specifically to support transistor-level RF circuit designers.

The *Original_RFAHDL_lib* category contains the RF AHDL library first released in IC443. The elements in this library are detailed behavioral models but they are not baseband equivalent. Models in *Original_RFAHDL_lib* are documented in section "Models for Transistor-Level RF Circuit Design" on page 1079.

Elements for Top-Down System-Level RF Design

Elements in the categories *top_dwnBB* and *top_dwnPB* exist to support system-level RF designers.

- The *top_dwnBB* category contains the Top-Down Baseband models of common architectural function blocks. The default view of these models is the baseband view (called *veriloga)* but most models in this category also have a differential passband view (called *veriloga_PB)*. The only exceptions are the BB_loss and VGA_BB models. The BB_loss model is meant only for baseband analysis. The VGA_BB is an engineering release of a variable gain amplifier. It is not documented or supported in IC446 and is subject to change in future releases.
- The *top_dwnPB* category contains the Top-Down Passband models—single-ended passband versions of the baseband models.

  The elements in *top_dwnBB* capture only information that can be modeled at baseband.The elements in *top_dwnPB* are passband versions of the baseband models which allow the RF system designer to switch between baseband and passband views during the design process.

The *top_dwnBB* and *top_dwnPB* models are documented in "Models for Top-Down RF System Design" on page 1116.

Elements for Use With Both Design Methodologies

Elements in the *measurement* and *testbenches* categories are not part of either RF architecture. They can be used by both RF system designers and RF circuit designers.

- The *measurement* category contains models of the instrumentation blocks and baseband signal generators used to facilitate measurements and diagnostics.
- The *testbenches* category contains the test circuits used to define model specifications. Where possible, the element names are in terms of standard RF measurements.

The models in the *measurement* category are documented in "Measurement Elements" on page 1097. The circuits in the *testbenches* category are documented in Chapter 12, "Methods for Top-Down RF System Design."

Elements for Bottom Up Transmitter Design

The *bot_upBB* category contains the Bottom-Up Baseband models—behavioral baseband J-models for transmitters. These J-models are documented in "Bottom-Up Design Elements" on page 1115.

Models for Transistor-Level RF Circuit Design

If you are a transistor-level RF circuit designer who creates top-down designs, you must model functional RF blocks at the behavioral level. You can use the high-level models from *Original_RFAHDL_lib* as building-blocks for complex RF systems or executable specifications. You can also use the models from *bot_upBB*. They are described in "Bottom-Up Design Elements" on page 1115.

When distributed in source code format, these models also help you write libraries that reflect your own design needs.

The Verilog-A® language lets you use the new RF library models in Spectre RF without an explicit equivalent circuit. This approach uses the high-level description language capability of SpectreRF to create high-level models that capture the essential information of RF functional blocks. You can then insert these models into regular RF circuits for simulation. The essential parameters are translated into coefficients and equations that describe the relations between the voltages and currents at the connecting nodes.

Original_RFAHDL_lib Elements

You can characterize most RF circuits with three sets of parameters: Linear Parameters, Nonlinear Parameters, and Noise Parameters.

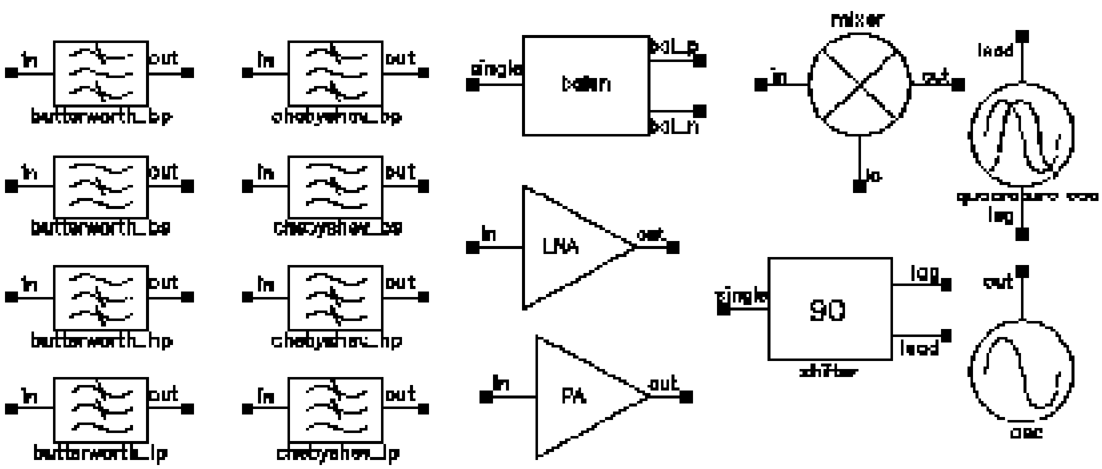
The RF AHDL library, *Original_RFAHDL_lib,* contains models for top-down design. It contains all of the elements of the original *rfLib*. The models in *Original_RFAHDL_lib* are designed specifically to work with SpectreRF as these models do not have hidden state.

The *Original_RFAHDL_lib* contains the seven elements, discussed in the sections that follow.

- `Balun`
- `Filters`
- `Low Noise Amplifier`
- `Mixer`
- `Power Amplifier`
- `Oscillator`
- `Quadrature Signal Generator`
- `Phase Shifter`

Figure D-1 shows the symbols used for the components that are available in the in *Original_RFAHDL_lib*.

Figure D-1 RF Library Components: Original_RFAHDL_lib



Balun

The balun (balancing transformer) is used in circuits that require single/differential signal transformation. Although a passive network (including the transformer) is used to achieve balun, this implementation employs a three-port network. There are three ports (or nodes), because the reference nodes are always at the global ground: *single*, *bal_p*, and *bal_n*.

When the ports are numbered as *single*(1), *bal_p*(2), and *bal_n*(3), the S-parameter for the three-port network is

$$S = \begin{bmatrix} 0 & t & -t \\ t & 0 & 0 \\ -t & 0 & 0 \end{bmatrix}$$

where

$$t = \frac{10^{-loss/10}}{\sqrt{2}}$$

when loss is specified in dB.

This module can also be used in common mode cancellation applications.

The module is declared as follows:

```
module balun(single, bal_p, bal_n);
  inout single, bal_p, bal_n;
  electrical single, bal_p, bal_n;
  parameter real rin = 50 from (0:inf);
  parameter real rout = 50 from (0:inf);
  parameter real loss = 0 from (-inf:0];
```

Parameters include the input impedance (for single end), the output impedance (for balanced end to ground), and the insertion loss (from single end to balanced end and from balanced end to single end). The parameters are

`rin`	Input impedance [Ω]
`rout`	Output impedance [Ω]
`loss`	Insertion loss [dB]

Filters

Filter properties are specified in the frequency domain, but it is not easy for the SpectreRF circuit simulator to process frequency-domain data. SpectreRF simulation requires a large signal, time-domain model to simulate filter behavior.

As part of the RF AHDL library, filters are implemented using a network synthesis technique which consists of the following two steps:

- First calculate the normalized low-pass filter prototype, which consists of serial inductors and parallel capacitors
- Then perform frequency transformation and scaling to synthesize the frequency responses of the filter type

The synthesized model contains many inductors and capacitors. They are implemented using the integral and differential functions of the Verilog-A language. Insertion loss is added using the S-parameter network technique. This network essentially dampens the signal flow by the specified insertion loss value.

Low-pass, high-pass, bandpass, and bandstop filters are implemented, and each of these can be a Butterworth or Chebyshev type filter (for a total of eight filters) as follows:

`butterworth_bp`   `chebyshev_bp`

`butterworth_bs`	`chebyshev_bs`
`butterworth_hp`	`chebyshev_hp`
`butterworth_lp`	`chebyshev_lp`

In the current implementation of the Verilog-A language, the order and internal states of the filter cannot be dynamically allocated. You must use the `'define` derivative in the source code of the Verilog-A file to specify the order. Use S-parameters to test the filters because S-parameters capture the input/output impedance matching.

For example, the Butterworth bandpass filter, `butterworth_bp`, has the following module declaration:

```
module butterworth_bp(t1, t2);
   inout t1, t2;
   electrical in, out;
   parameter real r1 = 50 from (0:inf);
   parameter real r2 = 50 from (0:inf);
   parameter real f0 = 1e9 from (0:inf);
   parameter real bw = 0.10 from (0:0.5);
   parameter real fc = 1e9 from (0:inf);
   parameter real loss = 0 from [0:inf);
```

where `t1` and `t2` are the input and output nodes, respectively.

The parameters are

`r1`	Input impedance [Ω]
`r2`	Output impedance [Ω]
`fc`	Corner frequency (3 dB point) for low-pass and high-pass filter [Hz]
`f0`	Center frequency for bandpass or bandstop filter [Hz]
`bw`	Relative frequency for bandpass or bandstop filter [Hz]
`loss`	Insertion loss [dB]

Figure D-2 is the simple schematic that tests the filter. Two ports are used to obtain the S-parameters.

Figure D-2 Schematic for Testing the Filter Model

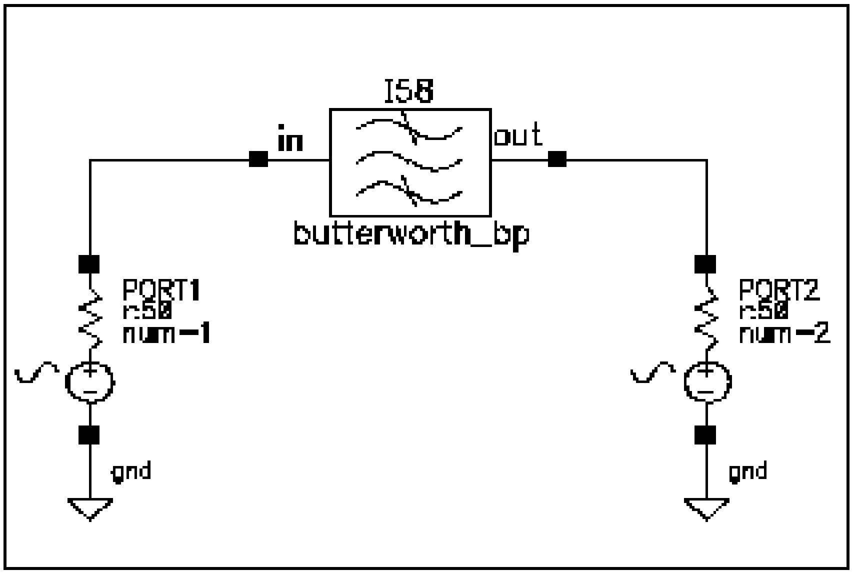
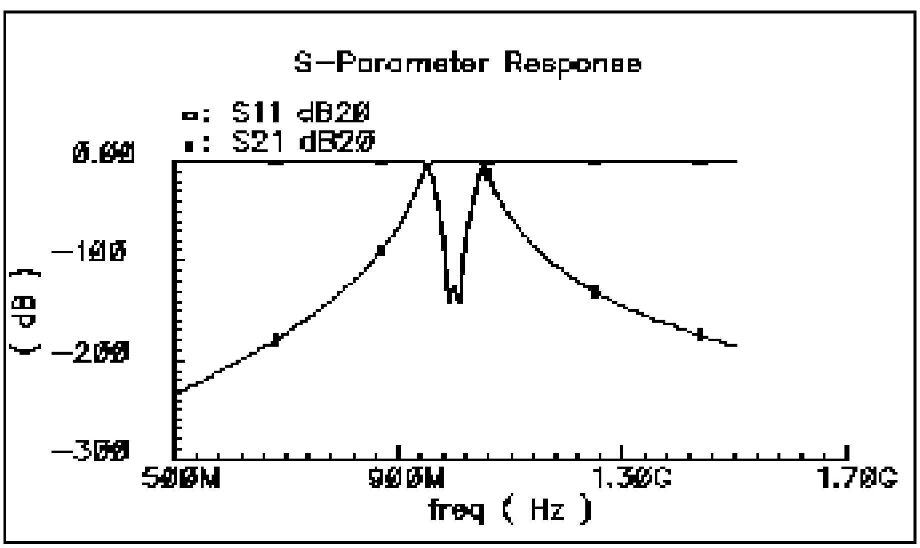


Figure D-3 shows the calculated S-parameters of this Butterworth bandpass filter, which has a center frequency of 1 GHz and a relative bandwidth of 10 percent. The order of this specific filter is 10.

Figure D-3 S-Parameters of a Butterworth Filter



Low-Noise Amplifier

Low-noise amplifiers (LNAs) are commonly used in the receiver design to amplify the signal with a low noise figure. A typical LNA has the following three sets of parameters:

- Linear model
- Nonlinear model
- Noise model.

The module is declared as follows:

```
module lna(in, out);
    inout in, out;
    electrical in, out;
    parameter real nf = 2 from [0:inf);
    parameter real ip3 = -10;
    parameter real gain = 15 from [0:inf);
    parameter real isolation = 200 from (0:inf);
    parameter real rin = 50 from (0:inf);
    parameter real cin = 0 from [0:100];
    parameter real rout = 50 from (0:inf);
    parameter real cout = 0 from [0:100];
    parameter real gammain = -150 from (-inf:0];

    parameter real mismatch = 1 from [-1:1] exclude (-1:1);
    parameter real gammaout=-150 from (-inf:0];
```

The parameters are

`nf`	Noise figure [dB]
`ip3`	Input referenced IP3 [dBm]
`gain`	S21 (power gain) [dB]
`isolation`	S12 [dB]
`rin`	Reference impedance of the input port [Ω]
`rout`	Reference impedance of the output port [Ω]
`gammain`	Input return loss [dB]
`mismatch`	Mismatch sign of input. 1: input impedance > reference impedance -1: otherwise

`gammaout`	Output return loss [dB]
`cin`	Parasitic input capacitance [pF]
`cout`	Parasitic parallel output capacitance [pF]

Internally, a set of linear equations is constructed to satisfy the S-parameters. Furthermore, nonlinearity, expressed by a third-order polynomial function, is added to the gain (or S21) to describe the IP3. Excess white noise is added at the input port to describe the noise figure.

IP3 is the measure of the corruption of signals due to the third-order intermodulation of two nearby tones as shown in Figure D-4. You measure this parameter using a two-tone test. Avoid the measurement of IP3 by a single tone test.

Figure D-4 Intermodulation of Two Nearby Signals

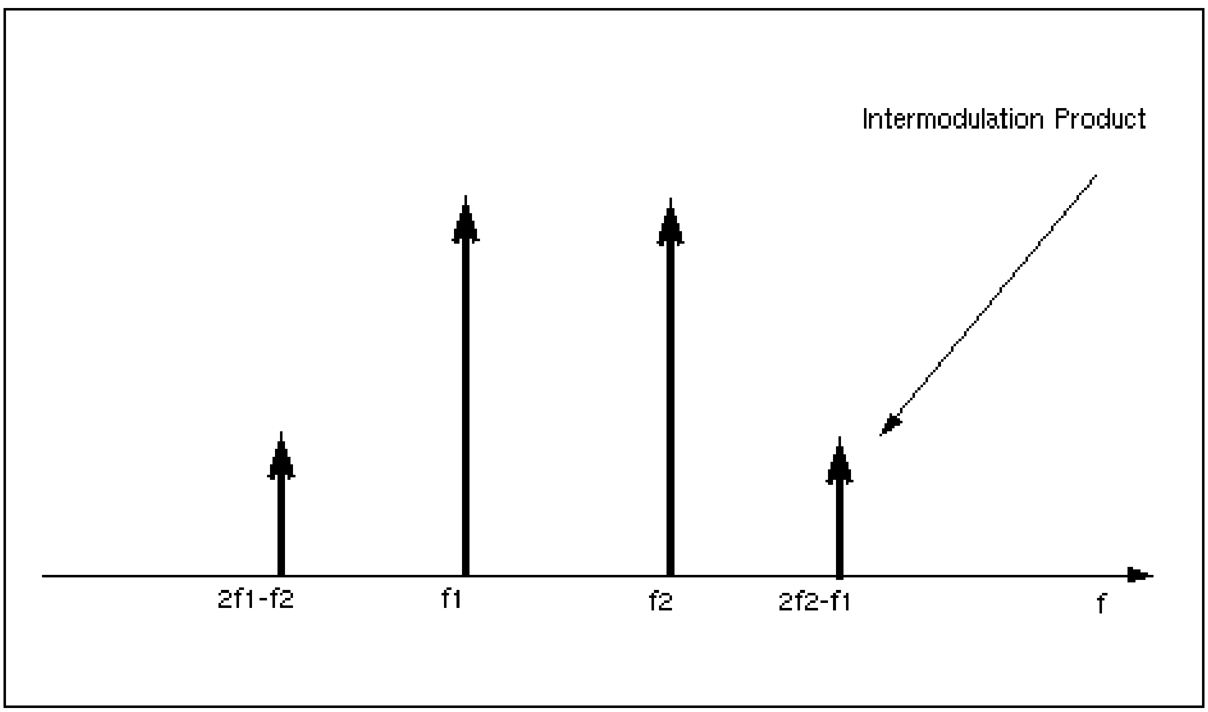
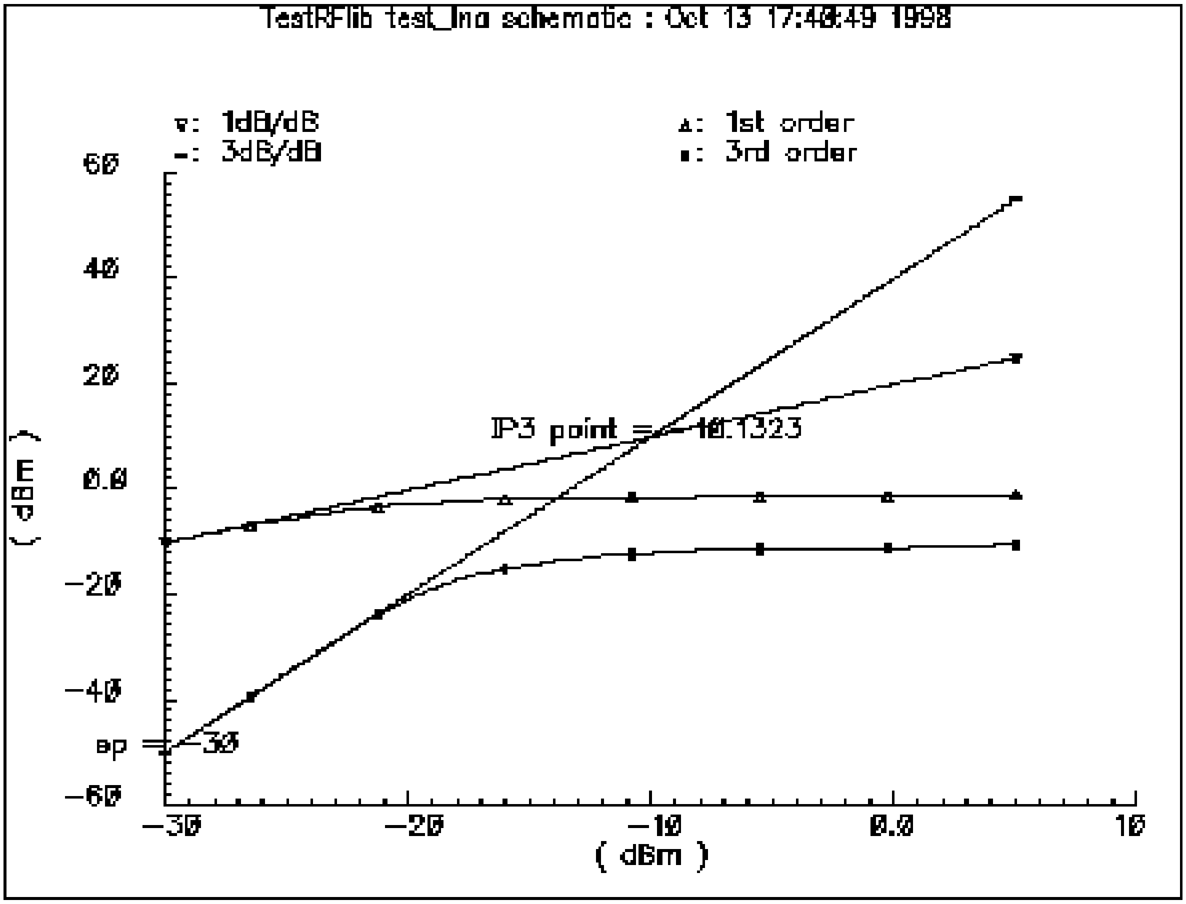


Figure D-5 shows the captured IP3 when the requested value of IP3 is –10dBm.

Figure D-5 IP3 from SpectreRF Simulation



Mixer

Mixers are very important for frequency translation in RF circuits. A typical mixer has the following three sets of parameters:

- Time-varying linear model
- Nonlinear model
- Noise model

This RF library model describes the typical behavior of integrated mixers. The LO switches the input signal on and off. Input LO power beyond the specified limit is effectively clipped off.

The module is declared as follows:

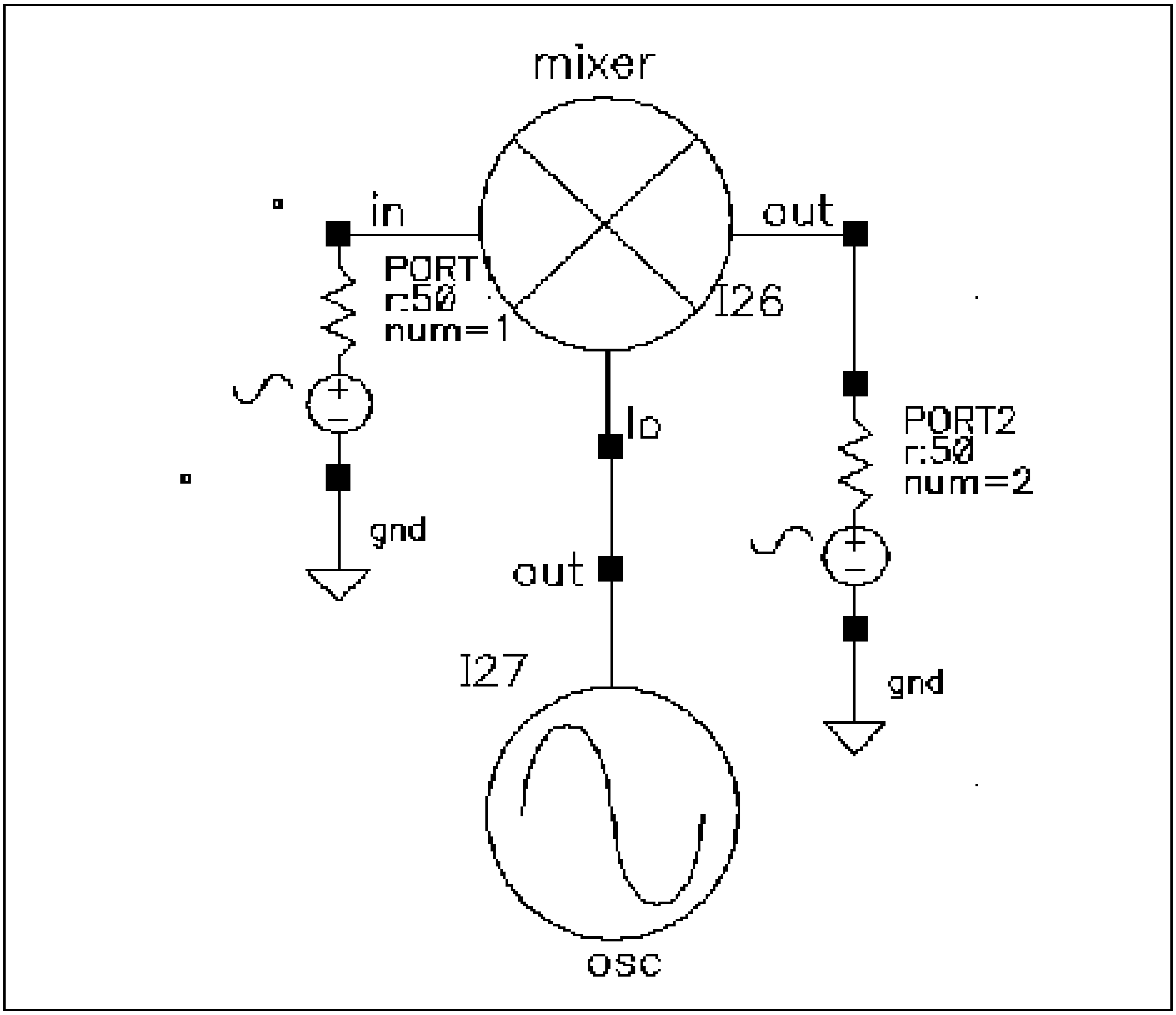
```
module mixer(in, lo, out);
   electrical in, lo, out;
   parameter real gain = 10 from [-50:50];
   parameter real plo = 10 from [-100:100];
   parameter real rin = 50 from (0:inf);
   parameter real rout = 200 from (0:inf);
   parameter real rlo = 50 from (0:inf);
   parameter real ip2 = 5;
   parameter real ip3 = 5;
   parameter real nf = 2 from [0:inf);
   parameter real isolation_LO2IN = 20 from (0:inf);
   parameter real isolation_LO2OUT = 20 from (0:inf);
   parameter real isolation_IN2OUT = 20 from (0:inf);
```

The parameters are

`gain`	Gain from IN to OUT [dB]
`plo`	Power of the LO input [dBm]
`rin`	Input impedance for IN [Ω]
`rout`	Output impedance for OUT [Ω]
`rlo`	Input impedance for LO [Ω]
`ip2`	Input referenced IP2 [dBm]
`ip3`	Input referenced IP3 [dBm]
`nf`	Noise figure (SSB) [dB]
`isolation_LO2IN`	Isolation from LO to IN [dB]
`isolation_LO2OUT`	Isolation from LO to OUT [dB]
`isolation_IN2OUT`	Isolation from IN to OUT [dB]

Figure D-6 is the simple schematic that tests the mixer.

Figure D-6 Schematic for Testing the Mixer Model

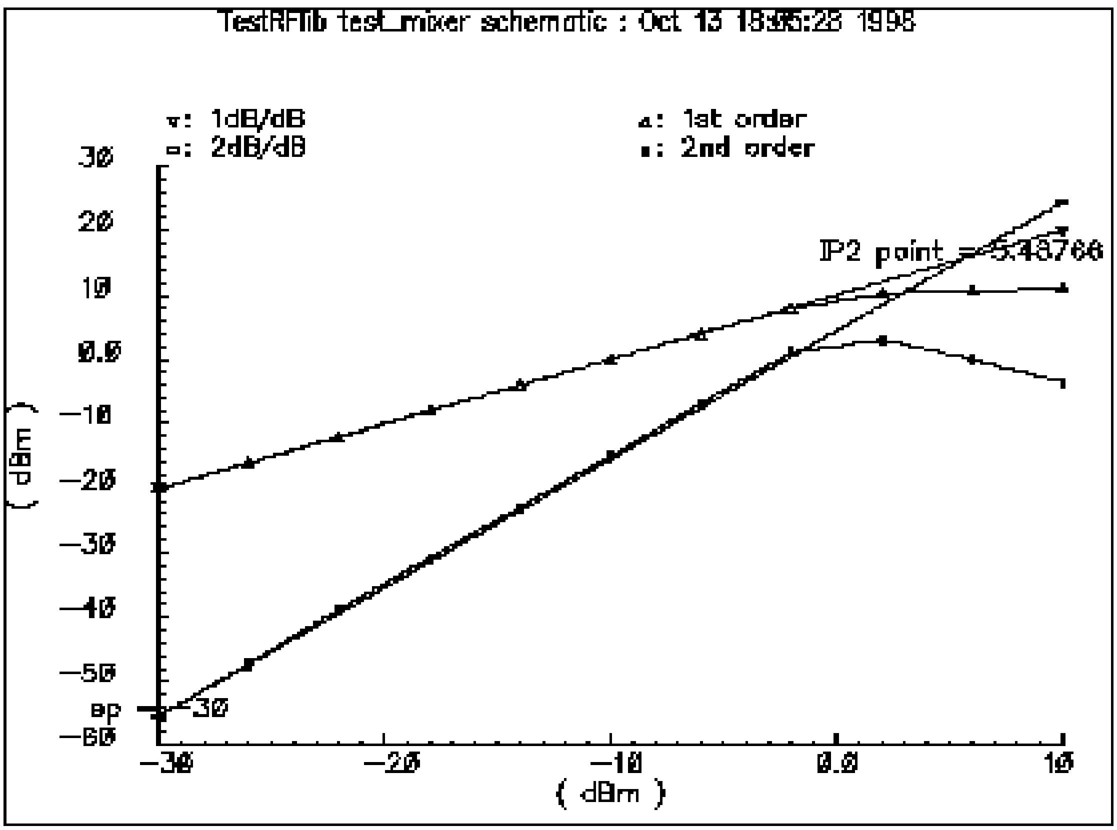


`plo` is the maximum power of the fundamental frequency of the local oscillator that can be used in the mixing process. Therefore, the gain, defined as the output power of the mixed product versus the input power of the RF signal, depends on the power level of the LO. The gain levels off, however, to the specified maximum value as the LO signal becomes larger.

You can measure both IP3 and IP2 with SpectreRF. You must select frequencies carefully when you measure IP3 to measure harmonic distortion (HD) and IP2. Testing IP3 requires two tones to measure the intermodulation distortion (IMD), while testing IP2 requires only one tone.

Assume the RF input frequencies are f_1 and f_2, and the LO frequency is f_{lo}. If the input power level at f_1 equals that at f_2, the IP3 is the intercept point of the extrapolated line of output power at frequency $|f_{lo}\text{-}(2f_2\text{-}f_1)|$ versus the extrapolated line of the linear output signal at $|f_{lo}\text{-}f_1|$. Input-referenced IP3, therefore, can be read as the X-axis value at the intercept point. The IP2, for the purpose of measuring the half-IF effects, is defined as the intercept point of the extrapolated line of output power at frequency $|2(f_{lo}\text{-}f_1)|$ versus the linear output signal. Figure D-7 shows that the intercept point of the 1 dB/dB and 2 dB/dB lines is at the X-axis reading of 4.78 dBm, while the requested IP2 value is 5 dBm. The order of the intercept point is based only on the order of the RF signals. The order of LO signal is not counted in the definition of the intercept point. In the implementation of this model, the orders of LO for IP3 and IP2 are 1 and 2 respectively.

Figure D-7 IP2 Measurement



Internally, a set of equations is built to satisfy a three-port S-parameter. A third-order polynomial describes the nonlinearity of IP3. The LO signal is further multiplied by itself to derive the second-order harmonic, which is then used to produce the IP2 effect. Excessive white noise is added in the RF input port to satisfy the noise figure. Remember, however, that the noise figure is a single-sideband. If the noise at the image frequency is not filtered out, the measured noise figure is 3dB larger than the SSB noise figure.

Power Amplifier

Power amplifiers (PAs) are used in RF transmitters to achieve output of a higher power level. The PA model differs from the LNA model in that it has greater power delivery capabilities with less stress on matching capabilities.

The Verilog-A module is declared as follows:

```
module pa(in, out);
   inout in, out;
   electrical in, out;
   parameter real nf = 2 from [0:inf);
   parameter real gain = 20 from [0:inf);
   parameter real rin = 50 from (0:inf);
   parameter real rout = 50 from (0:inf);
   parameter real p1db = 30;
   parameter real psat = 35;
   parameter real ip2 = 40;
```

The parameters are

`nf`	Noise figure [dB]
`gain`	S21 [dB]
`rin`	Input impedance [Ω]
`rout`	Output impedance [Ω]
`psat`	Maximum output power [dBm]
`p1db`	Output-referenced 1dB compression [dBm]
`ip2`	Input-referenced IP2 [dBm]

The power amplifier model has the following three parts:

- the linear model
- the nonlinear model
- the noise model

Internally, for simplicity, the reverse isolation is assumed to be ideal. A set of linear equations is constructed to satisfy these S-parameters. Nonlinear effects are added to the gain to describe the nonlinearity. The output power of the power amplifier compresses to 1 dB less than the output of an ideal linear amplifier at the 1 dB compression point. Further increase of the input power makes the output approach the saturation power only at the fundamental operating frequency. IP2 describes the second order effects of the amplifier, so use only one tone in the test. Excess white noise is added at the input port to describe the noise figure.

The implementation of `Psat` assumes a pure sinusoidal waveform. To maintain a restrained output power, the output waveform is clipped from a sinusoidal to a square wave form. Figure D-8 shows the input and output waveforms of the power amplifier. Because of the output waveform clipping, the input sinusoidal wave should have a DC component of zero.

Figure D-8 Input and Output Waveforms of the Power Amplifier

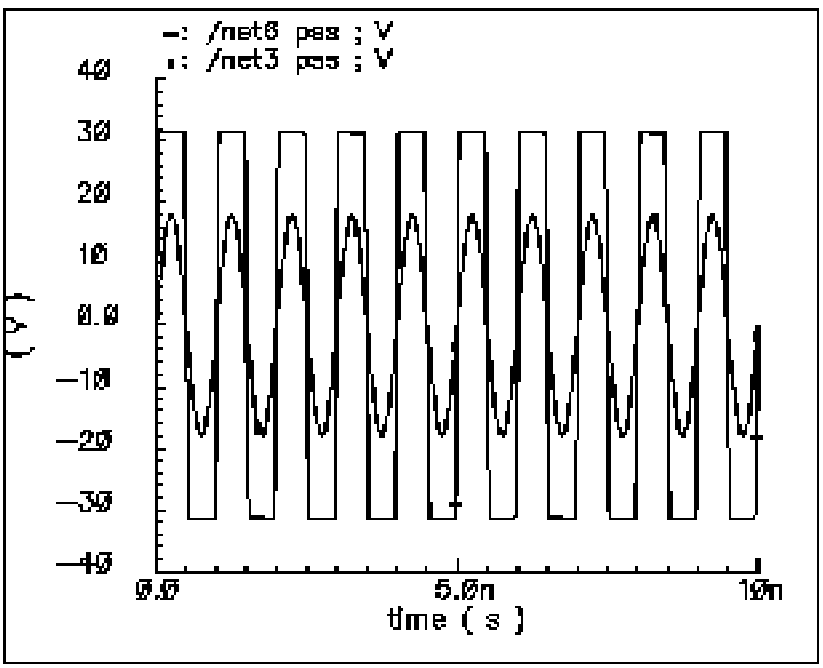
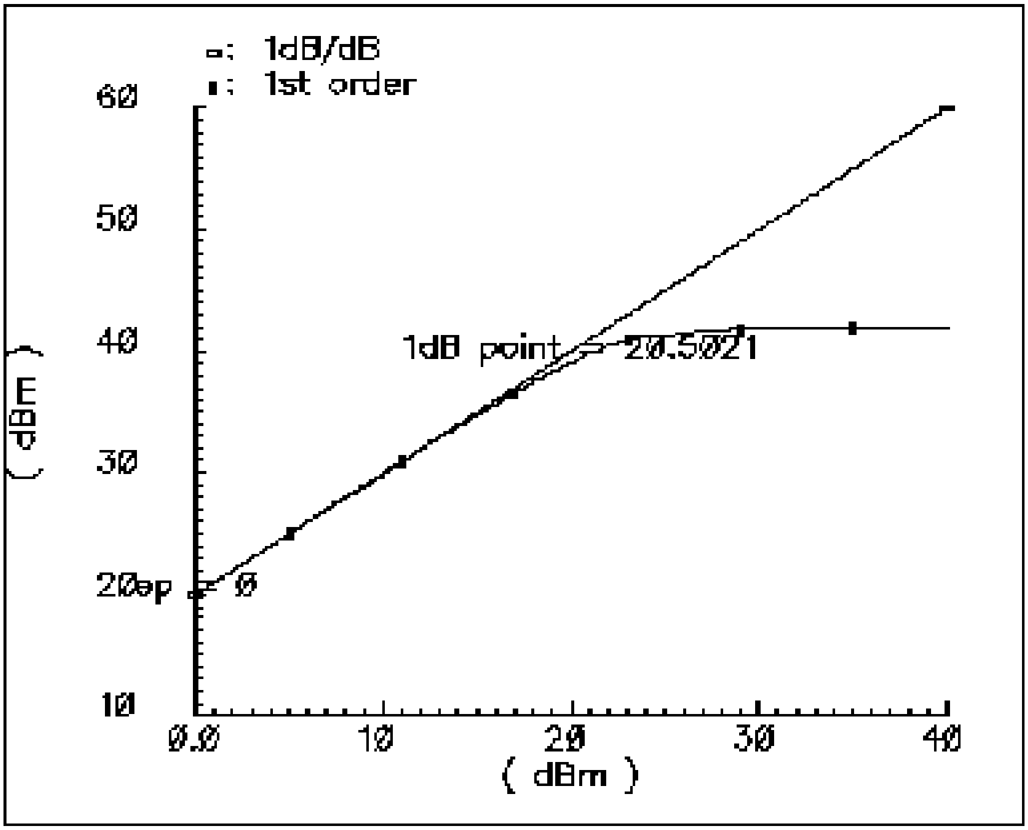


Figure D-9 shows the 1 dB compression point and the saturation power. This difference is caused by the 50 Ω load impedance. The specified output referenced 1 dB compression point is 40 dBm, which SpectreRF captures as 39.6.

Note: If `Psat` is too much larger than `p1db`, the `Psat` you want might not be satisfied.

Figure D-9 1dB Compression Point and Saturation Power



Oscillator

Oscillator models describe the essential information for a typical oscillator, more precisely, a local RF power source.

The definition of the model in the Verilog-A language is as follows:

```
module osc(out);
   electrical out;
   inout out;
   parameter real power = 10;
   parameter real f = 1e9 from (0:inf);
   parameter real rout = 50 from (0:inf);
   parameter real floor = -60 from (-inf:0);
   parameter real f1 = 1000 from (0:1e6);
   parameter real n1 = -40 from (bottom:0);
```

```
parameter real fc = 0 from [0:f1);
```

The parameters are

`power`	Output power when matched [dBm]
`freq`	Output frequency [Hz]
`rout`	Output impedance [W]
`floor`	Noise floor [dBc/Hz]
`f1`	Frequency point for `n1` [Hz]
`n1`	Phase noise at `f1` [dBc/Hz]
`fc`	Corner frequency of white phase and flicker phase [Hz]

This model is not an autonomous model. Rather, it simply generates a sinusoidal wave with the specified impedance, power level, and phase noise characteristics.

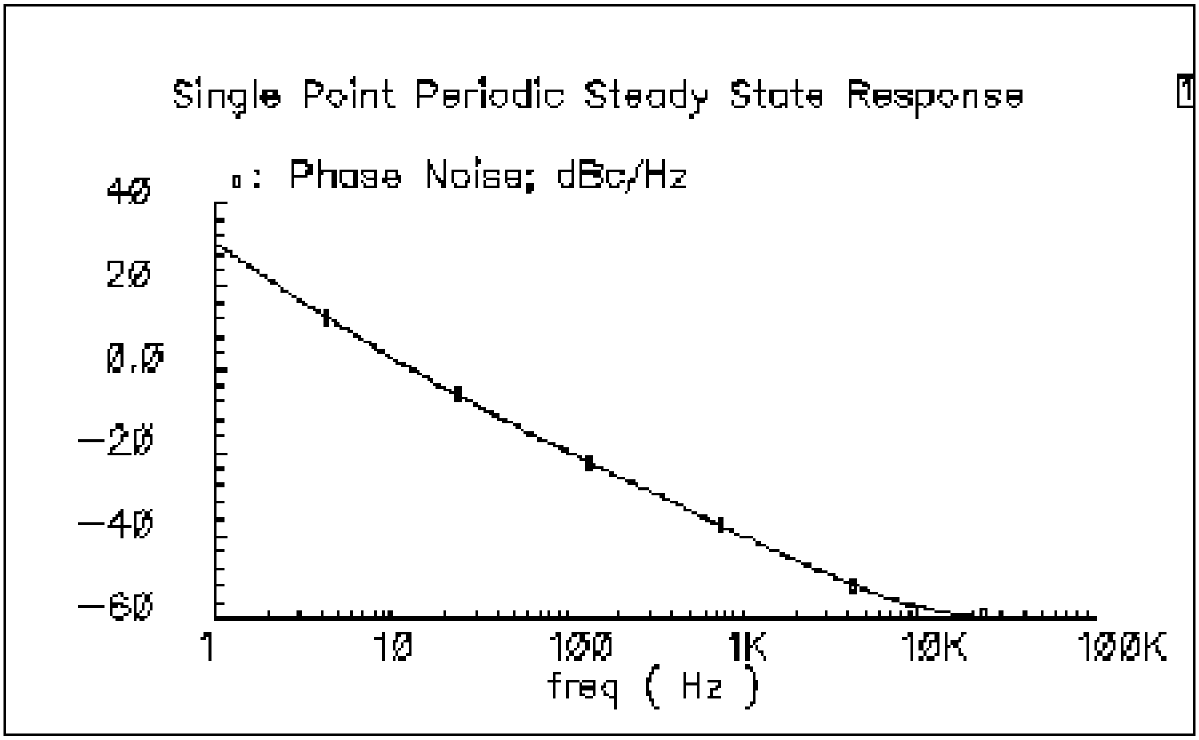
When the load is matched to the internal impedance, the load dissipates the specified output power. You can specify the noise floor of the output signal. Furthermore, by adding one point (frequency, phase noise), you can specify $1/f^2$ frequency noise (corresponding to the phase noise induced by white noise). If f_c, the corner frequency of white phase and flicker phase noise, is bigger than 0, $1/f^3$ frequency noise (flicker-noise-induced phase noise) is further specified. Otherwise, $1/f^3$ noise is not included.

The phase noises that are symmetric around the carrier are correlated. The noise floor, however, is not correlated.

Figure D-10 shows the phase noise of the oscillator model. In this figure, the specified parameters are:

- *noise floor* = -60 dBc/Hz
- f_1=1 K
- n_1=-40 dBc/Hz
- f_c =100

Figure D-10 Phase Noise for the Oscillator



Quadrature Signal Generator

The quadrature signal generator model is included because, in quadrature receiver design, a phase shifter is ordinarily used to generate the quadrature signal from one signal source such as the VCO. However, a phase shifter is hard to implement in a wide band model.

A quadrature signal consists of two signals with a 90-degree phase difference but with identical noise and amplitude.

The Verilog-A module is declared as follows:

```
module quadrature(lead, lag);
   electrical lead, lag;
   inout out_cos, out_sin;
   parameter real power = 10;
   parameter real f = 1e9 from (0:inf);
   parameter real rout = 50 from (0:inf);
   parameter real floor = -60 from (-inf:0);
   parameter real f1 = 1000 from (0:1e6);
```

```
parameter real n1 = -40 from (bottom:0);
parameter real fc = 0 from [0:f1);
```

The parameters are

`power`	Output power when matched [dBm]
`freq`	Output frequency [Hz]
`rout`	Output impedance [W]
`floor`	Noise floor [dBc/Hz]
`f1`	Frequency point for `n1` [Hz]
`n1`	Phase noise at `f1` [dBc/Hz]
`fc`	Corner frequency of white phase and flicker phase [Hz]

The difference between the quadrature signal generator model and the oscillator model is that the oscillator has only one output node but the quadrature signal generator has two output nodes, `lead` and `lag`. In the quadrature signal generator model, when the power levels, output impedances, and noise sources are identical, the two outputs, `lead` and `lag,` have a 90-degree phase difference.

Phase Shifter

In digital RF system designs, quadrature signal processing involves the phase splitting of high-frequency signals. The most common use of such components is to generate two signals that have a 90-degree phase difference based on one signal source (such as the RF signal or oscillator output). Another common use for a phase shifter is to combine two signals after adding a 90-degree phase difference, as in image-rejection receiver designs.

The Verilog-A module is declared as follows:

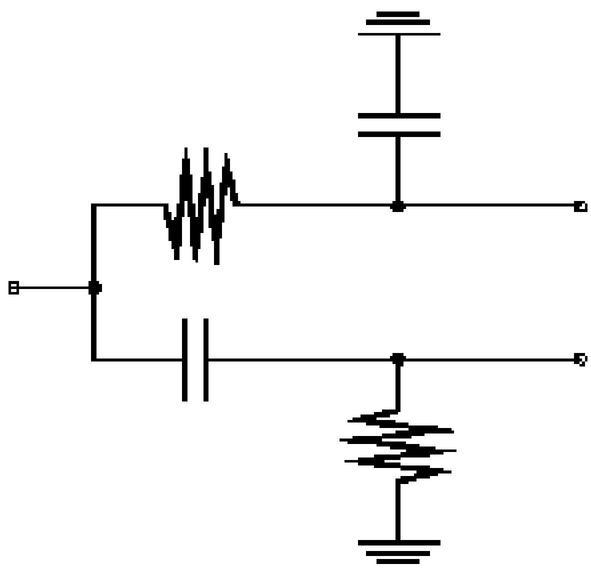
```
module shifter(single, lag, lead);
    inout single, lag, lead;
    electrical single, lag, lead;
    parameter real freq = 1e9 from (0:inf);
    parameter real r    = 50 from (0:inf);
```

The parameters are

`freq`	Frequency of operation [Hz]
`r`	Resistance [Ω] (see Figure D-11)

Internally, the phase shifter is implemented using the RC-CR circuit as shown in Figure D-11. While the phase difference is also 90-degrees when the `lead` and `lag` have the same output impedance, only at the operating frequency do the magnitudes remain the same. This circuit network also generates white noise.

Figure D-11 Phase Shifter



There are two buffered versions of the shifter:

- The *shifter_combiner* is used to combine two signals so that they add if one leads the other by 90 degrees and so that they cancel if it lags by 90 degrees.
- The *shifter_splitter* splits a signal into two signals 90 degrees out of phase with each other.

You specify the input and output impedances. These networks are noiseless.

Measurement Elements

The *Measurement* category contains elements used to facilitate measurements and diagnostics. It includes the instrumentation blocks and the baseband signal generators. These models are not part of any specific RF design architecture.

This section describes the *Measurement* elements and includes with an explanation of how to change the FIR filters inside the baseband signal generators.

Note: All of the baseband signal sources generate digitally filtered signals. The baseband sources will not work with SpectreRF because the digital filters have hidden states.

The *Measurement* category contains the following elements, discussed in the sections that follow.

Measurement elements available in previous releases

- CDMA Signal Source
- GSM Signal Source
- π/4-DQPSK Signal Source
- Eye-Diagram Generator

Measurement elements new in this release

- Rectangular-to-Polar Coordinate Transformation
- Polar-to-Rectangular Coordinate Transformation
- Ideal Transformer
- Instrumentation Block
- Offset Instrumentation Block
- Instrumentation Terminator
- Baseband Driver
- Phase Generator

CDMA Signal Source (CDMA_reverse_xmit)

The CDMA signal source (`CDMA_reverse_xmit`) generates a reverse-link (handset-to-base-station) IS-95 signal with the following characteristics:

modulation	Offset QPSK
symbol rate	1.2288 megasymbols per second
sample rate	4.9152 megasamples per second

Two separate 16-bit pseudo-noise generators generate the *I* and *Q* spreading sequences operating at the sample rate.

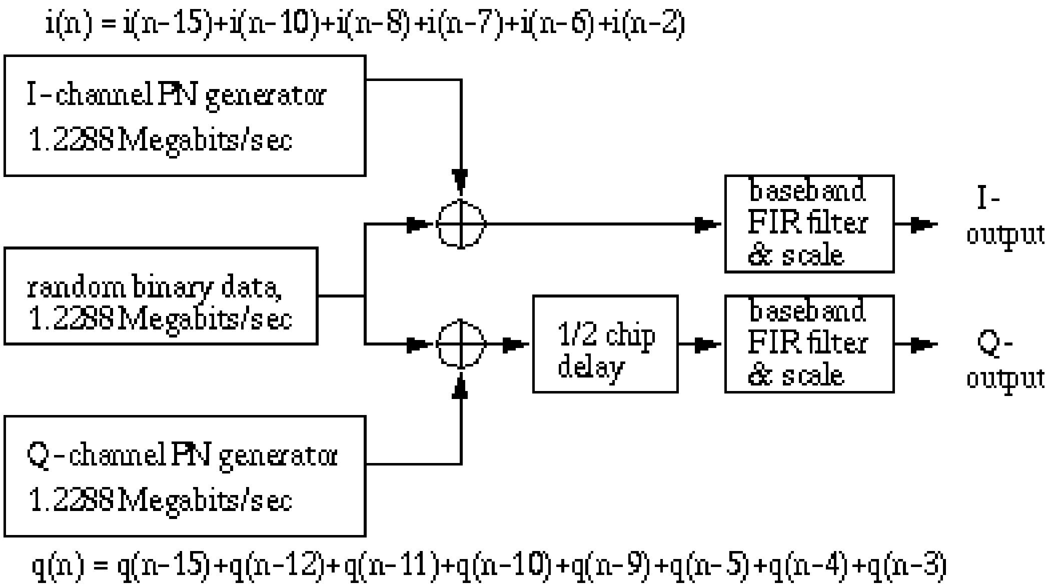
The CDMA source

- Generates a random bit at the symbol rate
- Oversamples it by a factor of 4
- Spreads the bit with the *I* and *Q* spreading sequences
- Filters each sequence with a 48-tap FIR filter. The filter coefficients are the impulse response of a raised cosine filter.

The CDMA signal source (`CDMA_reverse_xmit`) generates a reverse-link (handset-to-base-station) IS-95 signal. The modulation is offset QPSK with a symbol rate of 1.2288 Megasymbols per second and a sample rate of 4.9152 Megasamples per second. Two separate 16-bit pseudo-noise generators generate the *I* and *Q* spreading sequences operating at the sample rate. The CDMA source generates a random bit at the symbol rate, oversamples it by a factor of 4, spreads the bit with the *I* and *Q* spreading sequences, and then filters each sequence with a 48-tap FIR filter. The filter coefficients are the impulse response of a raised cosine filter.

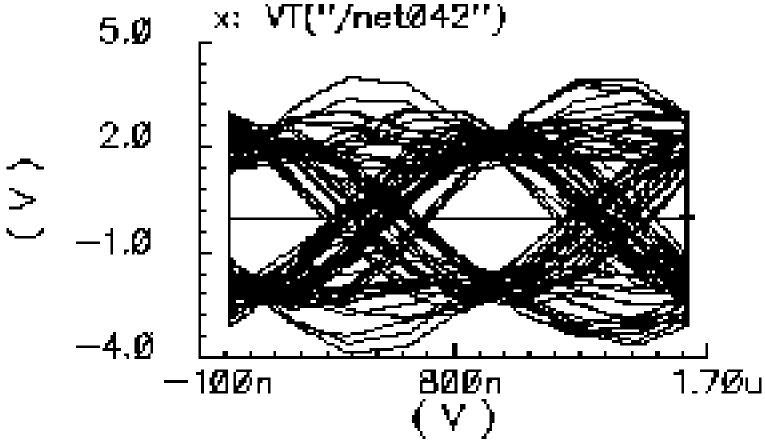
Figure D-12 shows a block diagram of the signal generator.

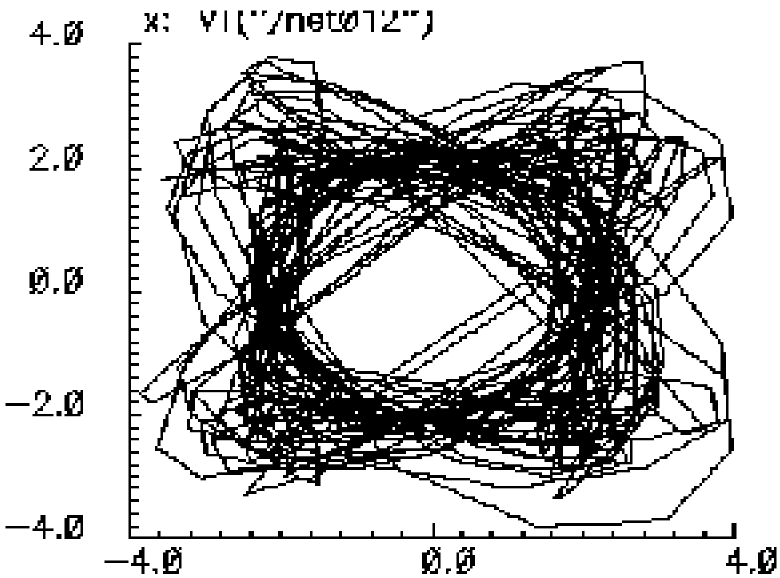
Figure D-12 CDMA Baseband Test Signal Generator



The eye-diagram generator (`eye_diagram_generator`) created the eye-diagram and trajectory. Figure D-13 shows the eye-diagram of one of the outputs and the trajectory of both outputs.

Figure D-13 Eye Diagram and CDMA Trajectory





Instance Parameters

The *amplitude* parameter sets the amplitude of the unfiltered signals. An amplitude of 1 means that each FIR filter is driven by 1-volt impulses. If you change the internal variable *IMPULSE_PULSE* to 2, the filters are driven by 1-volt pulses of four samples duration.

The *seed* parameter changes the seed for the random number generator.

Outputs

The CDMA signal generator creates four output signals:

i_bin_node	The *I* unfiltered binary output.
i_out_node	The filtered *I* output.
q_bin_node	The *Q* unfiltered binary output.
q_out_node	The *Q* filtered output.

Changing the FIR Filter

You cannot change the FIR filter, such as the tap length and tap coefficients, directly from the instance. However, you can do so using the Modelwriter as described in "Modifying the Baseband Signal Generators Using the Modelwriter" on page 1110.

Output Transitions

The filtered outputs slew linearly from one value to the next because the rise and fall times in the transition statements equal one period. To make the outputs take abrupt steps, copy the module to your library and change the rise and fall times in the last transition statements.

GSM Signal Source (GSM_xmtr)

The GSM source generates a signal conforming to the GSM standard. The modulation is GMSK and the data is generated in frames of 3 fixed start bits, 142 random data bits, 3 fixed stop bits, and 8.25 fixed guard bits. (The embedded deterministic pattern and quarter of a bit is necessary to produce the correct spectrum.) The bit rate is 270833.333 bits per second and the sample rate is four times that.

The FIR filter is a Gaussian filter implemented with 32 taps.

Figure D-14 shows a block diagram of the signal source.

Figure D-14 GSM Baseband Signal Generator

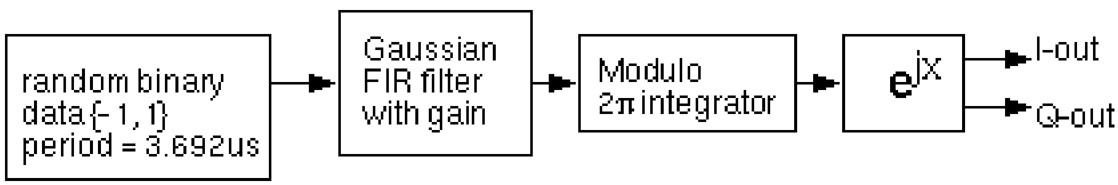
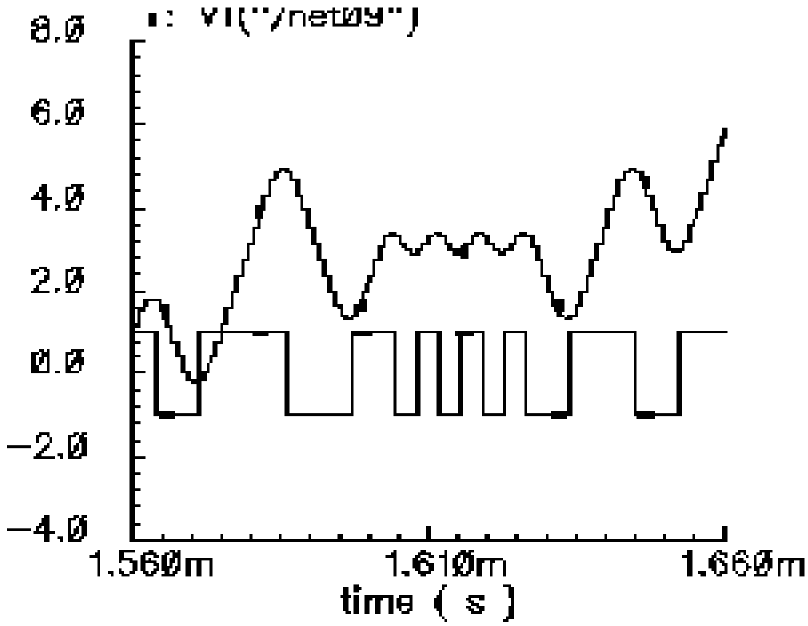


Figure D-15 shows the binary data stream and the corresponding angle.

Figure D-15 GSM Binary Data and Resulting Phase



Instance Parameters

The *amplitude* parameter sets the amplitude of the unfiltered signals. An amplitude of 1 means that each FIR filter is driven by 1-volt impulses. If you change the internal variable *IMPULSE_PULSE* to 2, the filters are driven by 1-volt pulses of four samples duration.

The *seed* parameter changes the seed for the random number generator.

Outputs

The generator creates four output signals

angular_node	The output signal.
i_out_node	The phase, multiplied by the amplitude.
bin_node	The bit stream being transmitted.
q_out_node	The phase multiplied by the amplitude.

Changing the FIR Filter

You cannot change the FIR filter, such as the tap length and tap coefficients, directly from the instance. However, you can do so using the Modelwriter as described in “Modifying the Baseband Signal Generators Using the Modelwriter” on page 1110.

Output Transitions

The filtered outputs slew linearly from one value to the next because the rise and fall times in the transition statements equal one period. To make the outputs take abrupt steps, copy the module to your library and change the rise and fall times in the last transition statements.

Pi/4-DQPSK Signal Source (pi_over4_dqpsk)

Figure D-16 shows the block diagram for this source.

Figure D-16 Π/4-DQPSK baseband signal generator

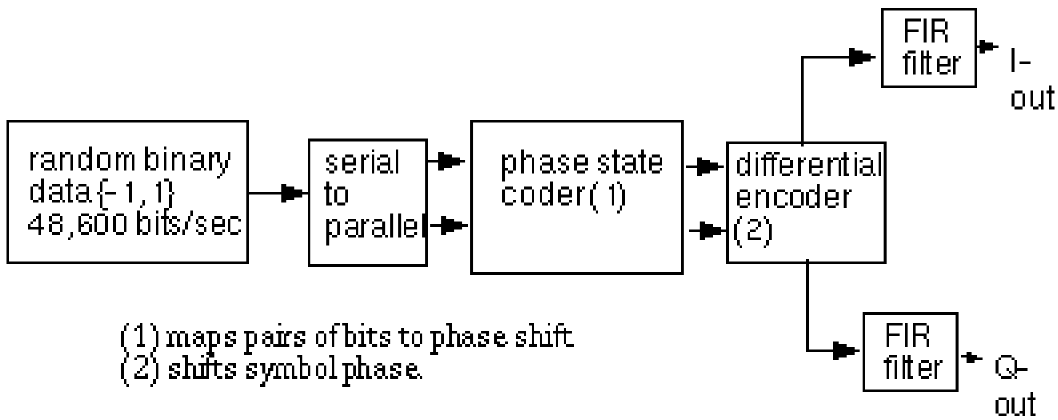


Table D-1 shows how the phase shift is generated.

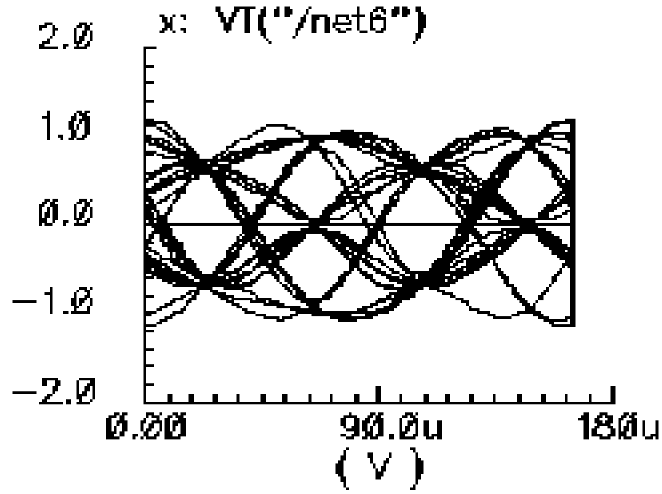
Table D-1 Phase Shift

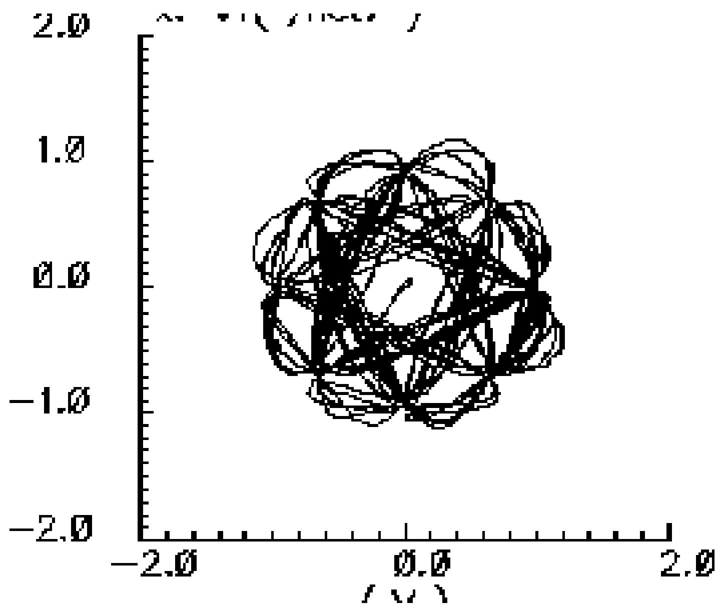
1st bit	2nd bit	Phase shift
0	0	$\pi/4$
0	1	$3\pi/4$
1	0	$-\pi/4$
1	1	$-3\pi/4$

The symbol rate is 24300 symbols per second and the sample rate is 8 times that. The FIR filter is a raised cosine filter implemented with 64-taps.

The eye-diagram generator (`eye_diagram_generator`) created the eye-diagram and trajectory. Figure D-17 shows the eye-diagram and trajectory for this generator.

Figure D-17 Eye Diagram and Pi/4 Trajectory





Instance parameters

The *amplitude* parameter lets you set the amplitude of the unfiltered signals. An amplitude of “1” means that each FIR filter will be driven by 1-volt impulses. If you change the internal variable *IMPULSE_PULSE* to 2, the filters will be driven by 1-volt pulses of four samples duration.

The *seed* parameter lets you change the random number generator seed.

Outputs

The generator creates four output signals.

i_out_node	The phase, multiplied by the amplitude.
q_out_node	The phase, multiplied by the amplitude.
phase_shift_out	The phase shift from one symbol to the next.

Changing the FIR filter

You cannot change the FIR filter, such as the tap length and tap coefficients, directly from the instance. However, you can do so using the Modelwriter as described in "Modifying the Baseband Signal Generators Using the Modelwriter" on page 1110.

Output transitions

The filtered outputs slew linearly from one value to the next because the rise and fall times in the transition statements equal one period. To make the outputs take abrupt steps, copy the module to your library and change the rise and fall times in the last transition statements.

Eye-Diagram Generator (eye_diagram_generator)

The eye-diagram generator creates eye-diagrams and trajectories for the baseband signal generators.

Input

The input to the eye-diagram generator is the *I* or *Q* output of one of the baseband signal generators.

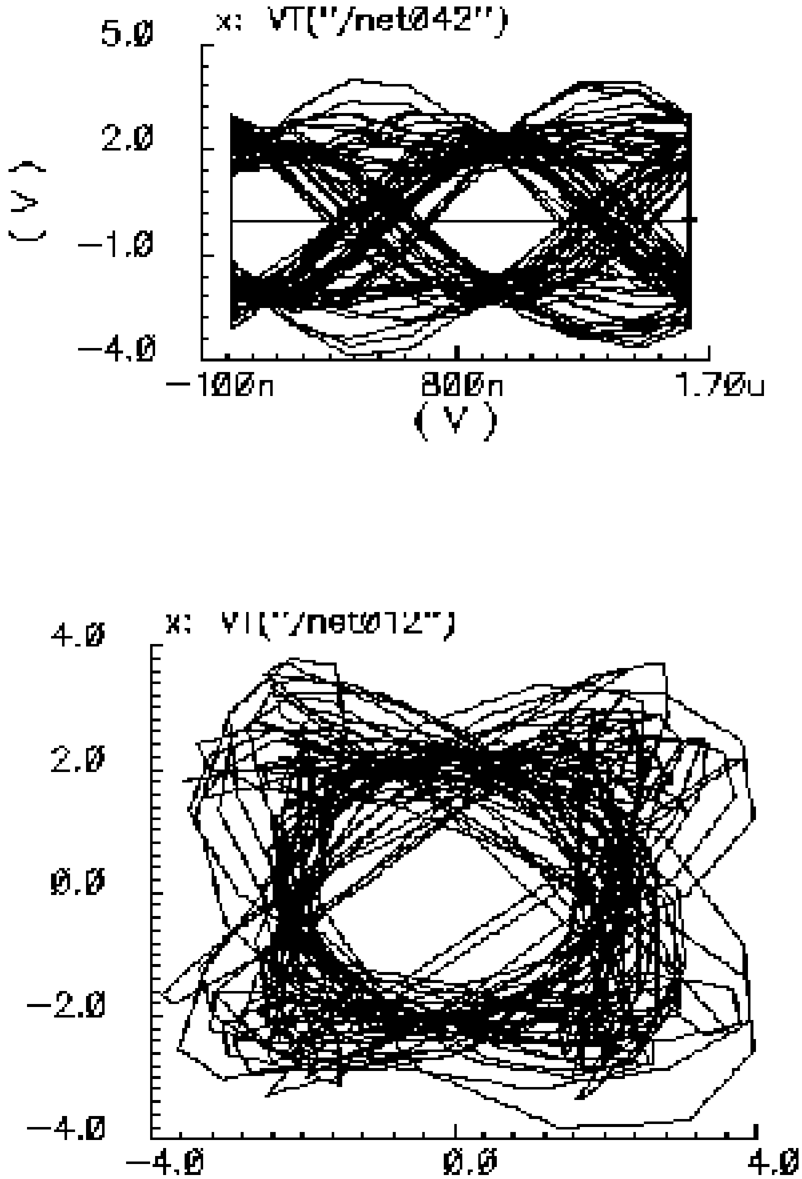
Output

The eye-diagram generator has two outputs labeled “y-axis” and “x-axis”. The eye-diagram is generated by plotting the “y-axis” output against the “x-axis” output.

Note: The eye-diagram generator does not work with Envelope Following Analysis to generate similar plots. This is because the Envelope Following harmonic time analysis is generated by a post-processing step and the eye-diagram generator works during simulation.

Figure D-13 shows an eye-diagram of one of the outputs and the trajectory of both outputs for the CDMA baseband signal generator.

Figure D-18 Example Eye Diagram and CDMA Trajectory



Modifying the Baseband Signal Generators Using the Modelwriter

The baseband signal generators use FIR (finite impulse response) filters to shape their output pulses. Shaped output pulses serve several purposes:

- They help keep the transmitted signal inside the specified band.
- They work with their receive counterparts to maximize the signal-to-noise ratio.
- Together with their receiver counterparts, they satisfy the Nyquist sampling criterion for an intersymbol-interference-free channel.

The Modelwriter gives you a convenient user-interface for creating variations on the baseband signal generators in the library. The most likely variation is in the FIR filter. This section explains how to use the Modelwriter to create a new GSM generator with different FIR filters.

1. Bring up the Modelwriter and do the following:

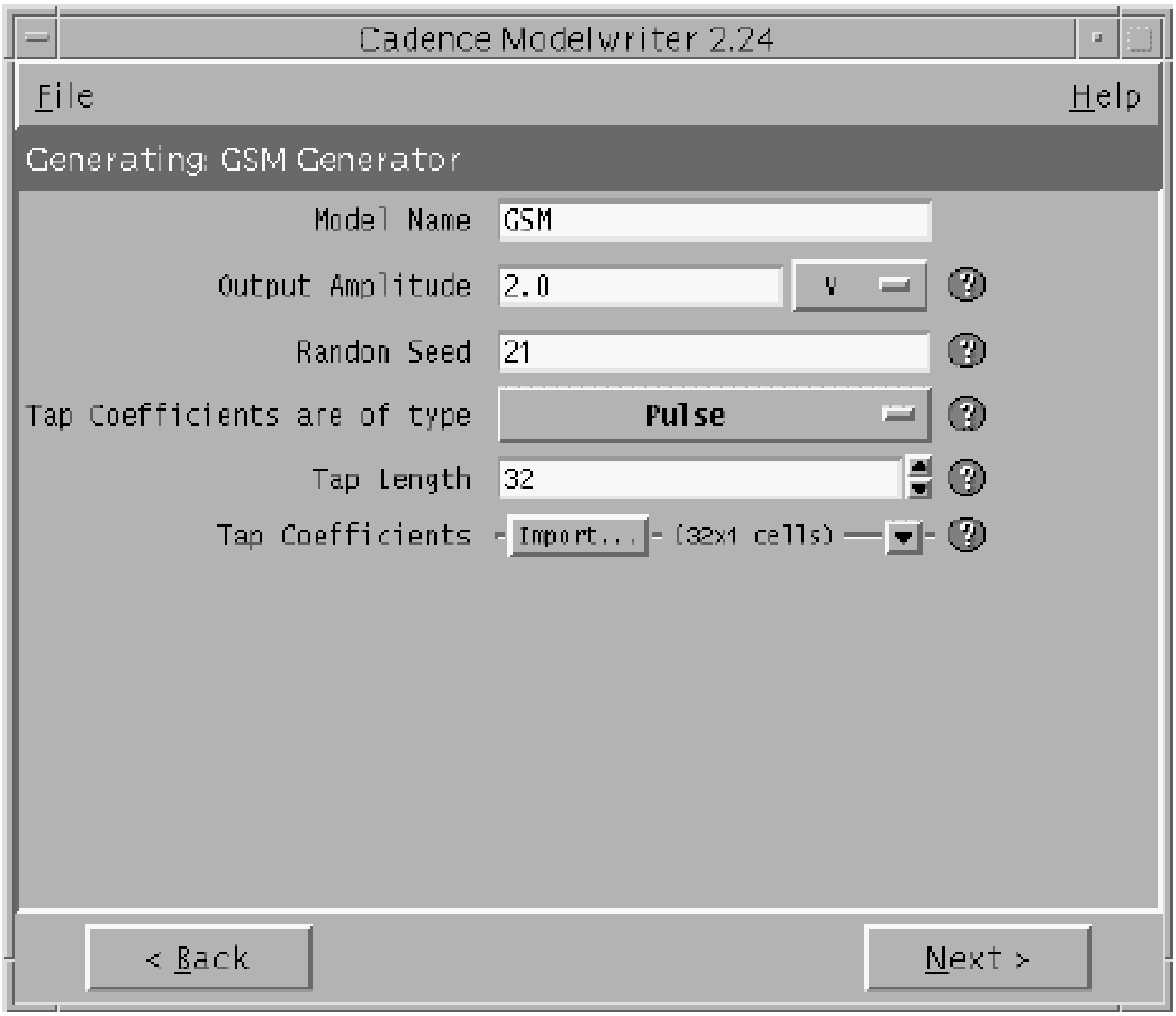
   a. Double click on the Telecom folder.

   b. Select the GSM generator.

   c. Click on the *Next* button in the lower right hand corner of the Modelwriter window.

   You should now see the picture in Figure D-19 in the window.

Figure D-19 GSM Generator



2. Specify how you plan to drive the filter by selecting the type of tap coefficients.

   FIR filters can be driven by pulses or impulses.

3. Specify the length of the FIR filter in the *Tap Length* field.

4. You can specify the tap coefficients in two ways

   ❑ From a file

   ❑ By direct manual entry.

   To read the coefficients from a file do the following:

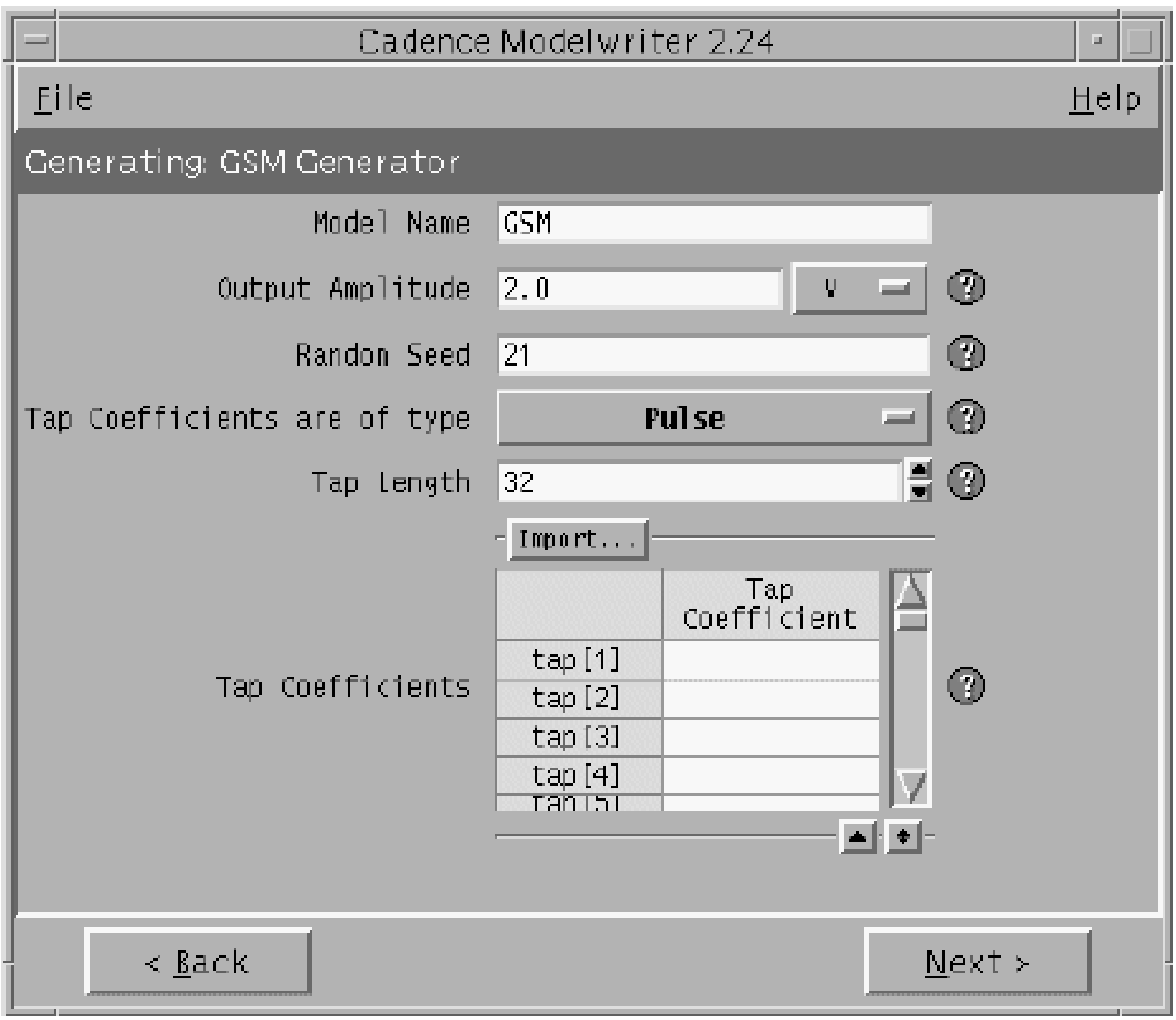
a. Select the *Import* button.

b. Enter the path to the file.

c. Click on *Open*.

The coefficients appear in the window as shown in Figure D-20. (The tap[1] coefficient multiplies the filter state with the least amount of delay, the filter state closest to the input.)

Figure D-20 Tap coefficients



To enter the coefficients manually do the following:

a. Click on the lower rightmost button flagged in the form.

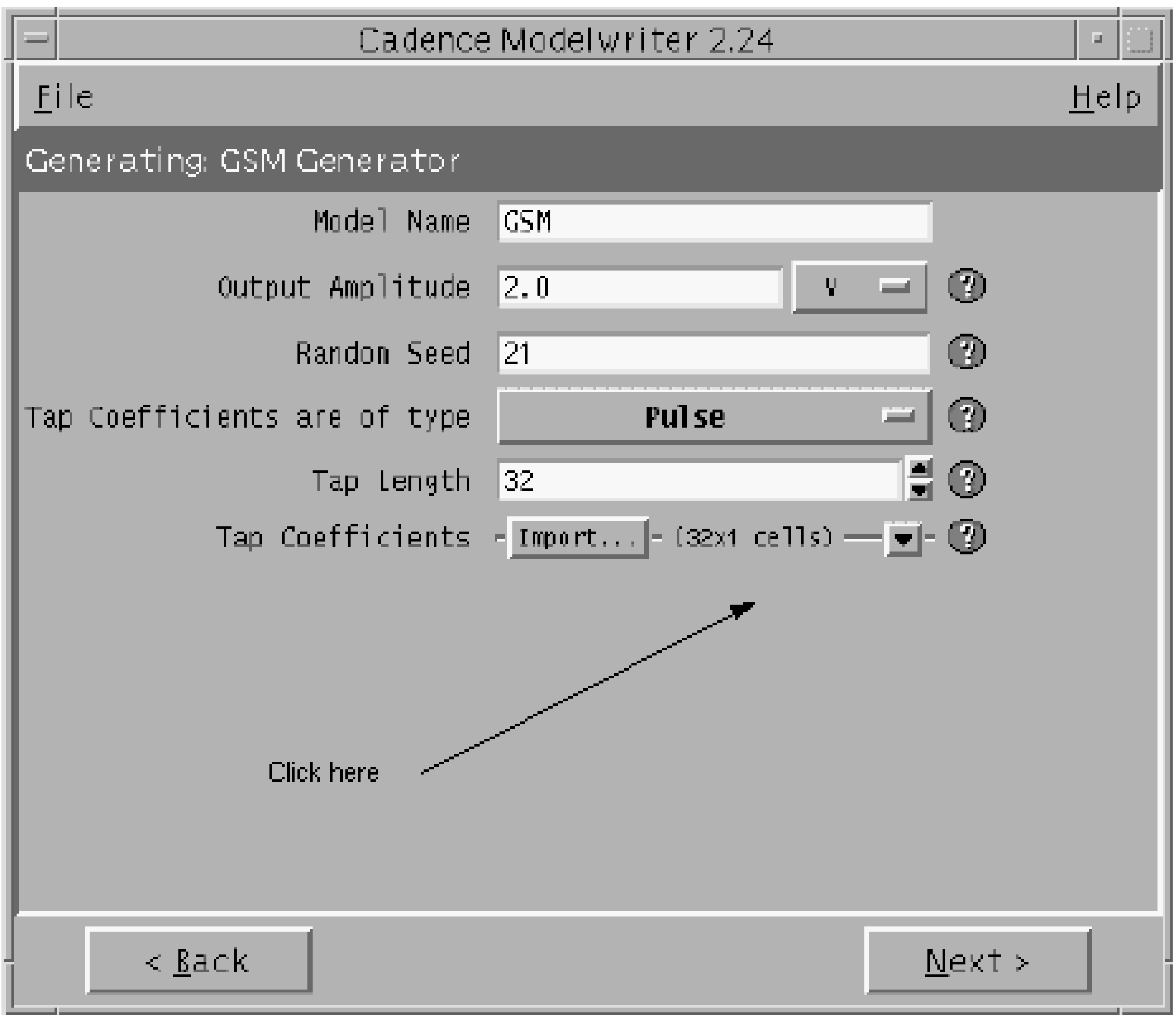
See Figure D-21.

b. Enter the coefficients,

c. Click *next* to view the model,

d. To write the model to a file, click *Save Generated Code…*.

Figure D-21 Manual entry of the tap coefficients



Testbenches Elements

The *testbenches* category contains the test circuits used to define model specifications. Where possible, the element names are in terms of standard RF measurements. The most precise way to describe a measurement is with a test circuit, set up instructions, and sample measurements. The circuits in the testbench category serve this purpose. Table lists the Testbenches elements and includes a link to where each one is used.

Table D-2 Elements in the Testbenches Category of rfLib

Element Name	Example Where the Element is Used
AM_PM_test_ckt	"AM/PM Conversion Parameters" on page 1135
BB_ind_cap_test	"RLC Test Circuits" on page 1172
PB_BB_filter_comparison	"Comparison of Baseband and Passband Models" on page 1195
PB_ind_cap_test	"RLC Test Circuits" on page 1172
ava_pwr_gain	"Available Power Gain Parameter" on page 1122
demod_ip3	"IQ Demodulator" on page 1154
dwn_cnvt_test	"RF-to-IFand IF-to-RF Mixers" on page 1162
mixer_ip3	"IP3 Parameter" on page 1129
mod_1dbcp	"Available power gain and 1dB compression point:" on page 1149
mod_demod_test	"IQ Demodulator" on page 1154
noise_figure	"Noise Figure Parameter" on page 1127
one_db_cp	"Output 1dB Compression Point Parameter" on page 1126
quad_and phase_error_demo	"Quadrature error and phase error:" on page 1151
shifter_combiner_test	"Phase Shifter Combiner" on page 1189
shifter_splitter_test	"Phase Shifter Splitter" on page 1185
up_cnvt_test	"Testing the up_cnvrt Mixer" on page 1166
view_switching	

Uncategorized Elements

Elements in the *Uncategorized* category are data files created by elements in the *testbenches* category. The following list shows typical files you might find here.

- cdma_2ms_idata
- cdma_2ms_qdata
- dqpsk_20ms_idata
- dqpsk_20ms_qdata
- gsm_5ms_idata
- gsm_5ms_qdata

See "Testbenches Elements" on page 1114 for more information.

Bottom-Up Design Elements

The *bot_upBB* library contains Verilog A versions of the J-model which have two primary uses:

- To facilitate estimation of Adjacent Channel Power Ratio (ACPR)
- To import transmitter impairments into the Alta™ SPW environment

The *bot_upBB* library contains the following 5 verilog A versions of the J-model:

- `1st_order_j_model`
- `3rd_order_j_model`
- `5th_order_j_model`
- `7th_order_j_model`
- `9th_order_j_model`

These verilog-A models read the same extracted J-model data files as the Alta SPW J-models.

For more information about using extracted J-model data files to estimate ACPR (adjacent channel power ratio) and about extracting J-models, see Chapter 9, "Creating and Using Transmitter J-Models."

Models for Top-Down RF System Design

If you are a system-level RF designer charged with designing RF systems from the specifications provided by DSP design teams, you can make use of the models described in this section. You can use the models described here for designing the top down analog RF subsystems that fit into larger DSP systems—from specifications provided by the DSP system designers. In particular, these models form a canonical set of top-down behavioral baseband models for exploring RF architectures in analog design environment. For methodology information and examples using these models, see Chapter 12, "Methods for Top-Down RF System Design."

The baseband and passband models come from the following *rfLib* categories

- Category *top_dwnBB* contains models of common RF function blocks.
  - The default view of each model is the baseband view (called *veriloga*).
  - Each model also has a differential passband view (called *veriloga_PB*).
  - Each model has a *symbol* view used in schematics.

  The only exception is the *BB_loss* model, which is meant only for baseband analysis and has no passband view.

  Note: The VGA_BB is an engineering release of a variable gain amplifier. It is not documented or supported in this release and is subject to change in future releases.
- Category *top_dwnPB* contains single-ended passband versions of the baseband models.
- Category *measurement* contains the instrumentation block and baseband signal generator models used to make RF measurements. These elements are not part of an RF architecture. They simply facilitate RF measurements and diagnostics.
- Category *testbenches* contains the test circuits used in this chapter to define the model specifications in the *rfLib*. Where possible, the models are specified in terms of standard RF measurements. The most precise way to describe a measurement is with a test circuit, set up instructions, and sample measurements. The circuits in the testbenches category serve that purpose

The *top_dwnBB* models provide RF designers with a fast method to map RF system specifications into detailed RF designs. The baseband models facilitate fast evaluation of candidate RF architectures specified with DSP metrics. The passband views of the baseband models provide a behavioral system testbench for checking detailed designs of individual RF system components.

Baseband models are behavioral models and all behavioral models sacrifice some accuracy for increased simulation speed. Such sacrifices are usually acceptable in architectural studies because many implementation-dependent details do not affect high level decisions. The modeling approach taken in top-down design is to simulate only those effects that drive the decisions at hand.

Baseband modeling in no way replaces passband modeling. Some effects missed by equivalent baseband models can affect high level decisions. However, the application of baseband models early followed by passband models later minimizes the number of slow simulations needed at low levels of design abstraction. Baseband models help you to quickly weed out designs that would surely fail tests simulated with passband models.

The success of a modeling approach to top-down design hinges on knowing two things

- How the models fit into the design flow
- Exactly what each modeling parameter means. This section defines, as clearly and concisely as possible, the parameters that specify the models

Baseband and Passband Models

The baseband and passband elements contained in the RF Library, *rfLib*, are organized into the following categories

.

Table D-3 Categories of Baseband and Passband Elements in the rfLib

Category in *rfLib*	**Description of Content**
`top_dwnBB`	The Baseband Library. This category contains models of common architectural function blocks. The default view of the models in this category is the baseband view called veriloga, but all models in this category have a differential passband view called *veriloga_PB*. The only exceptions are the *BB_loss* and *VGA_BB* models. The *BB_loss* model is meant only for baseband analysis. The *VGA_BB* is an engineering release of a variable gain amplifier. It is not documented or supported in this release and is subject to change in future releases
`top_dwnPB`	The Passband Library. This category contains single-ended passband versions of the baseband models

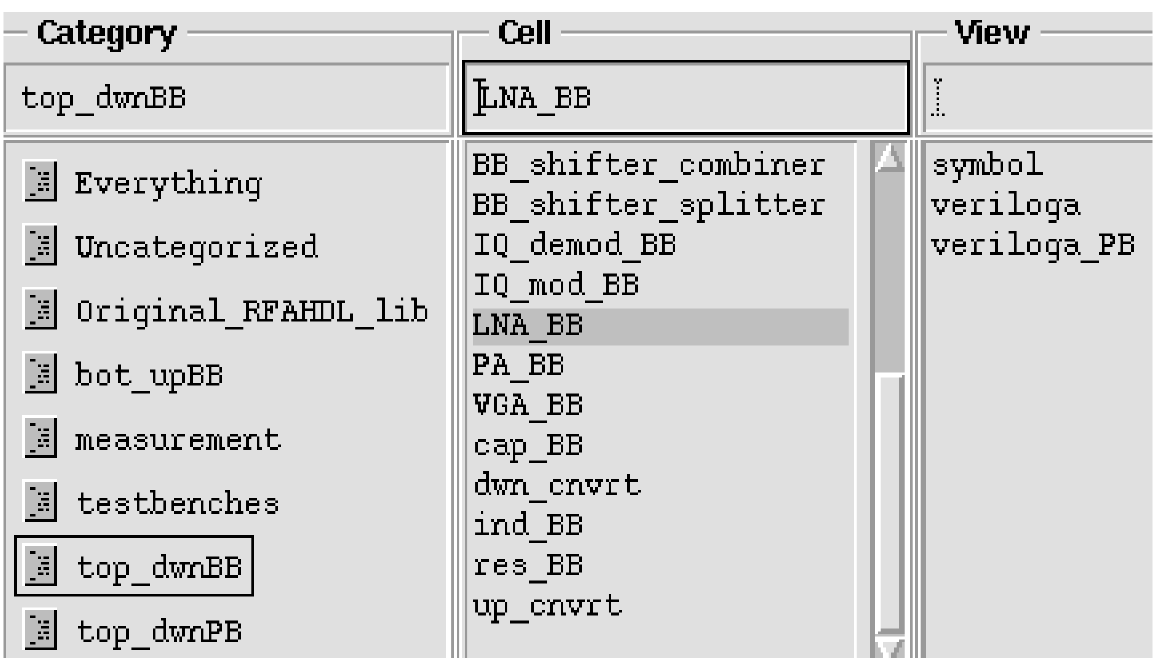
Table D-3 Categories of Baseband and Passband Elements in the rfLib

Category in *rfLib*	Description of Content
`measurement`	Contains models used to make measurements; includes the instrumentation blocks and the baseband signal generators. These elements are not part of an RF architecture. They simply facilitate measurements and diagnostics.
`testbenches`	Contains the test circuits used to define the model specifications in the *rfLib*. Where possible, the models are specified in terms of standard RF measurements. The most precise way to describe a measurement is with a test circuit, set up instructions, and sample measurements. The circuits in the testbenches category serve that purpose

Except for AM/PM conversion, all model specifications are defined by measurements taken from the passband models in the *top_dwnPB* category.

Most baseband models in *top_dwnBB* have both a baseband view (*veriloga*) and a differential passband view (*veriloga_PB*). Figure D-22 shows the Library Manager with *LNA_BB*, the low noise amplifier baseband cell selected.

Figure D-22 Library Manager Showing Views of LNA_BB in top_dwnBB



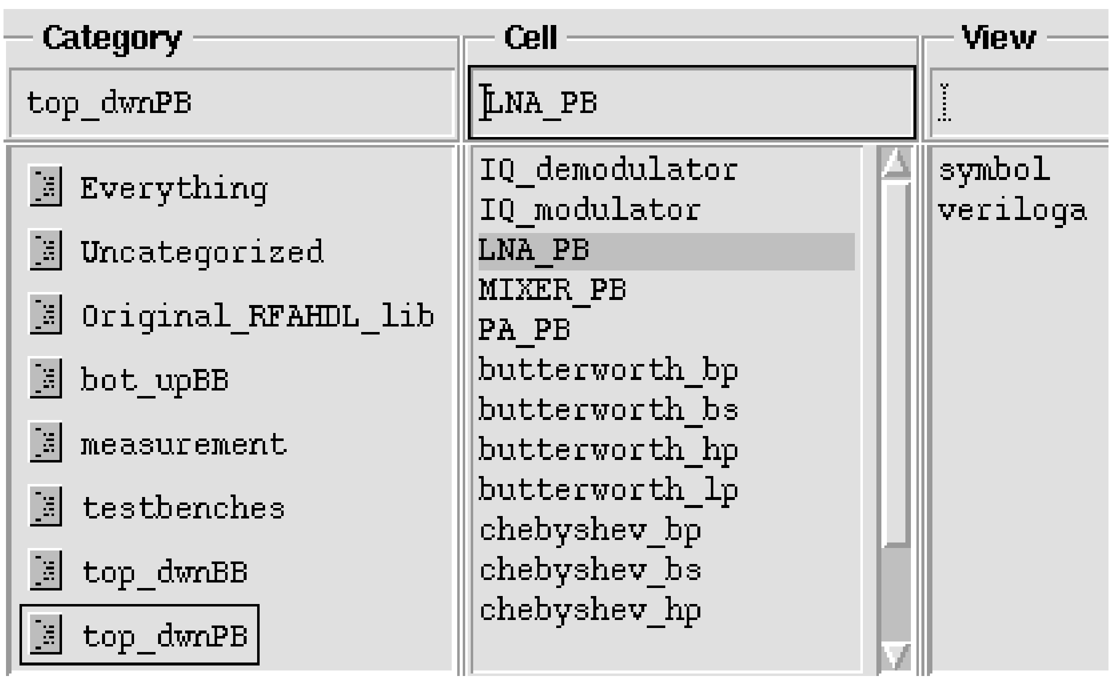
LNA_BB has three views:

`symbol`	The schematic symbol for *LNA_BB*.
`veriloga`	The baseband LNA element with third-order effects.
`veriloga_PB`	The passband differential view of the same LNA element.

Those baseband models that do not have a differential passband view (*veriloga_PB*) are meant only for baseband use.

The models in the *top_dwnPB* category of *rfLib* are singled ended versions of the differential passband views. The single-ended passband models describe the measurements but the measurements also apply to the differential passband versions. Figure D-23 shows the Library Manager with *LNA_PB*, the low noise amplifier baseband cell, selected.

Figure D-23 Library Manager Showing Views of LNA_PB in top_dwnPB



LNA_PB has two views:

`symbol`	The schematic symbol for *LNA_PB*.
`veriloga`	The passband LNA element with third-order effects.

Assumptions About Behavioral Models

All behavioral models require assumptions and the top-down models at hand are no exception. The assumptions are summarized below:

- Non-linear RF components are memoryless.
- Noise in the models is white, additive Gaussian noise which is split equally between the two output pins. In any model, *I* and *Q* noise sources are uncorrelated.
- Except for loading effects, signal flow is unilateral.
- Terminal impedances are purely resistive.
- The following is a single assumption stated three ways:
  - Only third-order non-linearities matter—even-order non-linearities do not matter.
  - Between any two baseband models, carrier harmonics are negligible.
  - Baseband models need not simulate IP2 or DC offsets.

  The basic problem stated here in three ways is that *even* non-linearities do not produce output power at the carrier's fundamental frequency. Baseband models model only effects such as IP3 because third order distortion affects the fundamental. Second order distortion only affects the fundamental at the output of a subsequent, cascaded, non-linear device. You *can* build a baseband model of the two cascaded, non-linear devices in order to simulate how second order distortion in the first device affects the output of the second device at the fundamental. You *cannot* cascade individual baseband models of the two devices and observe any second-order effects.
- You can model all mixers, including those in modulators and demodulators, as a static non-linearity followed by an ideal multiplier.
- All local oscillators are sinusoidal.

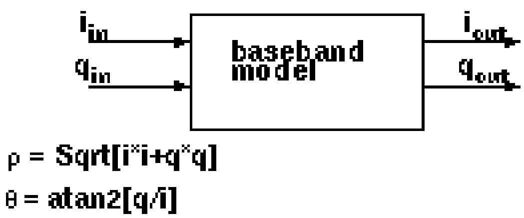
In addition, two compatibility assumptions apply.

- All baseband models, *except* the baseband signal generators and the instrumentation blocks, are compatible with SpectreRF.
- All models are written in Verilog-A.

Inputs and Outputs for Baseband Models

Except where noted, all baseband models have inputs and outputs as shown in Figure D-24.

Figure D-24 Inputs and Outputs on Baseband Models



Note: Be careful not to confuse baseband models with two-port S-parameter models.

When a device loads or drives adjacent devices with finite resistances, the terminals on the symbol are wider apart than when they load/drive adjacent models with ideal resistances (zero or infinite resistance).

Some Common Model Parameters

Model parameters specify the baseband and passband models in the *top_dwnBB* and *top_dwnPB* categories in *rfLib*.

- The AM/PM conversion parameters specify the baseband models only.
- All other parameters specify both the passband and baseband models of a function block.

The more common parameters are defined in terms of passband test circuits described in this section. Except for the AM/PM conversion parameters, baseband parameters are not described with test circuits because, given the same parameters, the baseband models simulate the same peak voltages and currents as simulated by the passband models. Given the identical parameter values, the baseband and passband models simulate the same peak voltages and currents, but not the same power levels.

Parameters described here include

- Available Power Gain Parameter
- Input and Output Resistance Parameters
- Output 1dB Compression Point Parameter
- Noise Figure Parameter
- IP3 Parameter

- AM/PM Conversion Parameters
  - AM/PM sharpness
  - |radians| @ 1 dB cp
  - |radians| @ big input
  - {1, 0, 1} for {cw, none, ccw}

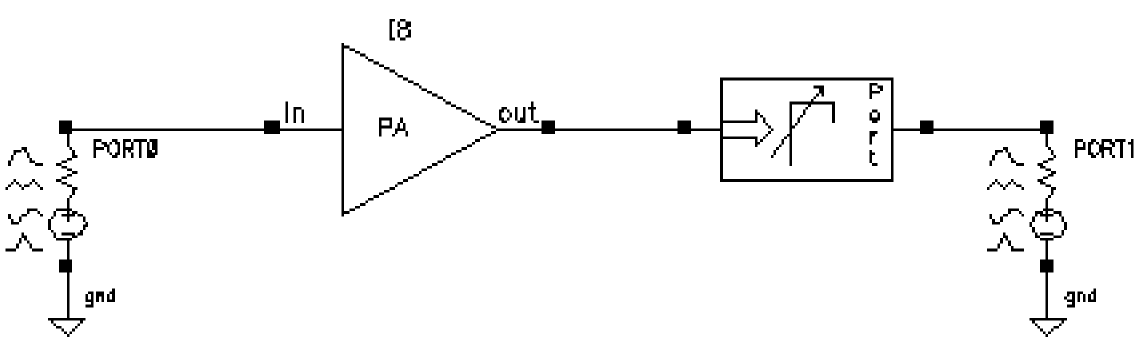
Available Power Gain Parameter

When an amplifier's load is equal to it's output resistance, available power gain equals the following:

$$10 \times \log\left(\frac{outputpower}{inputpower}\right)$$

The test circuit in Figure D-25 is listed as *ava_pwr_gain* in the *testbenches* category in *rfLib*.

Figure D-25 The ava_pwr_gain Circuit



Computing Constant Power Contours

The *ava_pwr_gain* test circuit is set up to compute constant power contours. As you would expect, maximum power transfer occurs when the load and output impedances are matched. The port adapter inserts reactive elements into the signal path to load the amplifier with the specified reflection coefficient.

Figure D-26 on page 1124 shows a Smith Chart that displays how the load power varies with the load refection coefficient.

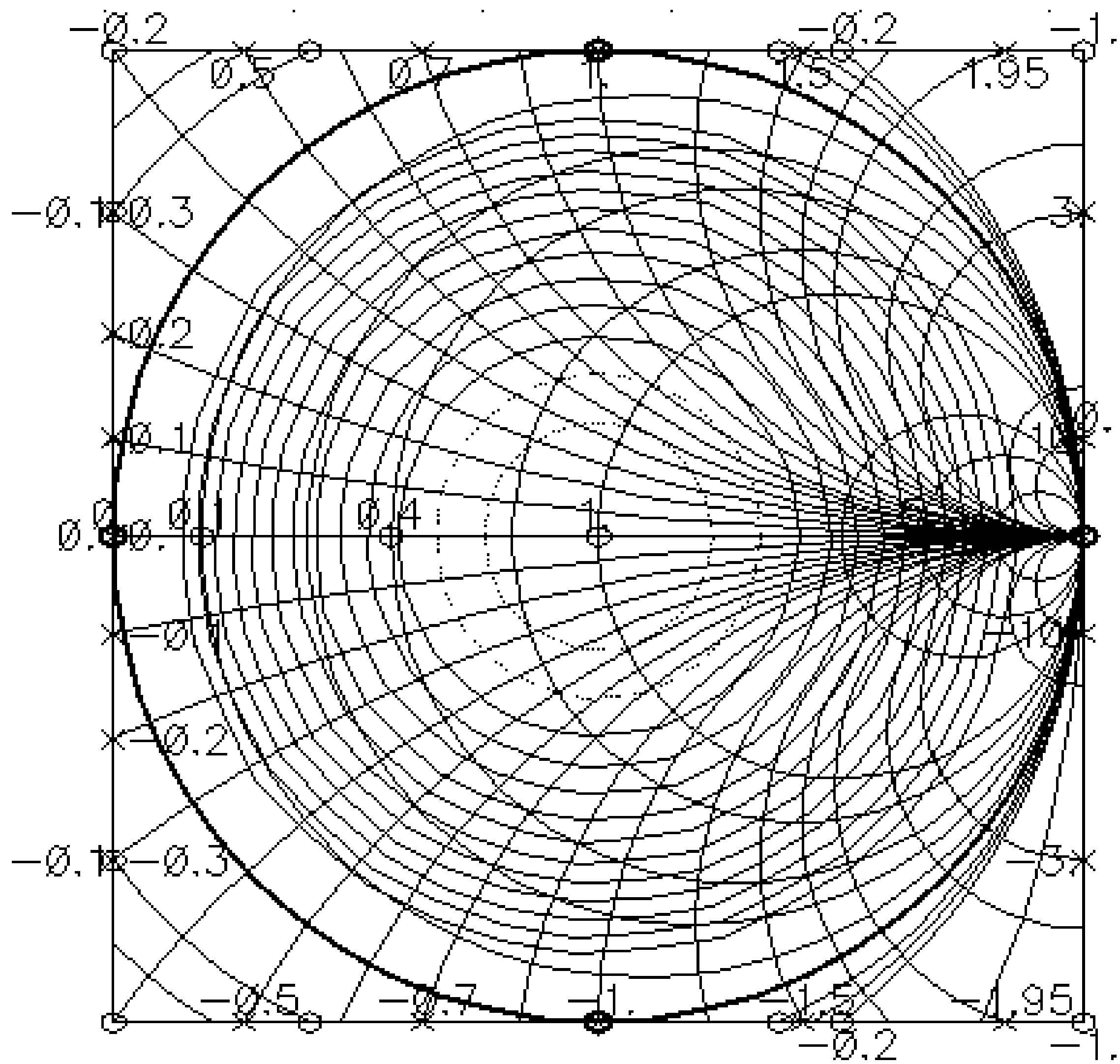
The load pull contours were computed by

- Sweeping the `pp` parameter in a PSS analysis (`pp` is the *phase* of the reflection coefficient)
- Sweeping the `mm` parameter with the Parametric Tool (`mm` is the *magnitude* of the reflection coefficient)

The *load reflection coefficient* is defined with reference to the amplifier output resistance, `300` Ohms in this case. The amplifier input resistance is `20` Ohms. The input source resistance is `50` Ohms. The amplifier 1dB compression point is set high enough to make the amplifier linear. The available power gain parameter is `20` dB.

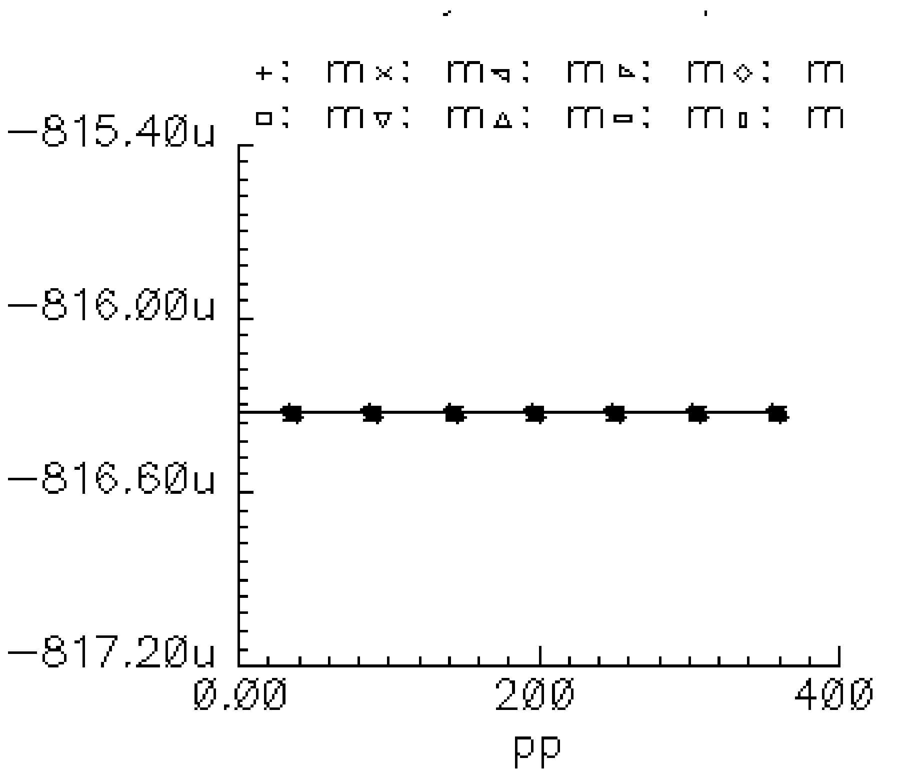
In order to generate the load pull contours you must save *both* the current flowing into the port adapter (*port*) as well as the current flowing into *Port0*.

Figure D-26 Smith Chart



When you place the cursor on the smallest contour on the Smith Chart, you can see that the amplifier delivers a maximum power of `81.63` mW to an optimum load of `300` Ohms (reflection coefficient = 0). When you plot the magnitude of the power coming from the input port against the sweep variable (`pp`, phase of the reflection coefficient) you will find that input power equals `816.3` uW, independent of load, as shown in Figure D-27. The ratio of maximum output to input power equals `100`, or dB, as specified.

Figure D-27 Input power



Note that the voltage gain in this test circuit does not equal 10 because the amplifier's input and output resistances are different. You can verify that the ratio of the output to input voltage is as follows

$$10\sqrt{\frac{R_{out}}{R_{in}}}$$

where, R_{out} is the amplifier output resistance and R_{in} is the amplifier input resistance. This assumes the amplifier is not driven into non-linear operation.

Input and Output Resistance Parameters

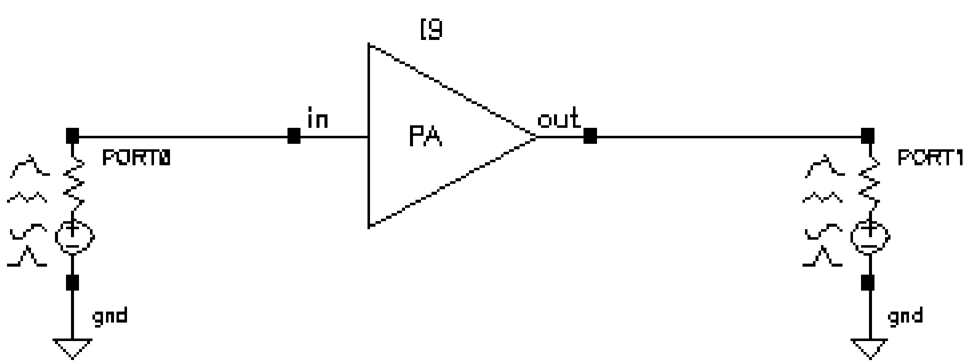
The input and output resistances specify the current drawn by the associated terminals as a linear function of terminal voltage. There is no test circuit for terminal resistances since the definition is so simple.

Output 1dB Compression Point Parameter

The 1dB compression point specifies a saturation non-linearity. It is the output power in dBm where the output power falls 1dB below the power extrapolated linearly from the amplifier's linear region of operation.

The test circuit in Figure D-28 on page 1126 is listed as *one_db_cp* in the *testbenches* category in *rfLib*.

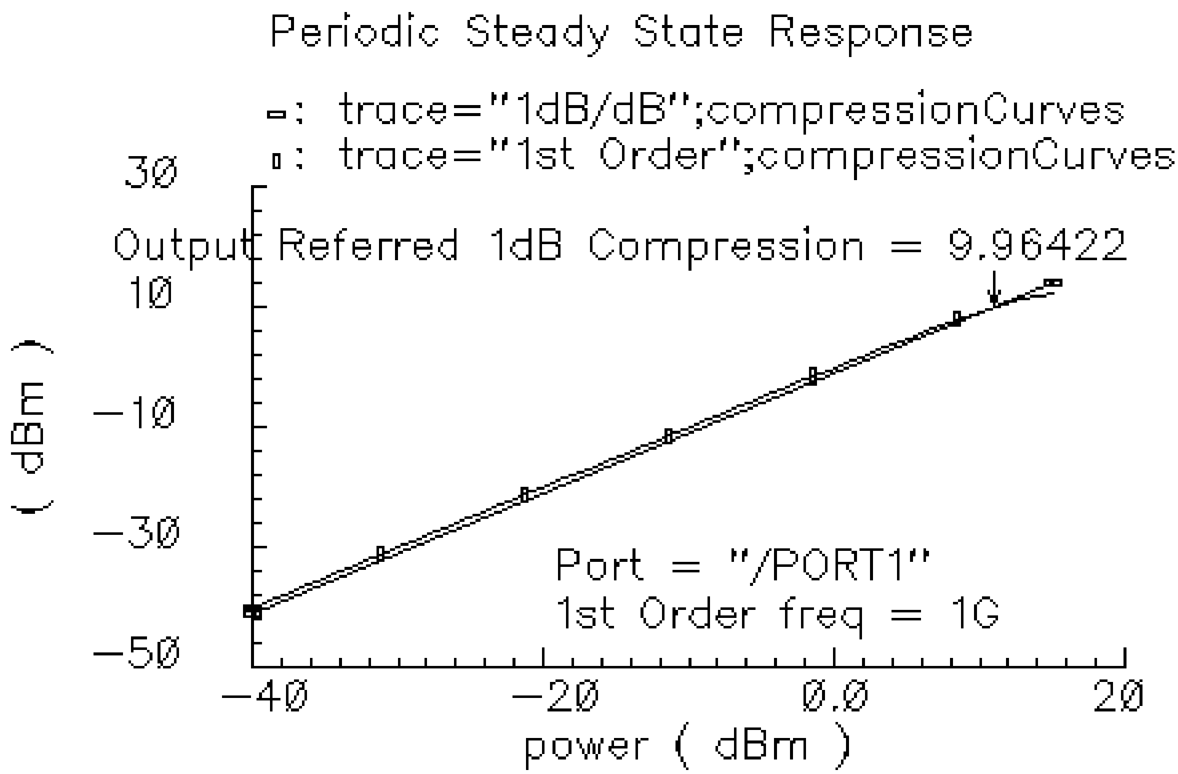
Figure D-28 The one_db_cp Circuit



In the *one_db_cp* test circuit, *power* is the dBm of power delivered by the leftmost port. The available power gain is 0dB. The 1dB compression point is 10dBm. The input and output resistances are 50 Ohms and so are the port resistances.

To measure the 1dB compression point, perform a swept PSS analysis. Sweep *power* from -40dBm to 15dBm in 50 linear steps. The output referred 1 dB compression point is computed for the 1st harmonic with an *Extrapolation Point (dBm)* of -40. Click on the rightmost port device to display the output as illustrated in Figure D-29 on page 1127.

Figure D-29 Resulting 1dB Compression Point

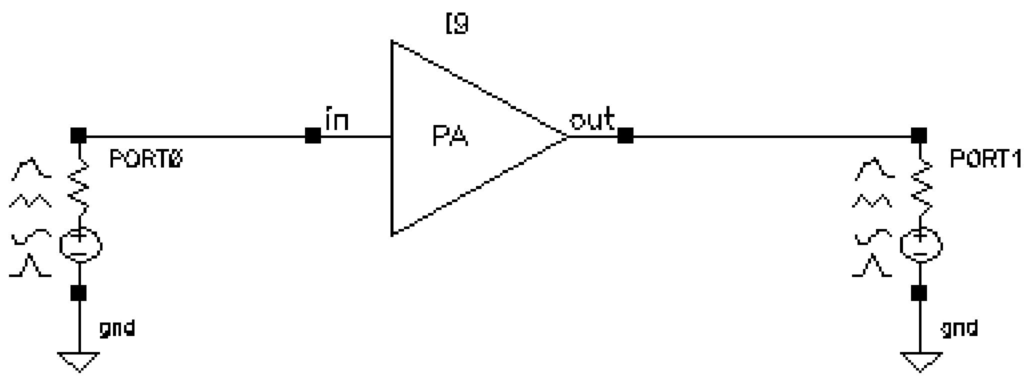


The specified output referred compression point is 10dBm. The measured value is 9.964dBm, which is fairly close to the specified value. The measured 1db compression point is as specified only when the driving source resistance matches the amplifier input resistance and the load port resistance matches the amplifier's output resistance. In all compression point and IPN calculations, input power is computed from the maximum power the input Port can deliver, not from an actual power measurement. If you mismatch either terminal you will not measure the specified compression point.

Noise Figure Parameter

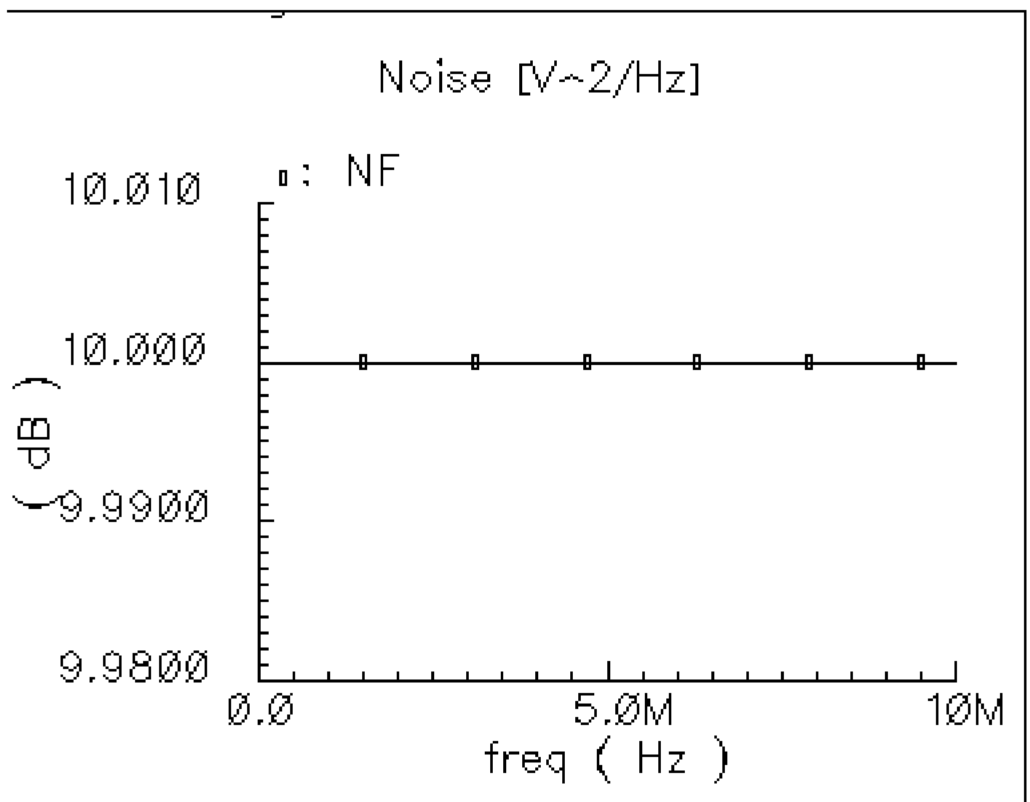
Noise figure is calculated as the input signal-to-noise ratio divided by the output signal-to-noise ratio. The test circuit for defining the noise figure parameter is shown in Figure D-30 on page 1128. The circuit is listed as *noise_figure* in the *testbenches* category of the *rfLib*. It is similar to the *one_db_cp* test circuit.

Figure D-30 The noise_figure Circuit



The specified noise figure is 10dB. A Spectre Noise analysis produces the noise figure shown in Figure D-31 on page 1129. To measure the specified noise figure, the driving port resistance must match the amplifier's input resistance. The port at the output does not have to match the amplifier's output resistance but the Port impedance should be resistive. The input probe is the leftmost port, the output port is the rightmost port. Since the model is static, you can compute noise figure over any frequency interval.

Figure D-31 Noise Figure Results

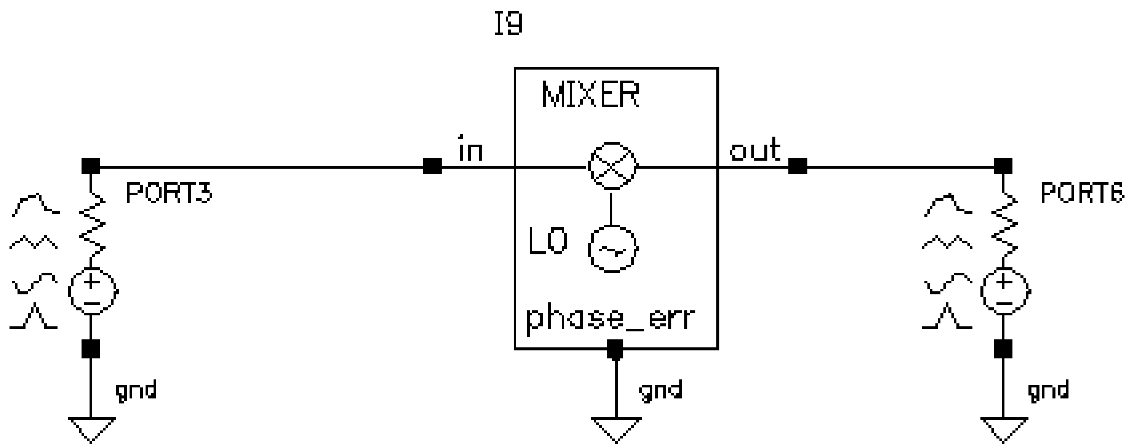


IP3 Parameter

IP3 is measured with a two-tone test. One tone is the fundamental PSS frequency while the other is the frequency in a single point PAC analysis. IP3 is defined as the input power level in dBm where the extrapolated power in one of the third order intermodulation terms equals the extrapolated power in the fundamental term. As with the 1dB compression point measurement, input and output terminals must be matched to the source and load respectively.

The IP3 specification is demonstrated step by step on the mixer model because the mixer IP3 measurement can be confusing. Figure D-32 on page 1130 shows the test circuit. The circuit is listed as *mixer_ip3* in the *testbenches* category of the *rfLib*.

Figure D-32 The mixer_ip3 Test Circuit



Measuring IP3 for a Mixer

1. Recall the circuit and bring up an Analog Environment window.
2. Set up a PSS analysis. Add a Fundamental Tone called “eee” (the name is arbitrary) at 1GHz. The 920MHz tone should already appear in the form. Autocalculate the Beat Frequency. It should be 40MHz. Use 2 Output harmonics. Sweep the PSS analysis using the “power” variable. Sweep power from -100 to 0 in 10 linear steps.
3. Set up a PAC analysis at 921MHz. Select the sidebands from an Array of indices. Add indices -21 and -25. **Why select the -21 and -25 sidebands?** Recall from the assumptions, the non-linearity occurs before the frequency translation. The input tones to the non-linearity are the large 920Mhz tone and the small signal 921MHz tone. In an IP3 measurement only one tone must be large, the other can be small. PAC analysis performs small signal perturbations on the PSS solution. One perturbation term exiting the non-linearity appears at 921MHz, right where is started. One of the third order intermodulation perturbation terms exiting the non-linearity appears at 2*920-921 = 919MHz. The ideal mixer, driven by a pure 1GHz local oscillator, translates the 921MHz tone to 921-1000=-79MHz while translating the 919MHz tone to 1000-919=81Mhz. A single point 921MHz PAC analysis produces tones displaced from harmonics of the fundamental by 921MHz. The PAC sidebands specify which harmonics to use. You save

the 79MHz tone by saving the -25th sideband because the fundamental frequency is 40Mhz and 921 - 40*25 = -79MHz. You save the 81MHz tone by saving the -21 sideband because 921-40*21 = 81MHz. Figure D-33 on page 1131shows the PAC setup.

4. Run the analysis.
5. Do a Direct Plot of the PAC results. To do this, set up the Direct Plot form as shown in Figure D-34 on page 1132.
6. In the Composer window, click on the output Port. You should see the results as shown in Figure D-35 on page 1133.

Figure D-33 PAC Setup

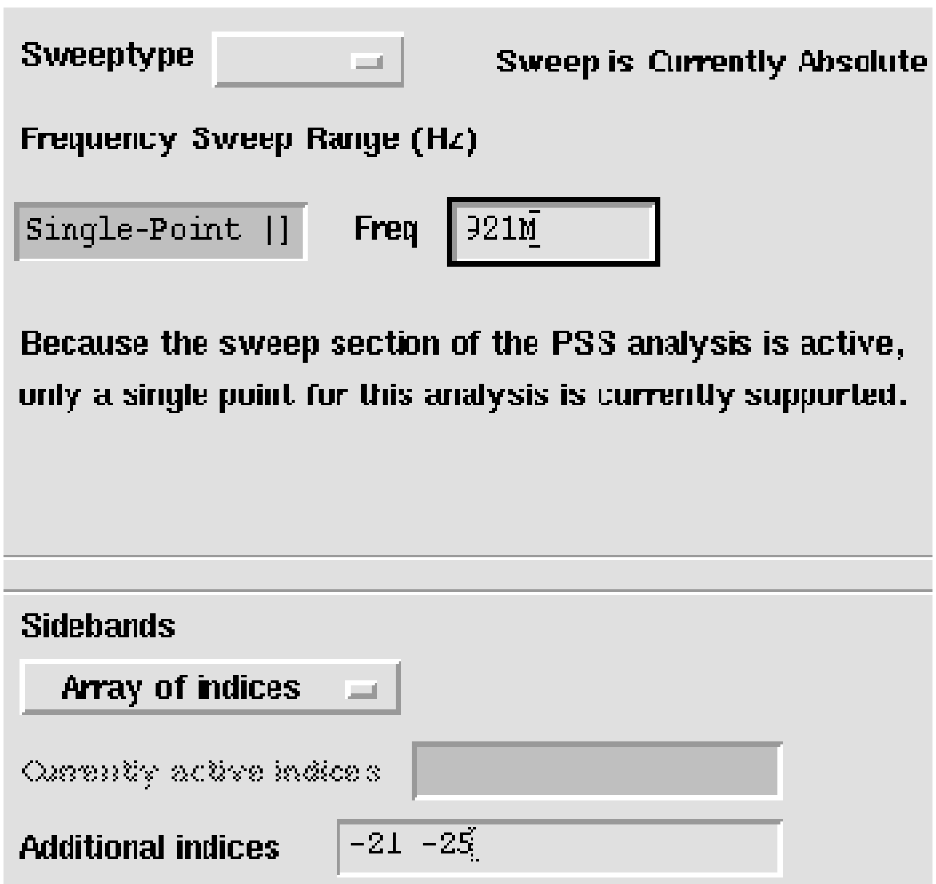


Figure D-34 Direct Plot Form

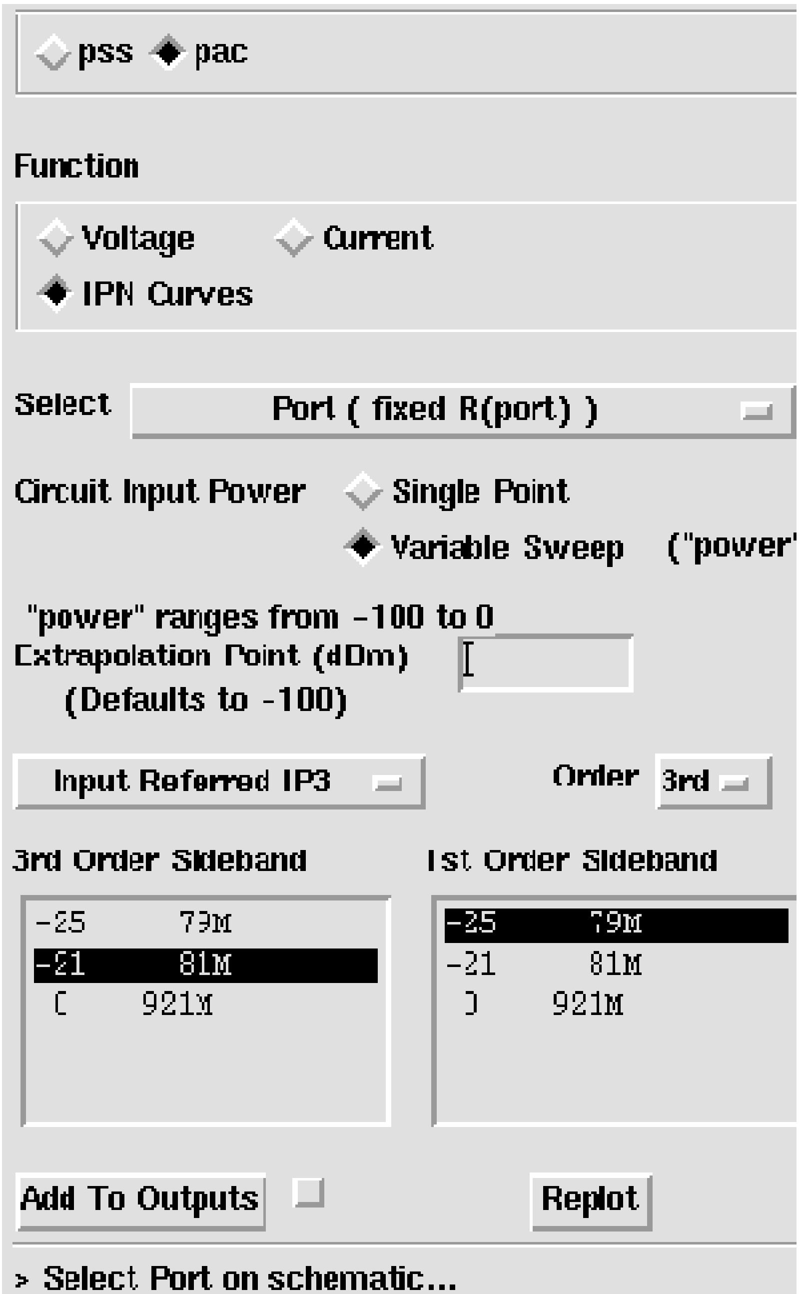
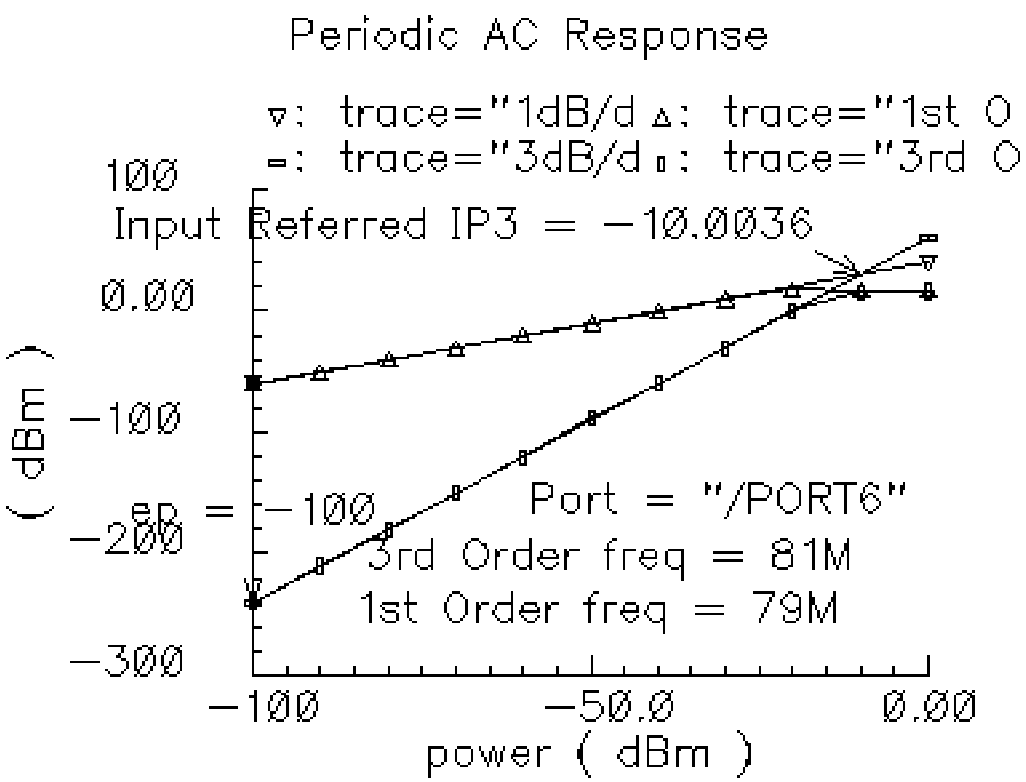


Figure D-35 IP3 Results



The measured IP3 is, -10dBm, as specified. The measured IP3 is as specified only if the input port resistance matches the input resistance of the device-under-test. Other input resistances will produce a measured IP3 different than the one specified.

Measuring IP3 for an LNA

You can measure IP3 of an LNA by replacing the mixer with an LNA and ensuring the input terminal remains matched. In this example, remove the 1GHz Fundamental Tone from the PSS analysis. The Beat Frequency should now be 920MHz. In the PAC set up, change the additional indices from -21 and -25 to -1 and -2. After the analysis completes, set up the PSS Results form as shown in Figure D-36 on page 1134. As before, the input referred IP3 is 10dBm, as specified. Figure D-37 on page 1135 shows the LNA IP3 results.

Figure D-36 Direct Plot Form for the LNA

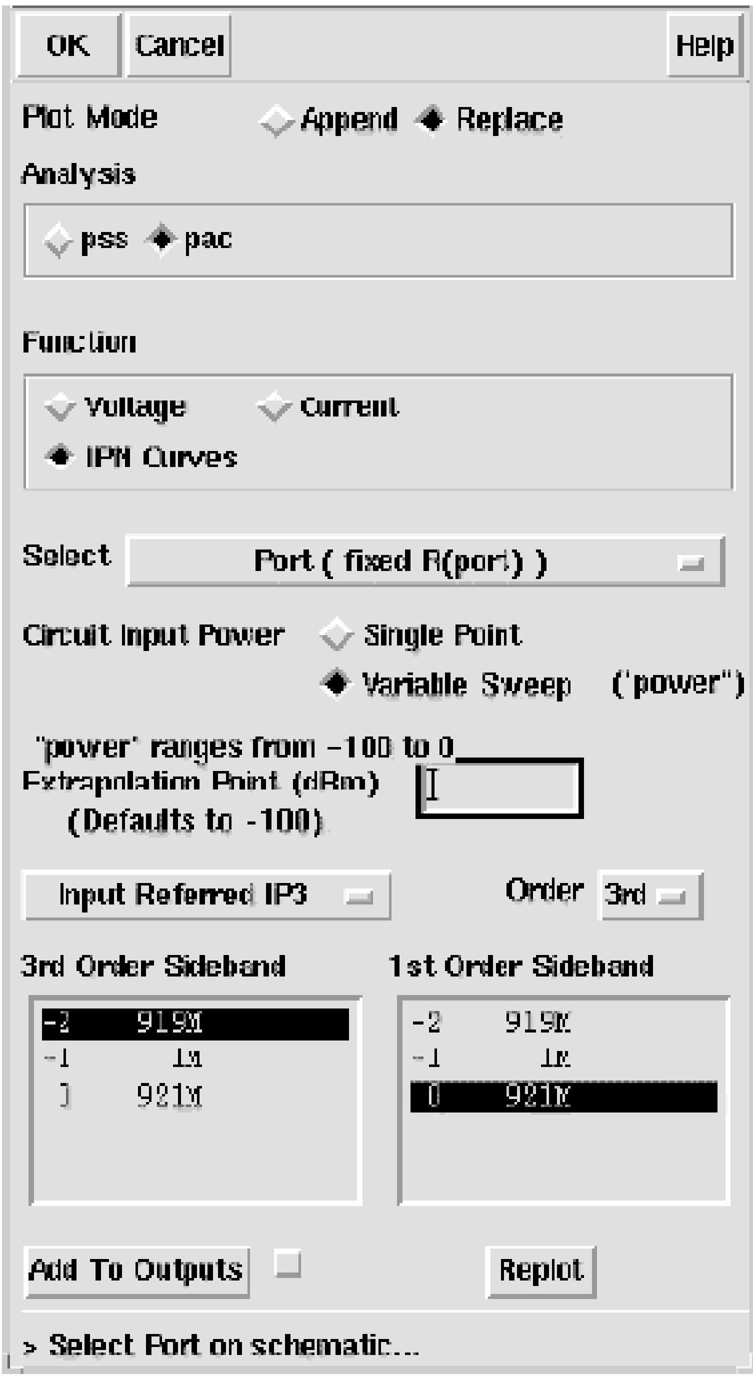
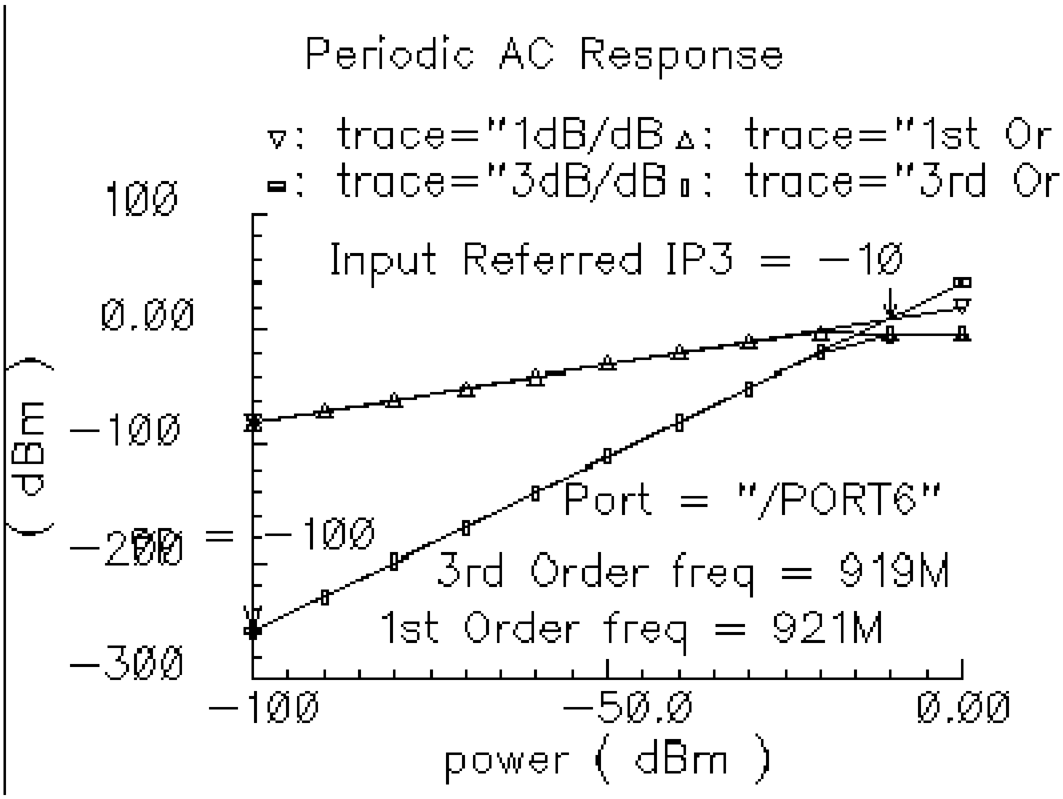


Figure D-37 Results for the LNA



AM/PM Conversion Parameters

Only the baseband models include the 4 parameters for AM/PM conversion.

Table D-4 AM/PM Conversion Parameters for Baseband Models

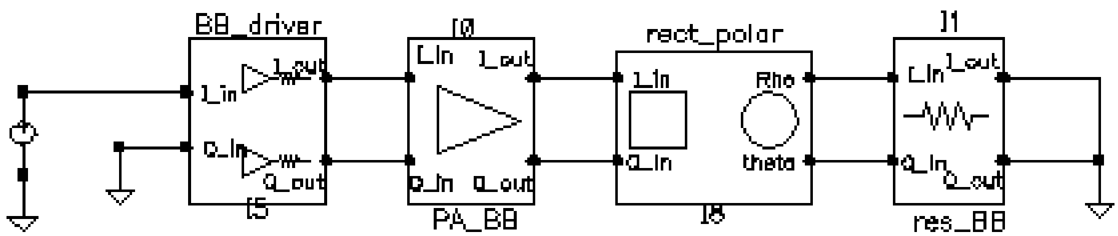
AM/PM Parameter	Definition
AM/PM Sharpness	Defines how steep the output phase shift changes are with respect to input power.
|radians|@1dB cp	Defines the absolute value of the output phase shift at the 1dB compression point for power amplifiers. This is the phase shift at an arbitrary output power level for some models.

Table D-4 AM/PM Conversion Parameters for Baseband Models

AM/PM Parameter	Definition
|radians|@big input	Defines the absolute value of the output phase shift as input power goes to infinity (if it could go to infinity)
{1, 0, -1} for {cw,none,ccw}	Defines the direction of the phase shift. 1 for clockwise, 0 for no phase shift, -1 for counter clockwise.

The test circuit in Figure D-38 on page 1136 is listed as *am_pm_test_ckt* in the *testbenches* category in *rfLib*.

Figure D-38 The am_pm_test_ckt Circuit

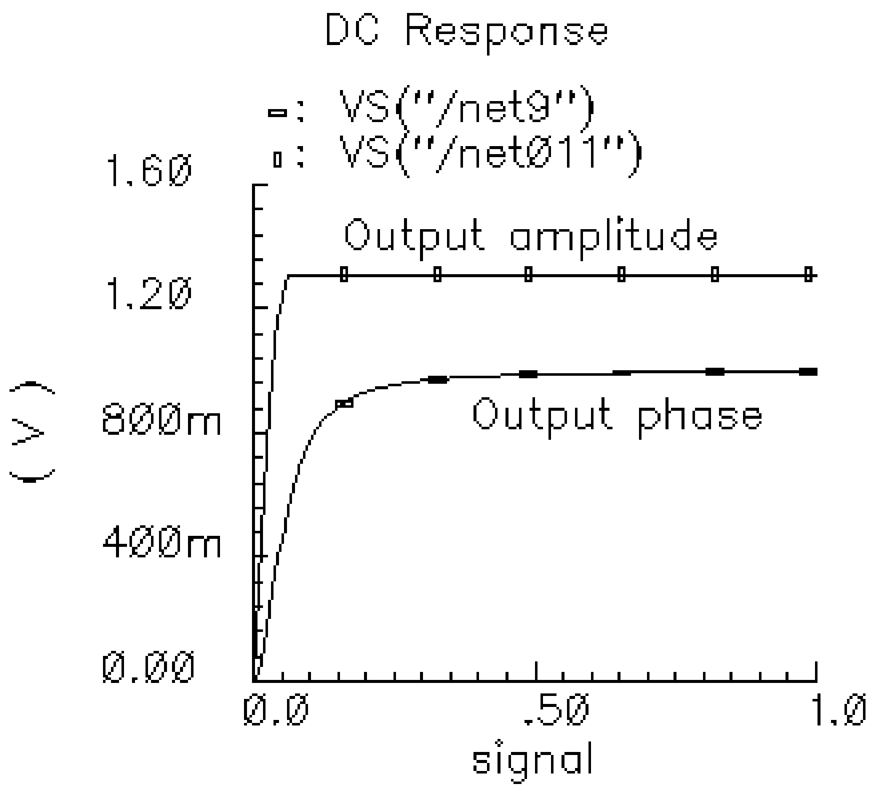


In the *am_pm_test_ckt* test circuit

- The first block (*BB_driver*) scales the control voltage generated by the leftmost element so that the output equals the specified dBm when the control voltage equals 1 volt. This is done so you can specify maximum dBm but still sweep linearly from zero signal.
- The second block (*PA_BB*) is a power amplifier.
- The third block (*rect_polar)* transforms the rectangular description of the baseband signal into polar coordinates so you can observe the phase shift and output signal level directly.

Figure D-39 on page 1137 shows the output amplitude and phase as functions of the input signal level. Generate these with a swept DC analysis. Sweep the *signal* variable from 0 to 1 in 200 linear steps and display the *rect_polar* outputs.

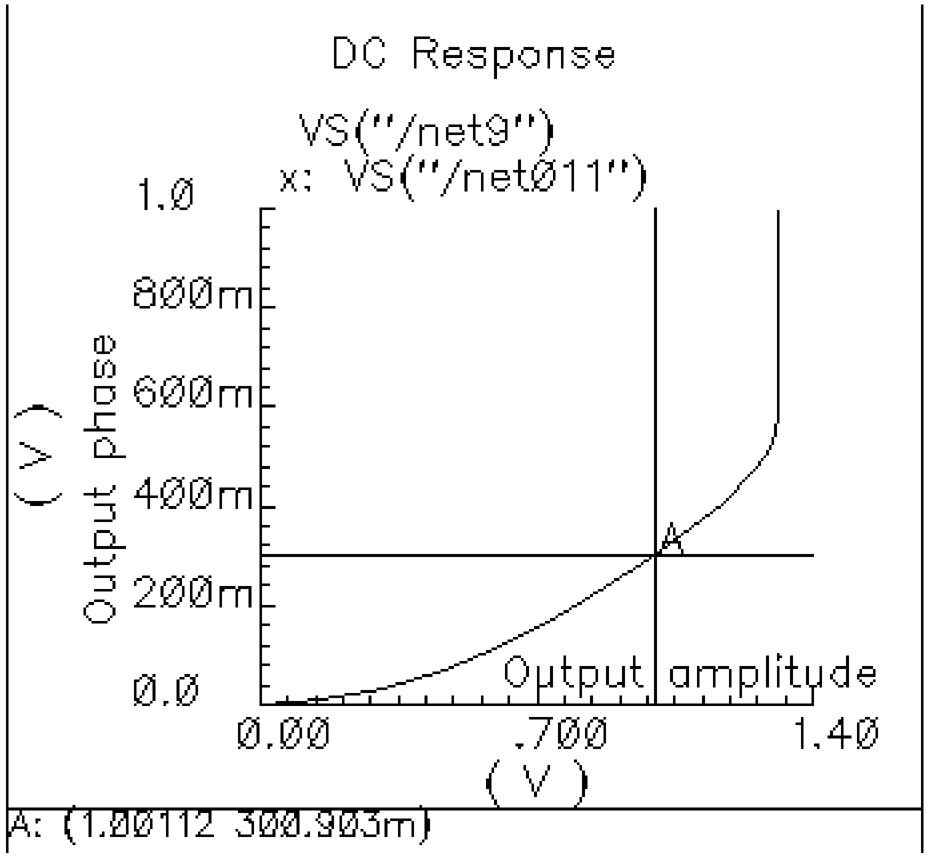
Figure D-39 Output Amplitude and Phase



By changing the x-axis to be the output amplitude trace, you can confirm that the phase shift at the output referred 1 dB compression point of 10dBm (or 1 volt peak across a 50 ohm load) equals 0.3 radians, as specified. Figure D-40 on page 1138 shows the plot.

Note that the measured power across the load is as specified only when the load matches the amplifier output resistance. If you mismatch the load you will not measure the specified phase shift at the specified output power level.

Figure D-40 Output phase versus output amplitude



In the next three figures, output phase is plotted against input signal level. Each plot shows the effect of one of the AM/PM conversion parameters. You can generate the plots by applying the Parametric Tool to the existing analysis.

Figure D-41 on page 1139 shows the effect of the |*radians*|@*1 db cp* parameter. Sweep `rad_cp` from 10m to 100m in 5 linear steps.

Figure D-41 Output Modified by the |radians|@1 db cp Parameter

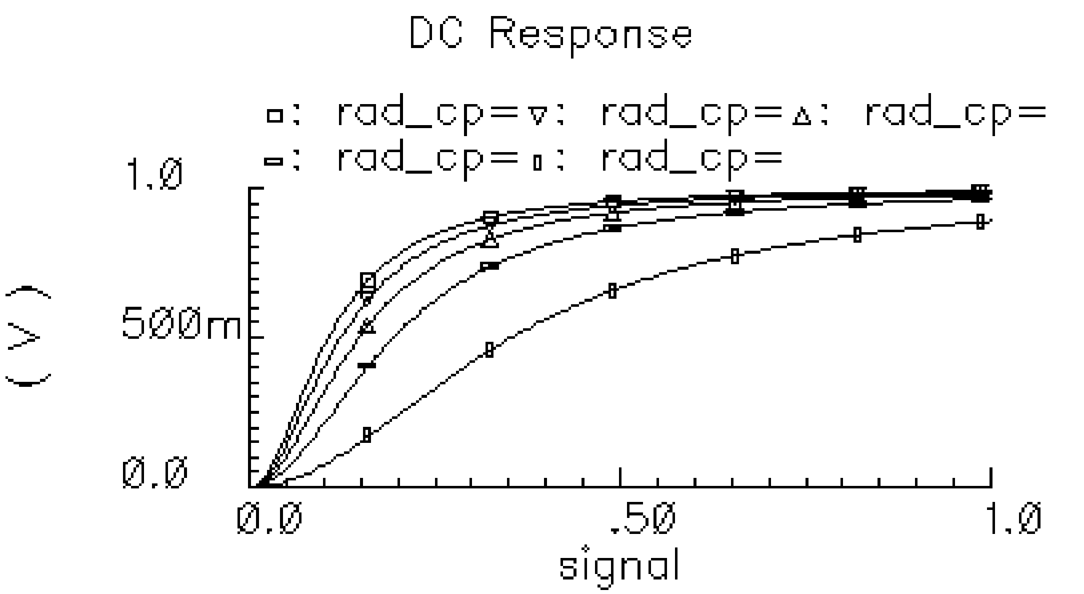


Figure D-42 on page 1139 shows the effect of the *am/pm sharpness* parameter. Sweep `sharpness` from 1 to 6 in 5 linear steps.

Figure D-42 Output Modified by the Sharpness Parameter

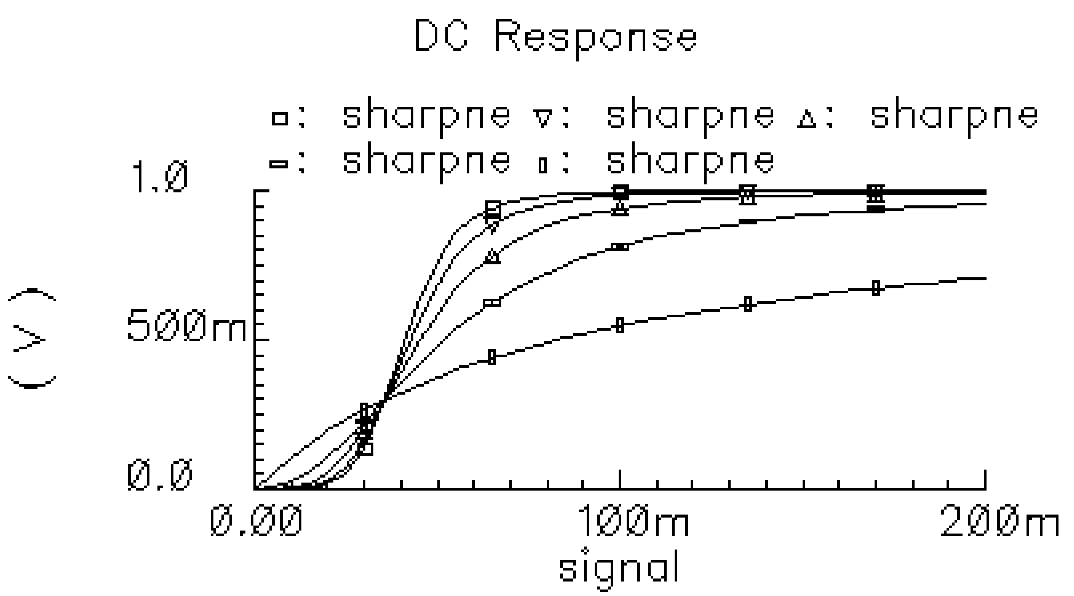
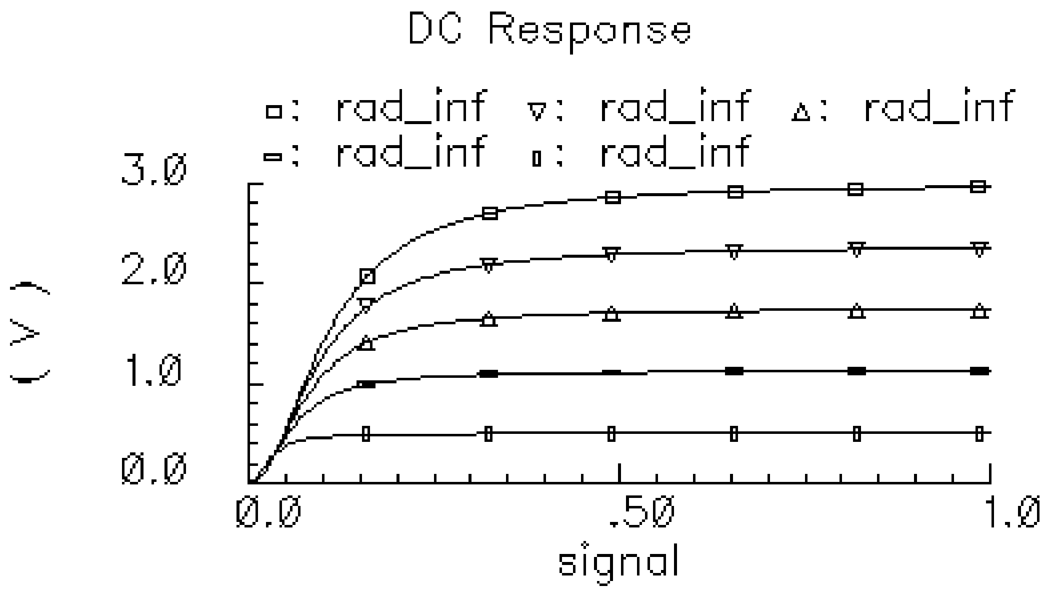


Figure D-43 on page 1140 shows the effect of the *rad_inf* parameter. Sweep `rad_inf` from 0.5 to 3 in 5 linear steps.

Figure D-43 Output Modified by the rad_inf Parameter



Library Description

The new models are listed in the table and described in the following sections.

Top-Down Model	Model Name in *rfLib*
Power Amplifier	*PA_BB*
Low Noise Amplifier	*LNA_BB*
I/Q Modulator	*IQ_mod_BB*
I/Q Demodulator	*IQ_demod_BB*
Upconverting Mixer	*up_cnvrt*
Downconverting Mixer	*dwn_cnvrt*
Inductor	*ind_BB*
Capacitor	*cap_BB*
Resistor	*res_BB*
Linear Time-Invariant Filters	
Butterworth Bandpass Filter	*BB_butterworth_bp*
Butterworth Bandstop Filter	*BB_butterworth_bs*
Butterworth Highpass Filter	*BB_butterworth_hp*
Butterworth Lowpass Filter	*BB_butterworth_lp*
Chebyshev Bandpass Filter	*BB_chebyshev_bp*
Chebyshev Bandstop Filter	*BB_chebyshev_bs*
Chebyshev Highpass Filter	*BB_chebyshev_hp*
Chebyshev Lowpass Filter	*BB_chebyshev_lp*
Baseband Shifter Combiner	*BB_shifter_combiner*
Baseband Shifter Splitter	*BB_shifter_splitter*
Rectangular-to-Polar Coordinate Transformation	*rect_polar*
Polar-to-Rectangular Coordinate Transformation	*polar_rect*
Ideal Transformer	*BB_xfmr*

Top-Down Model	Model Name in *rfLib*
Loss	*BB_loss*
Instrumentation Block	*comms_instr*
Offset Instrumentation Block	*offset_comms_instr*
Instrumentation Terminator	*instr_term*
Baseband Driver	*BB_driver*
Phase Generator	*phase_generator*

Notes on models involving frequency translation:

Passband models involving frequency translation have internal oscillators. When running a PSS or Transient analyses, be sure to make Maxstep small enough to prevent aliasing. Future releases of the models will include Boundstep lines to control maxstep from within the models. Envelope Following and QPSS analyses do not require a non-default maxstep because those analyses require external sources at the same frequencies that exist inside the VerilogA modules. That is because there is currently no way to name sources inside VerilogA modules.

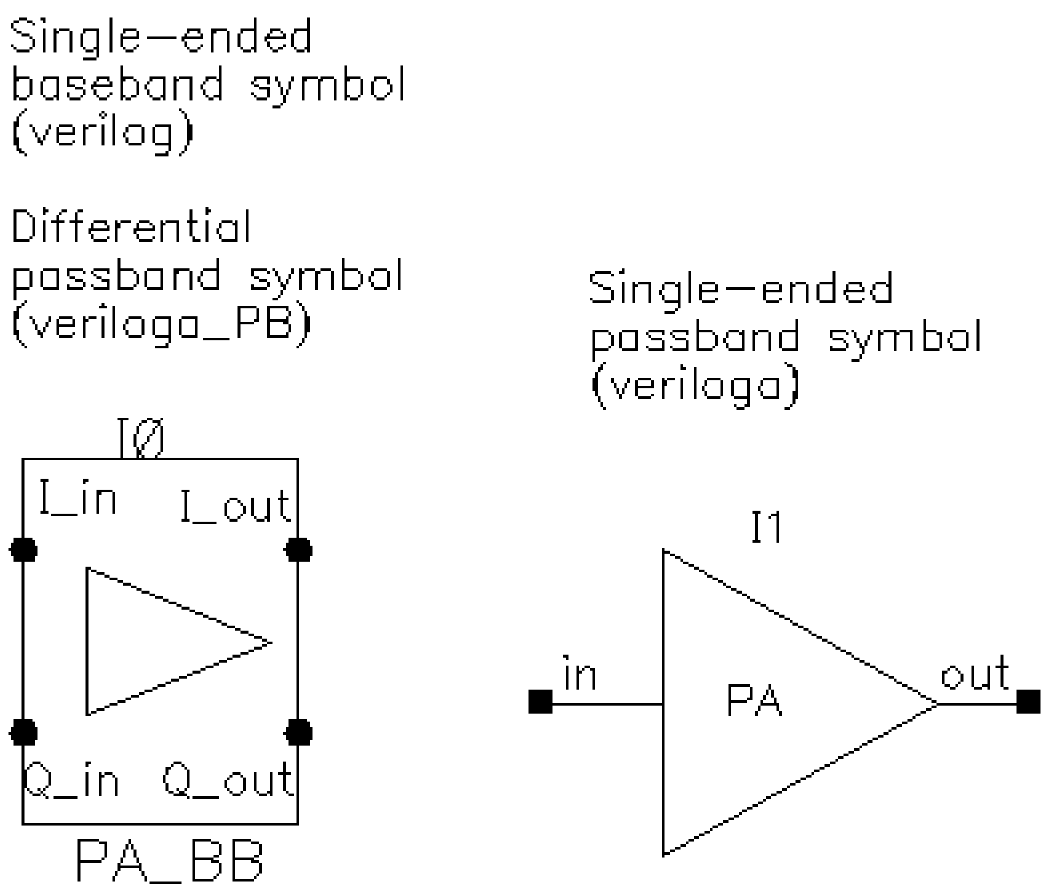
The local oscillators were absorbed into the mixer/modulator/demodulator models to maintain a close relationship between the baseband and passband models. With the ability to use an external source, it would be possible to drive a mixer with non-sinusoidal oscillator signals. Such signals introduce harmonics that the baseband models can not simulate. The first passband models encountered in the top-down flow, called level 1 passband models, are meant to introduce as few new effects as possible. The idea is simplify diagnostics by introducing new effects sequentially. The next level of detail in the behavioral passband models, i.e. level 2 models, are left to the user for now. Level 3 behavioral models are considered unnecessary because they do not offer a significant advantage over device-level models.

Top Down Baseband and Passband Models: top_dwnBB and top_dwn PB

Power Amplifier Model

`(baseband = PA_BB, passband = PA_BB)`

Figure D-44 Baseband and Passband Power Amplifier Models



The following parameters specify the power amplifier model.

Both Passband and Baseband models.

- available power gain

- input resistance
- output resistance
- output 1db cp in dBm
- noise figure

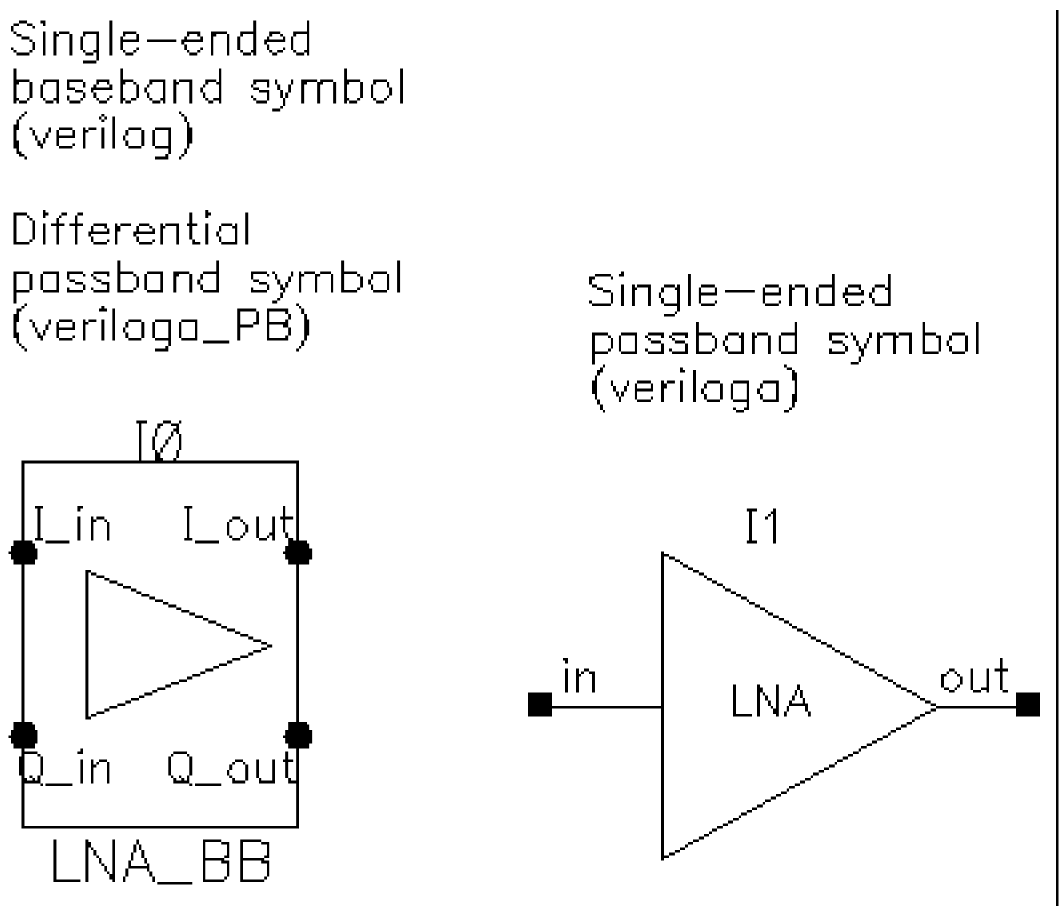
Baseband model only.

- am/pm sharpness
- |radians|@1db cp
- |radians|@ big input
- {1,0,-1} for {cw,none,ccw}

Low Noise Amplifier Baseband Model

`(baseband = LNA_BB, passband = LNA_PB)`

Figure D-45 Baseband and Passband Power Amplifier Models



The following parameters specify the low noise amplifier model.

Both Passband and Baseband models.

- available power gain
- input resistance
- output resistance
- input referred IP3[dBm]

- noise figure

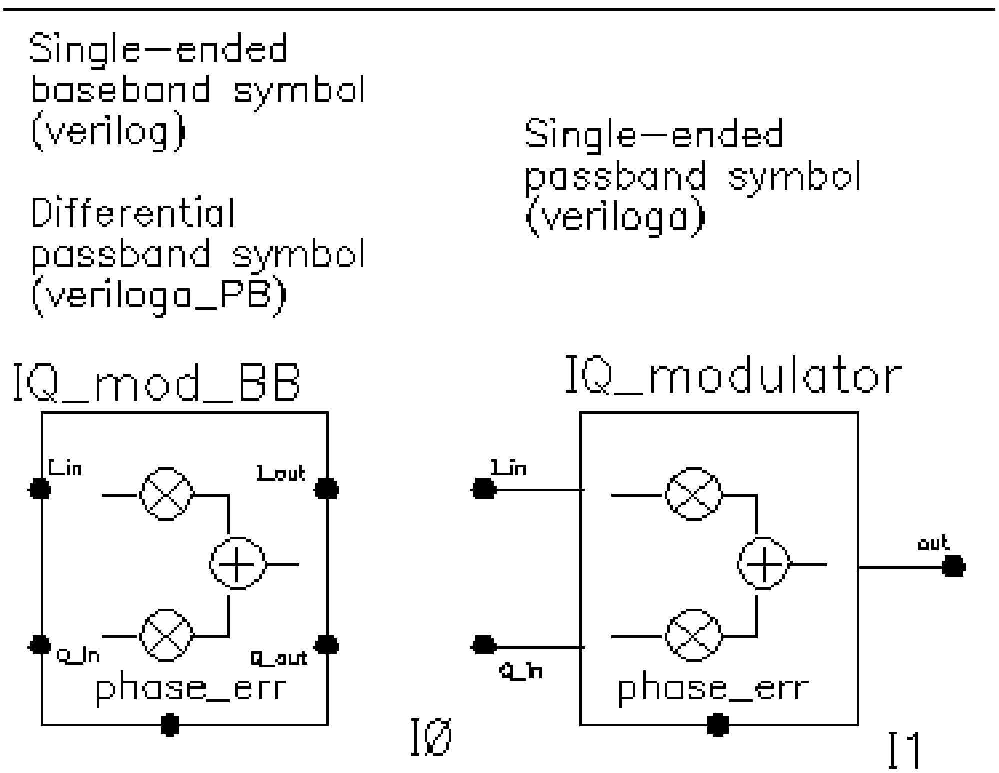
Baseband model only.

- am/pm sharpness
- cmp[dBm] = output power level where the next parameter is defined
- |radians|@cmp = output phase shift at output power of cmp.
- |radians|@ big input
- {1,0,-1} for {cw,none,ccw}

IQ Modulator Models

`(baseband = IQ_mod_BB, passband = IQ_modulator)`

Figure D-46 Baseband and Passband IQ Modulator Models



The IQ_modulator converts baseband signals to RF or IF. Figure D-47 on page 1149 summarizes exactly what the passband IQ modulator model does. The only difference between the baseband and passband models is carrier suppression. The non-linear functions, g_i and g_q, are specified by their available power gain and 1dB compression points just as in the power amplifier. The functions γ_i and γ_q characterize AM/PM effects in each mixer and are specified by the same parameters that specify power amplifier AM/PM conversion. Since noise is always added at the input, and the input is at baseband in this case, the noise sources are not doubled as they are in the power amplifier or LNA models. Noise figure is defined with reference to one input. Noise is injected at both inputs but the noise injected at just one input alone produces the specified noise figure. Thus, the noise figure parameter should be interpreted as noise figure per input. This model also includes a parameter called "quadrature error" which specifies how far away the two local oscillators signals are from being exactly in quadrature.

"Phase error" is the voltage on the phase error pin. The phase error pin has a fixed noiseless resistive input impedance of 50 ohms. The phase error pin can be used to introduce a dynamic phase error or phase noise. Phase noise can be fed into the phase error pin from a phase-domain PLL model or from a Port. Noise in Port models can be specified either by the internal resistance or by a data file that tabulates a power spectral density. The phase error pin can also be driven by a ramp or circular integrator output to model a frequency error between the incoming carrier and local oscillator.

The following parameters specify the IQ modulator. The available power gain and one dB compression point are explained first. The effects of the phase_error pin and the quadrature error parameter are discussed at the end of this section.

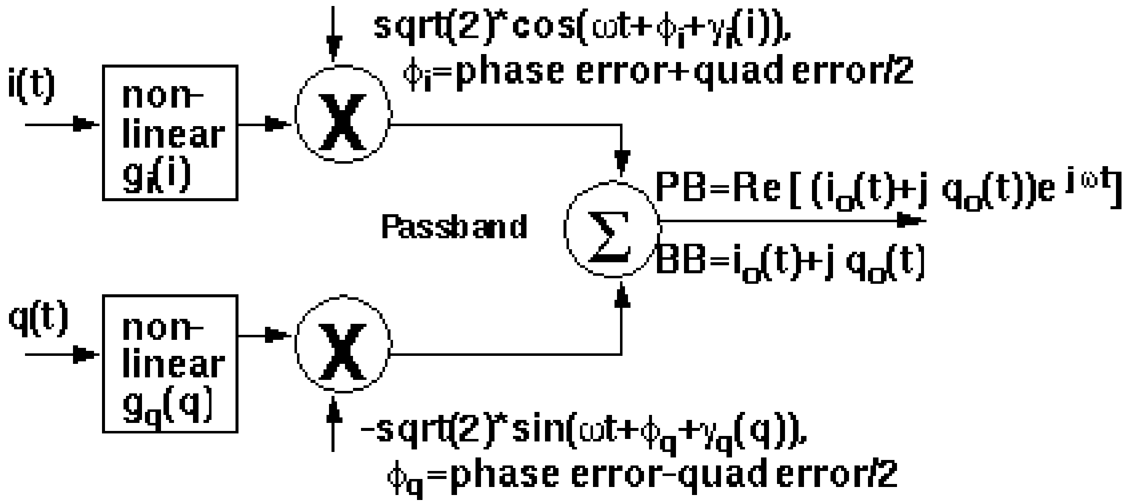
Both Passband and Baseband models.

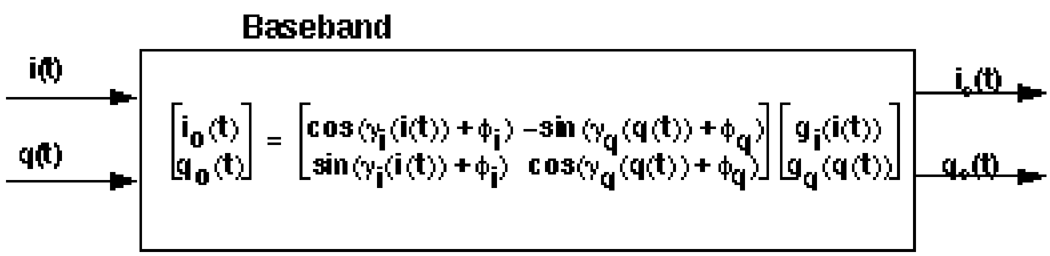
- available I-mixer gain[dB]
- available Q-mixer gain[dB]
- input resistance
- output resistance
- I-output 1dB CP[dBm]
- Q-output 1dB CP[dBm]
- noise figure [dB]
- quadrature error

Baseband models only.

- I-sharpness factor
- Q-sharpness factor
- Q-radians@Q_cmp
- I-radians@I_cmp
- I-radians@big I-input
- Q-radians@big-input
- {1,0,-1} for {cw,none,ccw}
- {1,0,-1} for {cw,none,ccw}

Figure D-47 IQ Modulator Calculations





Available power gain and 1dB compression point:

Available power gain of the IQ-modulator is best explained with an example. Recall the circuit called *mod_1dbcp* listed in the testbenches category of the rfLib. The schematic contains two disjoint circuits. One shows how not to measure gain and compression point, the other shows the proper measurement.

1. Set up a pss analysis. Both test circuits will run in the same simulation. The beat frequency will be 100MHz. Save the first and 11th harmonics. In the options, set maxstep to 50ps. Sweep the variable, `power`, linearly in 50 steps from -40 to 15.
2. After the analysis completes, plot the output referred 1dB compression point of the top circuit using -40 dBm as the Extrapolation point. First select the 11th harmonic (1.1GHz) and click on the output port in the top test circuit, the bad test circuit. Note that the linear gain is 3dB lower than specified, as is the output referred 1dB compression point. The gain was specified as zero dB and the 1dB compression point was 10 dBm. The error arises from the fact that the input signal power splits between upper (1.1GHz) and lower

(900MHz) sidebands but theanalog design environment measurement only looks at one output sideband. The bottom test circuit resolves the ambiguity by defining the gain of the IQ-modulator as the gain from the baseband input to an ideally-demodulated baseband output. The bottom test circuit follows the IQ-modulator with an ideal IQ-demodulator. The gain of the demodulator is zero dB and the 1dB compression point is high enough to render the demodulator distortionless.

3. Repeat the steps for plotting the 1dB compression point but this time chose the first harmonic and select the output port that loads the bottom circuit. Select the first (100MHz) harmonic and plot the 1dB compression point again. Now you should see a 1dB compression point plot that reflects the specified parameters of the IQ-modulator. The gain is now also correct, which can be computed from the ratio of the output to input power well below the compression point. Figure D-49 on page 1151 shows such a plot.

Figure D-48 1db Compression Point Test Circuit

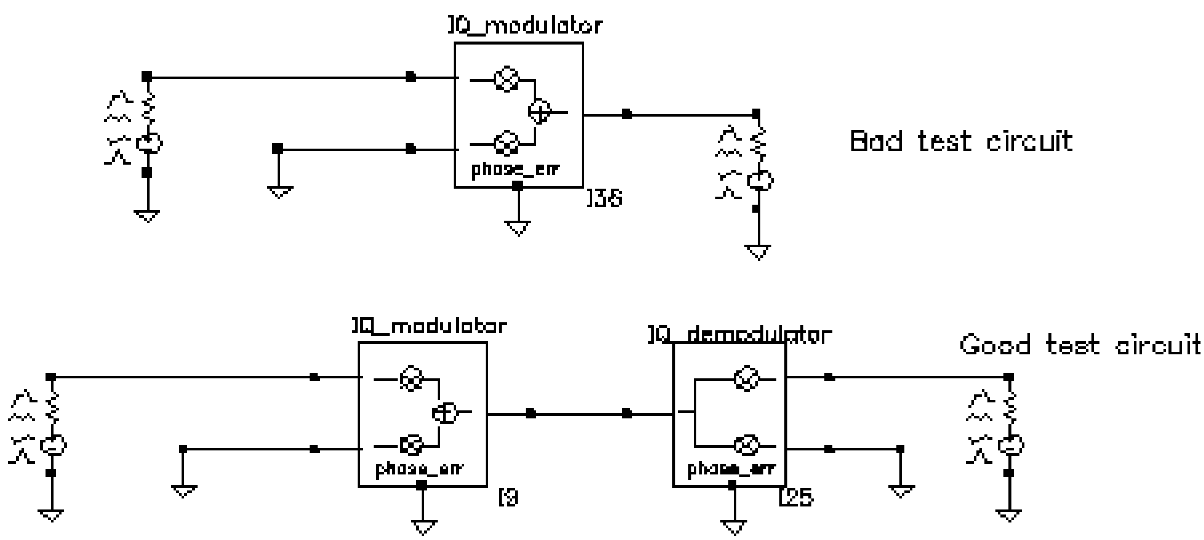
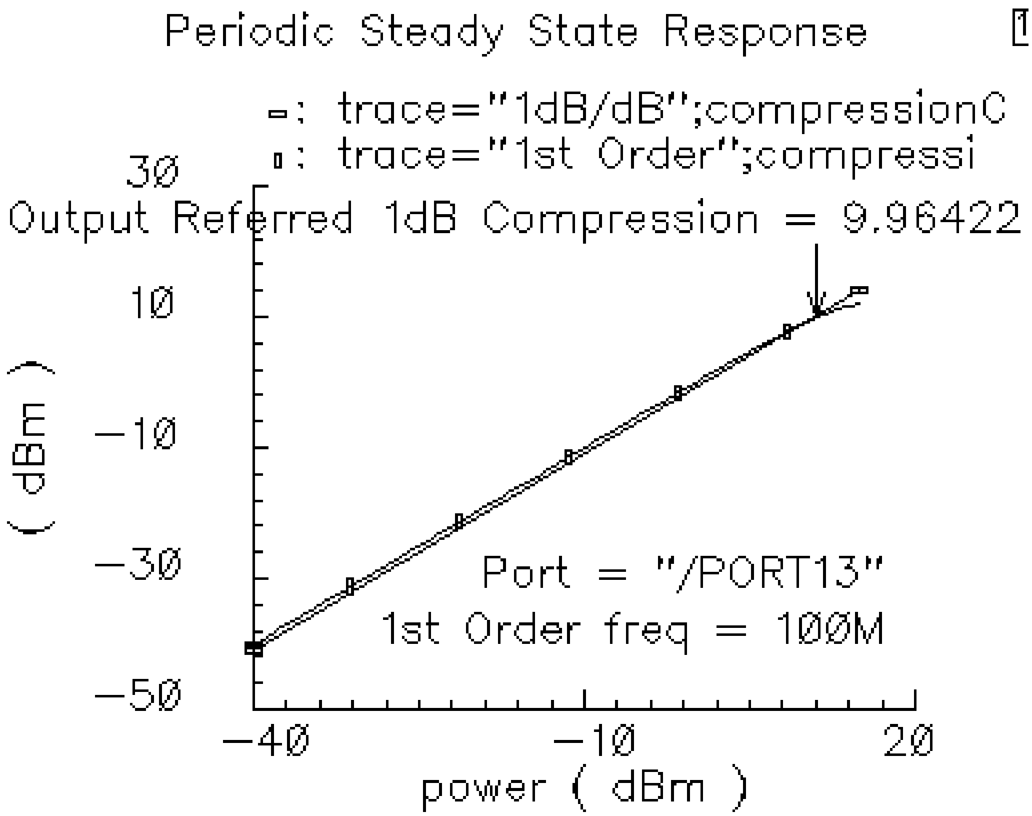


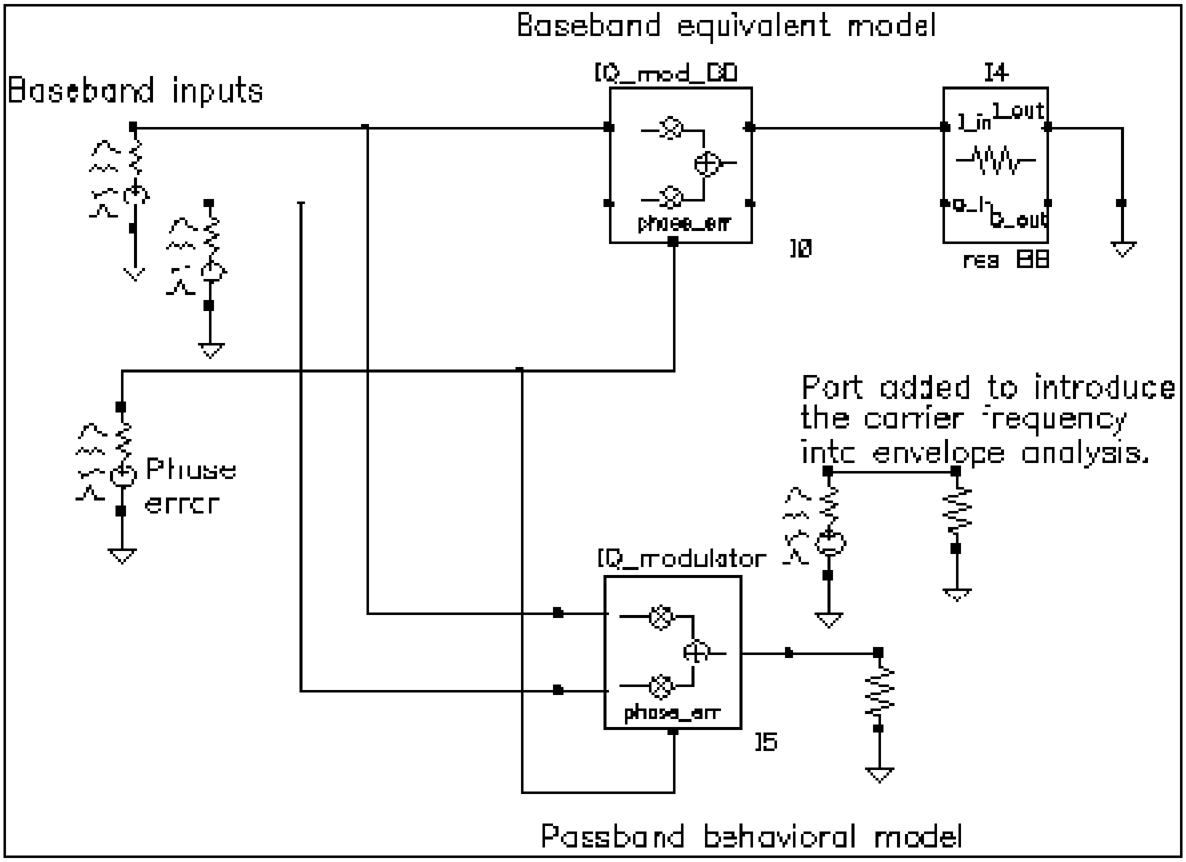
Figure D-49 1db Compression Point Plot



Quadrature error and phase error:

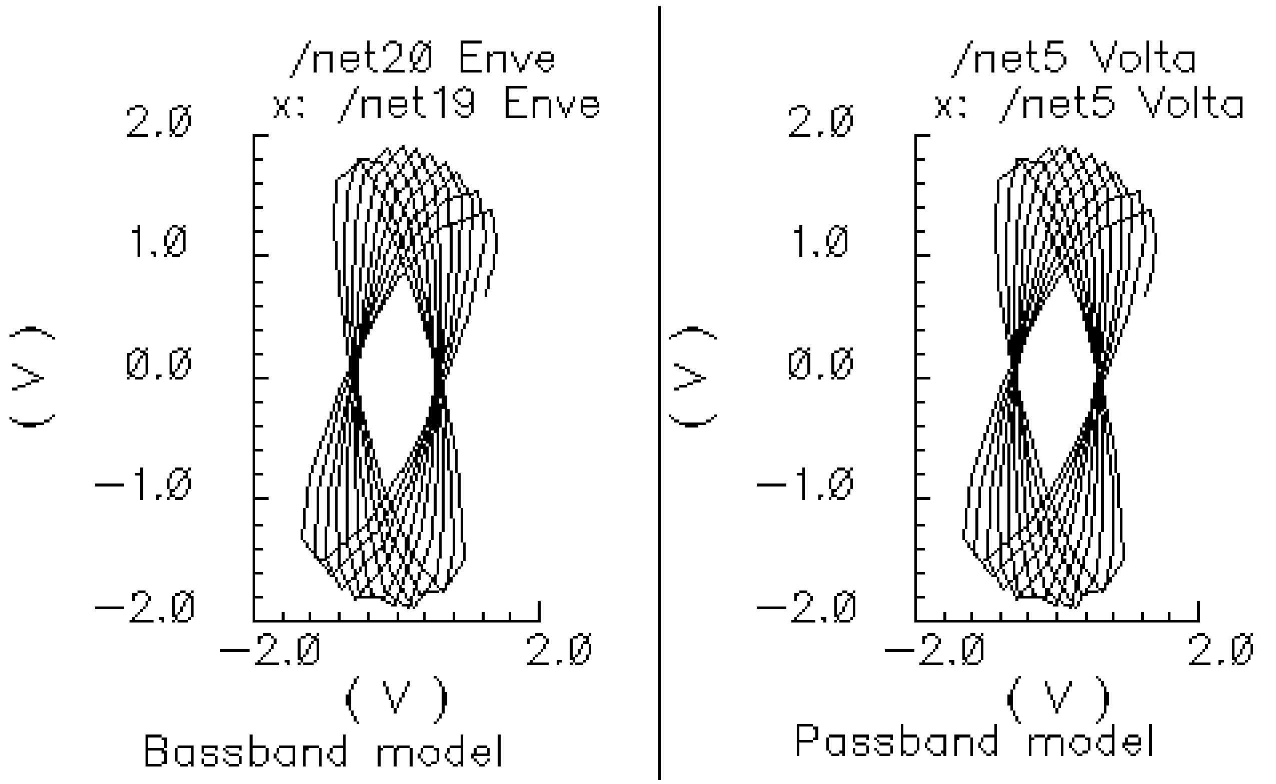
Quadrature error describes how far away from 90 degrees the two local oscillators are from each other. Ideally, they are exactly 90 degrees, or Pi/2 radians, apart in phase. In practice, parasitics and asymmetric delays can drive the phase shift away from Pi/2. Figure D-50 on page 1152 show a baseband test circuit and its passband equivalent. The schematic is listed in the rfLib *testbenches* category under the name *quad_and_phase_error_demo*. Both circuits are driven from a common set of baseband sources. The test circuit serves two purposes, it shows the correspondence between baseband and passband models and it demonstrates how quadrature error and phase error affect the baseband trajectory. The baseband input signal is a complex tone, which makes a circular input baseband trajectory. If there were no quadrature error, the baseband representation of the modulator output would also be a circle. With quadrature error, the output trajectory is an ellipse. If the phase_err pins are driven by a ramp, the ellipse precesses. The ramp represents a small but fixed difference between carrier and local oscillator frequencies. To see these effects:

Figure D-50 quad_and_phase_error_demo Circuit



1. Set up an Envelope analysis with `carrier` as the Clock Name. Simulate 10 us of action and save the first harmonic.
2. When the analysis completes, open the Envelope Following Results window and set the sweep to *time*. Plot the two outputs of the *IQ_mod_BB* block.
3. Change the x-axis to be the I-output. You should see the left trajectory in Figure D-51 on page 1153.
4. Add a subwindow for the passband equivalent result.
5. In the Envelope Results window, change the sweep to *harmonic time* and plot the real and imaginary parts of the first harmonic of the IQ_modulator output voltage.
6. Change the x-axis to be the real part of the first harmonic. You should now two pictures that match those in Figure D-51 on page 1153.

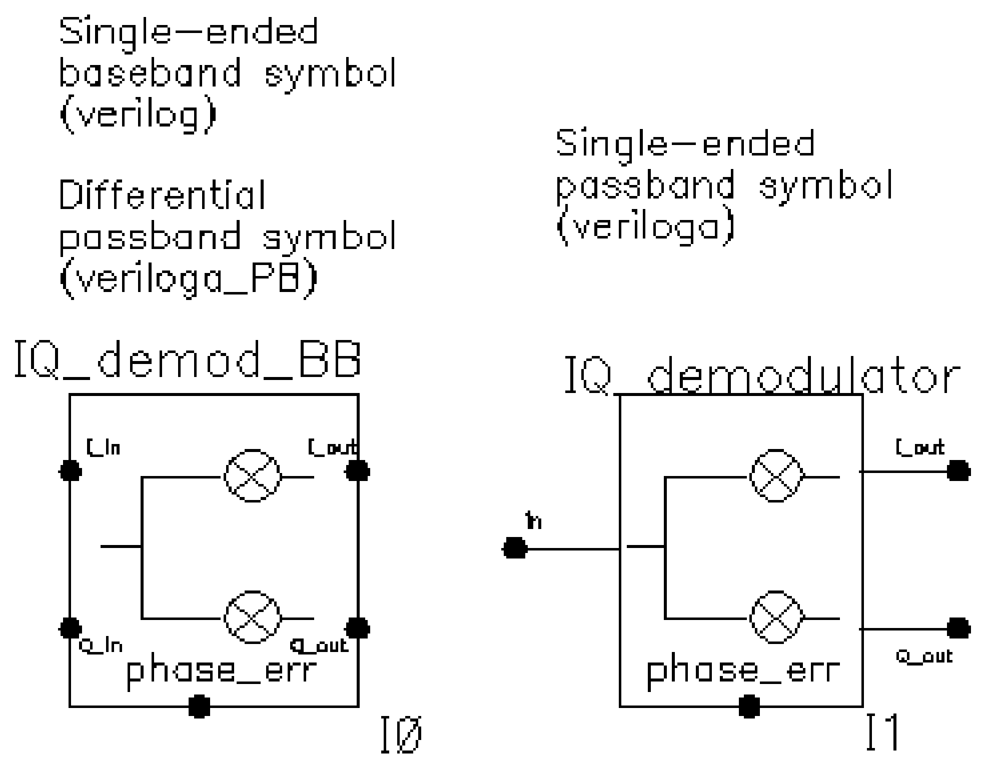
Figure D-51 Output Trajectories



IQ Demodulator

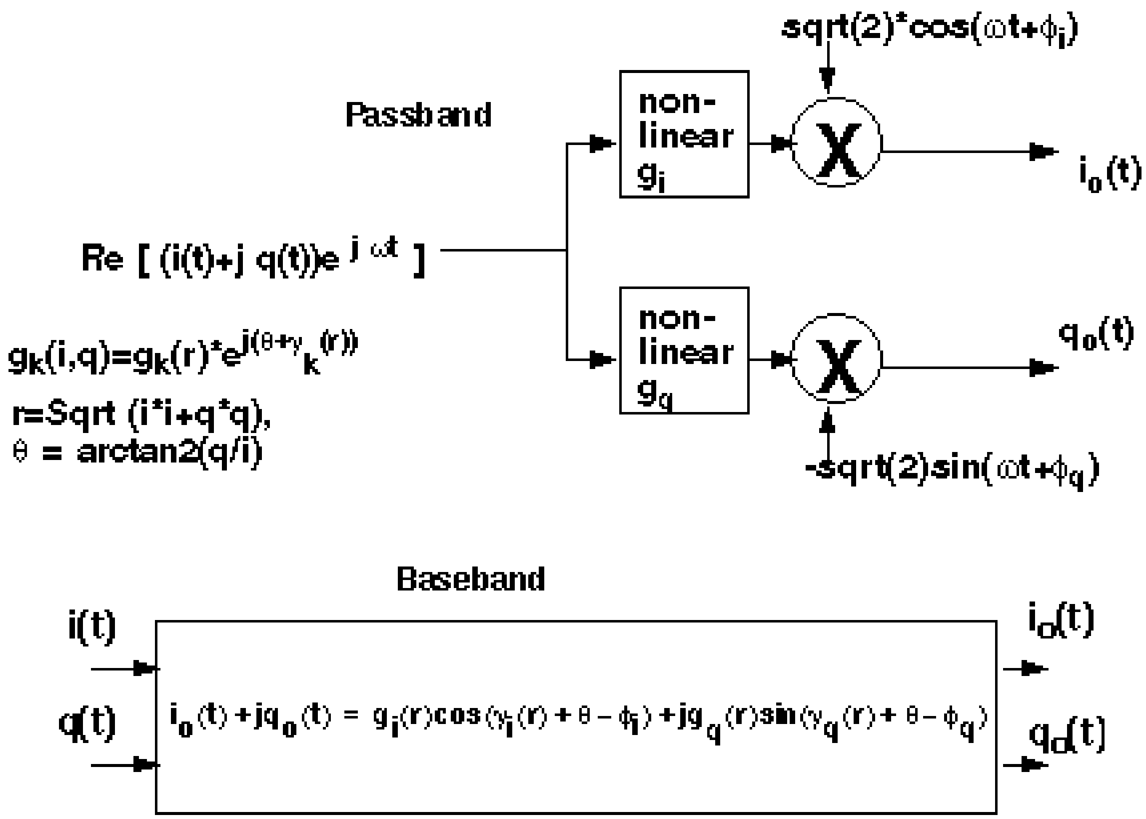
`(baseband = IQ_demod_BB, passband = IQ_demodulator)`

Figure D-52 Baseband and Passband IQ Demodulator Models



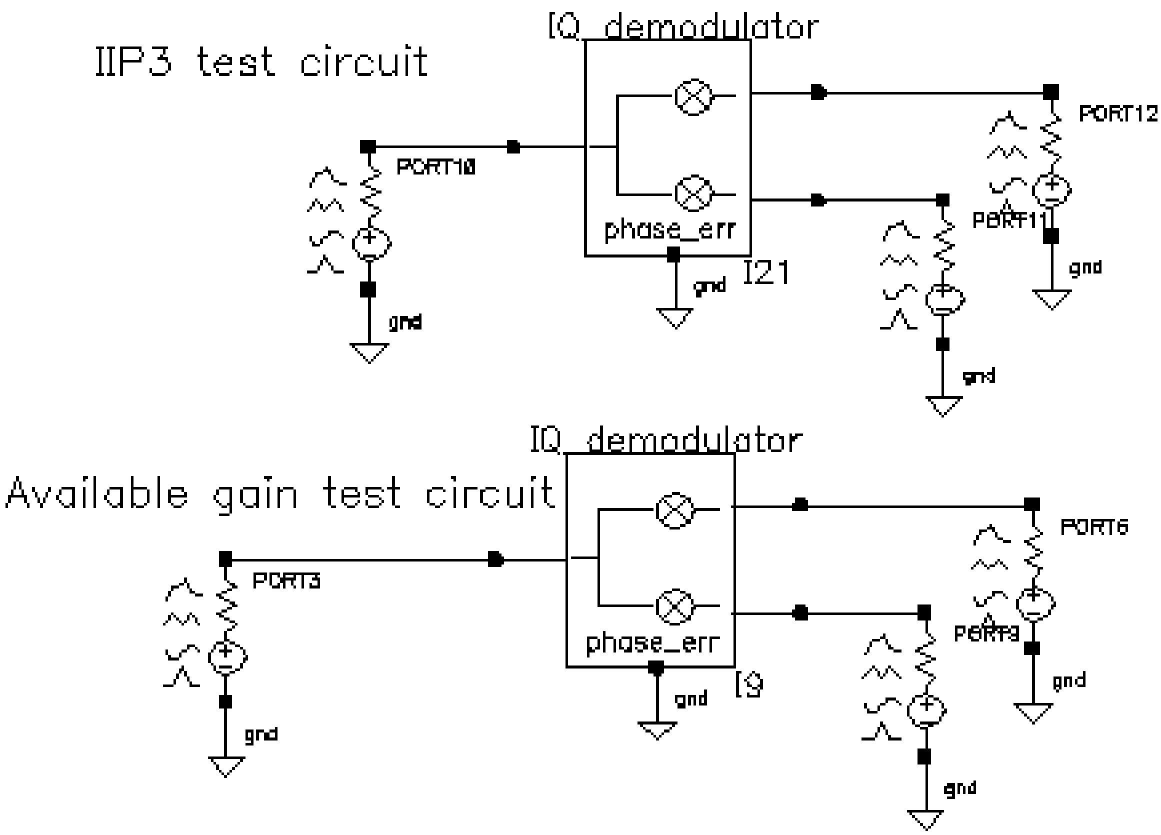
The IQ_demodulator converts RF (or IF) to baseband.Figure D-53 on page 1155 shows exactly what the passband demodulator model does. The parameters are like those in the modulator blocks except saturation is specified by input referred IP3 instead of the 1 dB compression point. IP3 was chosen over the one dB compression point for specifying saturation because the demodulator usually lies in the receive path and receiver blocks are usually specified with IP3.

Figure D-53 IQ Demodulator Calculations



The circuit called demod_ip3 in the testbenches category of the rfLib shows how the gain and IP3 parameters are defined. Figure D-54 on page 1156shows the schematic. Both the input and the output resistances are matched.

Figure D-54 The demod_IP3 Schematic



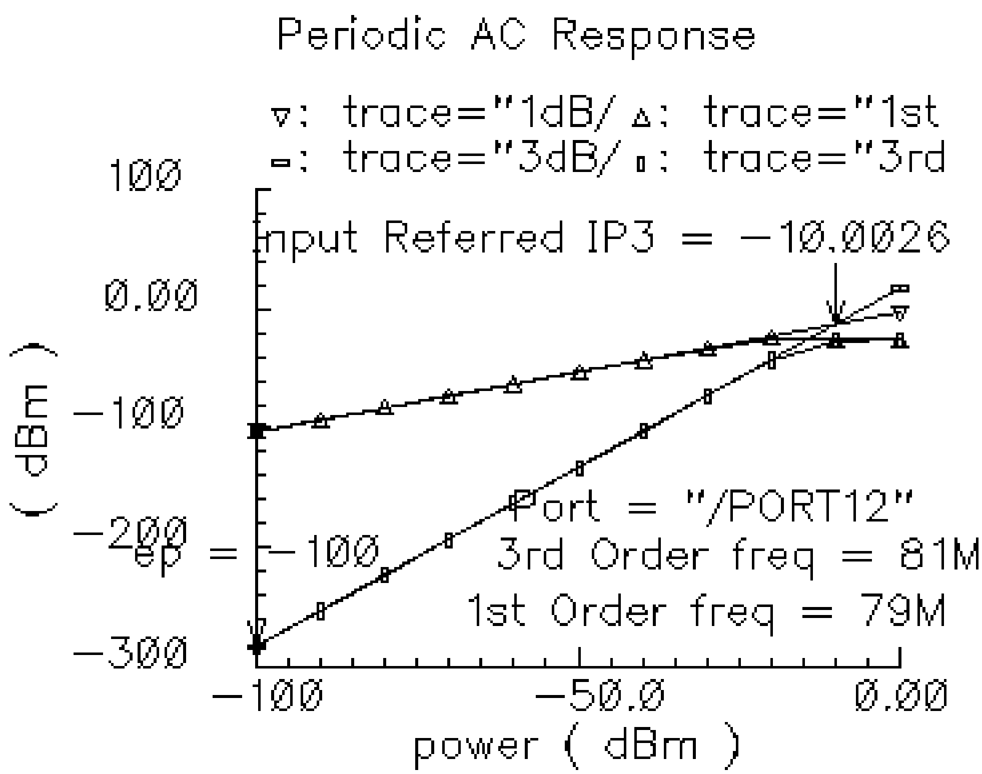
1. Recall the circuit and set up a swept PSS analysis. Let the Beat Frequency be Auto Calculated. Keep 2 harmonics. Sweep the *power* parameter from -100 to 0 in 10 linear steps.

2. Set up a single point PAC analysis at 921MHz and keep the -25 and -21 sidebands.

3. After running the analysis, from the PAC output window plot the input-referred IP3 curves with 81MHz as the 3rd order sideband and 79MHz as the 1st order sideband. The procedure is similar to mixer IP3 example covered in the last section. Use *Variable Sweep* for the Circuit Input Power and -100 for the Extrapolation point. Make sure to plot Input Referred IP3. Click on the I-output port in the top circuit. You should see -10dBm as the IP3, just as specified. Figure D-55 on page 1157 shows the IP3 plot. Note that 1st order line indicates the gain is 3dB below the specified gain of 0dB. That is because not all of the power lies at 1000MHz-921MHz = 79MHz; Some of the power lies at 1000MHz + 921MHz = 1921MHz. Use the bottom test circuit to measure available power gain. The bottom circuit drives the demodulator at the same frequency as the demodulator's

internal local oscillator, which runs at 1GHz. Now the output power is not split, it lies in the zero harmonic of the I-output.

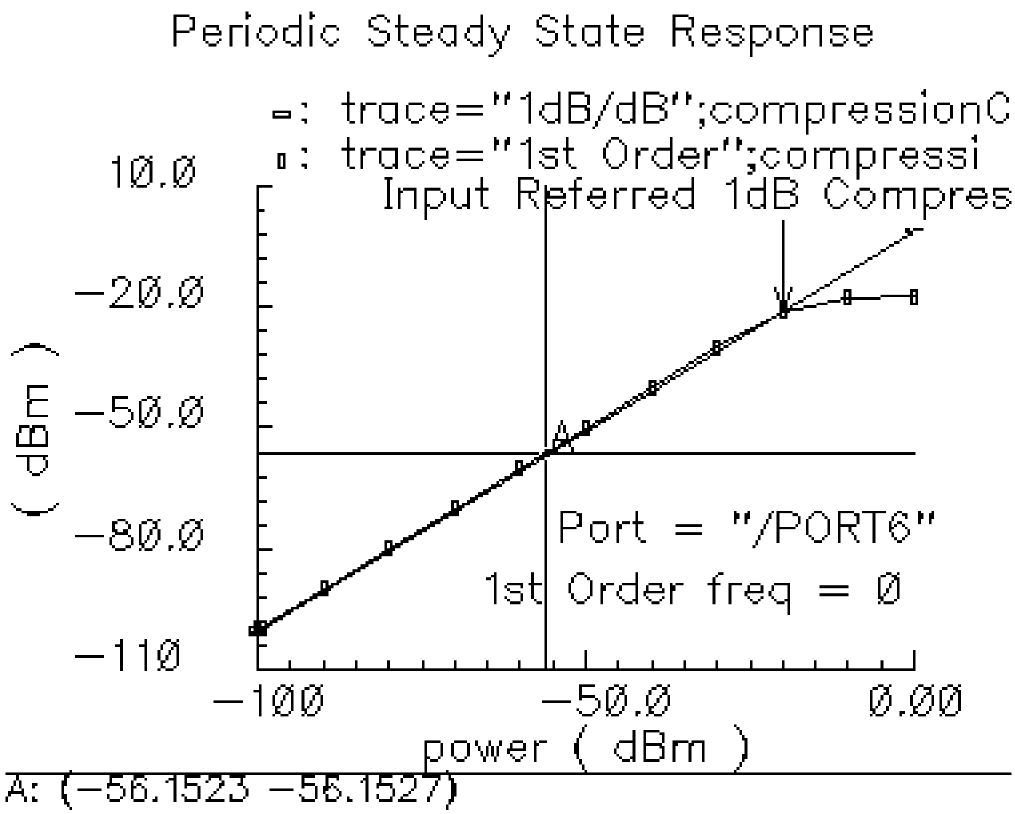
Figure D-55 Demodulator IP3



4. Plot the 1dB compression point at the port loading the I-output of the bottom circuit. Use the zeroth harmonic. The ratio of output to input power should be unity in the linear region. Figure D-56 on page 1158 shows the compression point plot. The measured 1dB compression point is of no use in this test. We want the gain. At low power levels where the gain is constant, the gain is as specified.

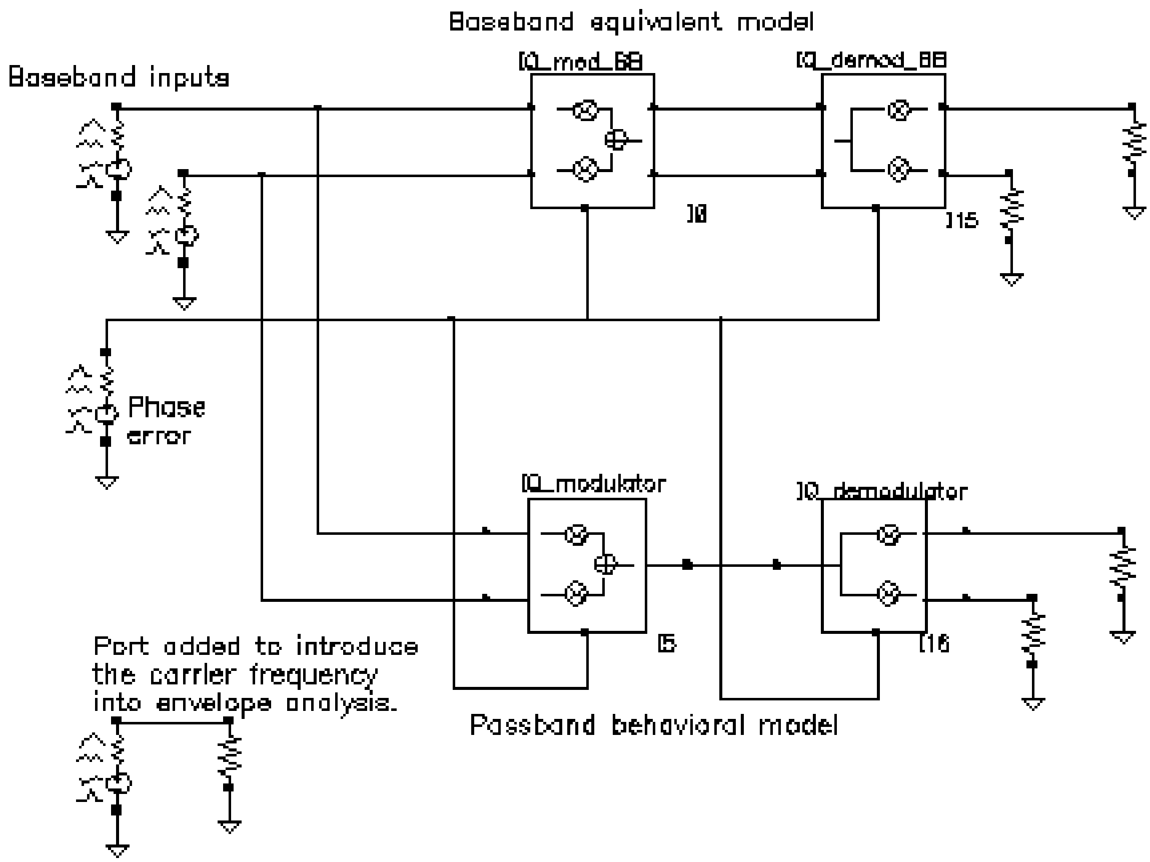
   Remember, in this test circuit the load resistance and output resistance are equal so that the output power is maximal. Also, the input resistance equals the source resistance so that the horizontal axis truly equals input power.

Figure D-56 Demodulator available power gain



Phase errors behave like their counterparts in the modulator models except for a change of sign. Quadrature error behaves exactly as it does in the modulator models. Figure D-57 on page 1159 shows a test circuit for illustrating the relationships between phase error and quadrature error in the modulators and demodulators. The test circuit is called *mod_demod_test* and is listed in the *testbenches* category. The test circuit also shows that the passband and baseband models give comparable results, as they should, as long as the passband carrier is not severely clipped. The baseband input trajectory is a complex 1MHz tone, which produces a circular input trajectory. The demodulator outputs are not matched and are not symmetric with respect to I and Q paths. The modulators and demodulators are not perfectly linear and the non-linearities are asymmetric with respect to I and Q. The modulators and demodulators are driven by the same phase error and the quadrature error parameters are a common variable set to 0.785 radians.

Figure D-57 mod_demod_test Circuit



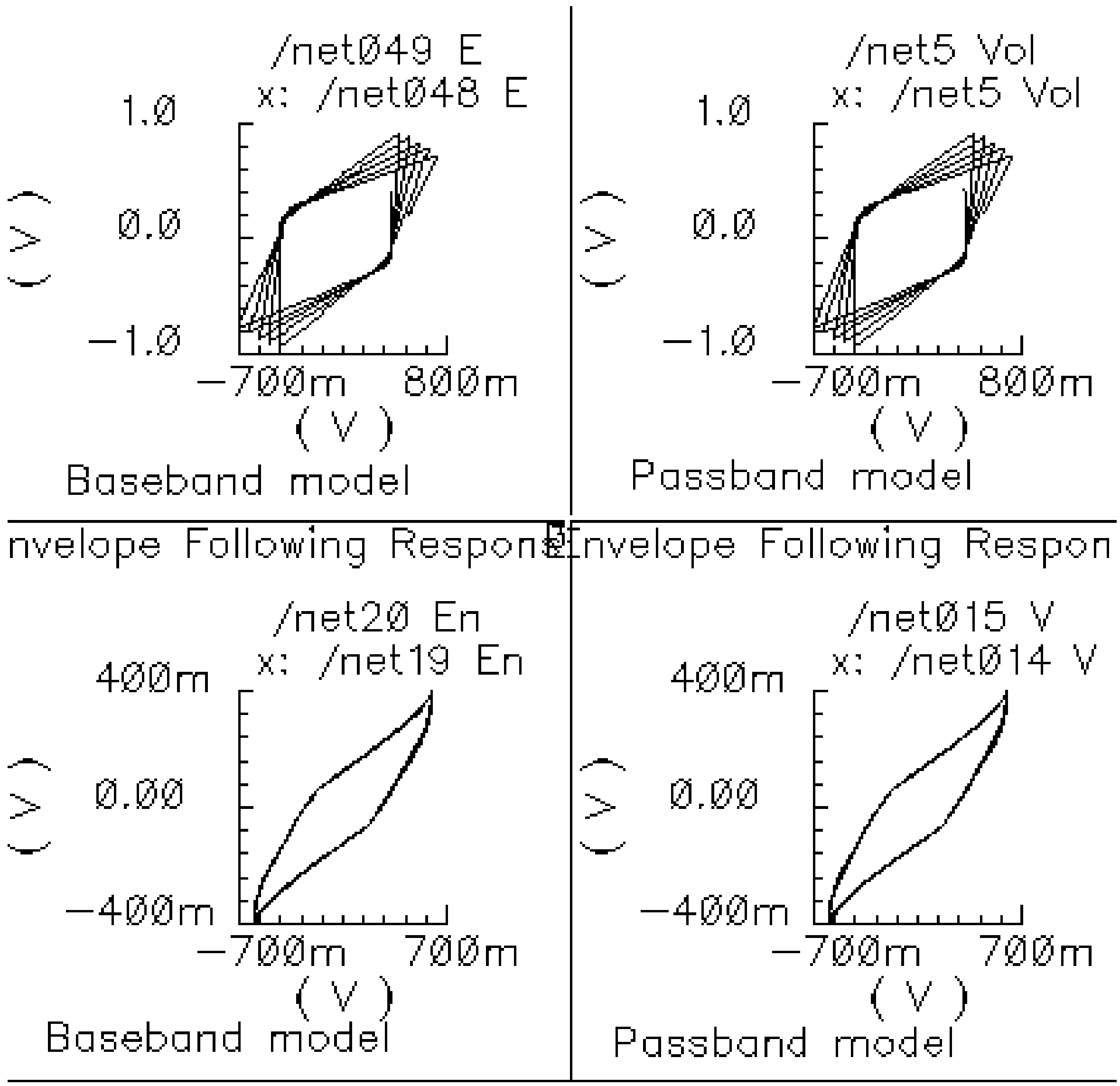
To use the circuit:

1. Recall the circuit and set up a 5us Envelope Following analysis with “carrier” as the Clock Name.
2. After the analysis completes, plot the IQ_mod_BB outputs and make the I_out signal the x-axis.
3. Open a subwindow and in it, plot the harmonic time waveforms of the IQ_modulator output. Use the first harmonic and plot the real and imaginary waveforms. Make the real waveform the x-axis.
4. Open a third subwindow and stretch the Waveform Display window so that the third subwindow appears below the first window.
5. Plot the time waveforms at the IQ_demod_BB outputs and make the I_out waveform the x-axis.

6. Open a fourth subwindow and plot the harmonic time results at the IQ_demodulator outputs but this time use the zeroth harmonic and only plot the real parts. Make the I_out waveform the x-axis. Figure D-58 on page 1161 shows what you should now see.

The leftmost pictures are from the baseband models and the rightmost are from the passband models. Passband and baseband models agree quite well. The top pictures are the voltages at nodes that lie between the modulator and demodulator. Quadrature error squashes the baseband trajectory at that node. The trajectory precesses because phase error ramps up linearly with time just like in the last test. The non-linearities produce the sharp corners. The bottom trajectories do not precess because the same phase error rotates the demodulator output in the reverse direction; driven by the same phase error ramp, the demodulator undoes the precession introduced in the modulator. The demodulator outputs are nearly in phase because the quadrature errors of pi/4 in the modulators and demodulators add to give a total quadrature error of Pi/2, which in this case puts the baseband I and Q outputs nearly in phase with each other.

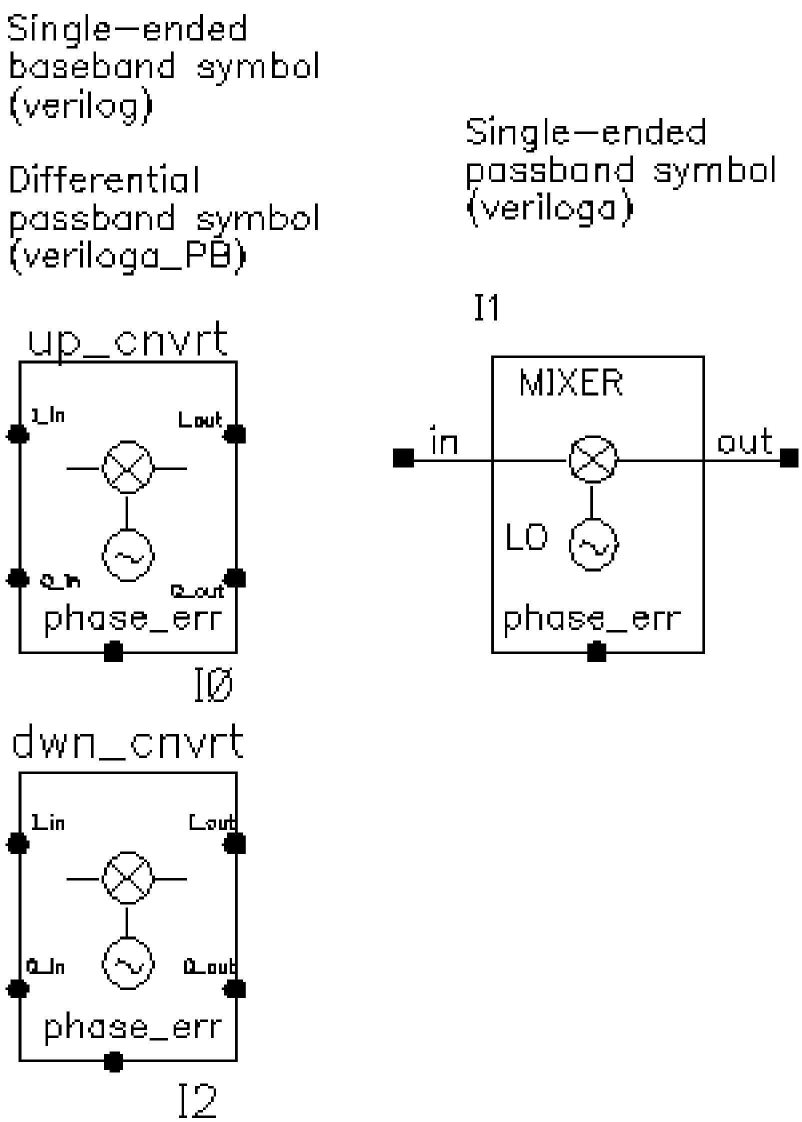
Figure D-58 mod_demod results



RF-to-IFand IF-to-RF Mixers

`(passband = MIXER_PB, baseband = dwn_cnvrt and up_cnvrt)`

Figure D-59 Baseband and Passband Mixer Models



MIXER_PB is a passband model that converts RF to IF and IF to RF. dwn_cnvrt model is a baseband equivalent model of a mixer used to convert from RF to IF. up_cnvrt model is a baseband equivalent model of a mixer used to convert from IF to RF. There are some minor differences in the baseband models that depend on whether conversion is up or down. Figure D-60 on page 1163 and Figure D-61 on page 1163 show what the models do.

Figure D-60 Calculations for up_cnrt Mixer

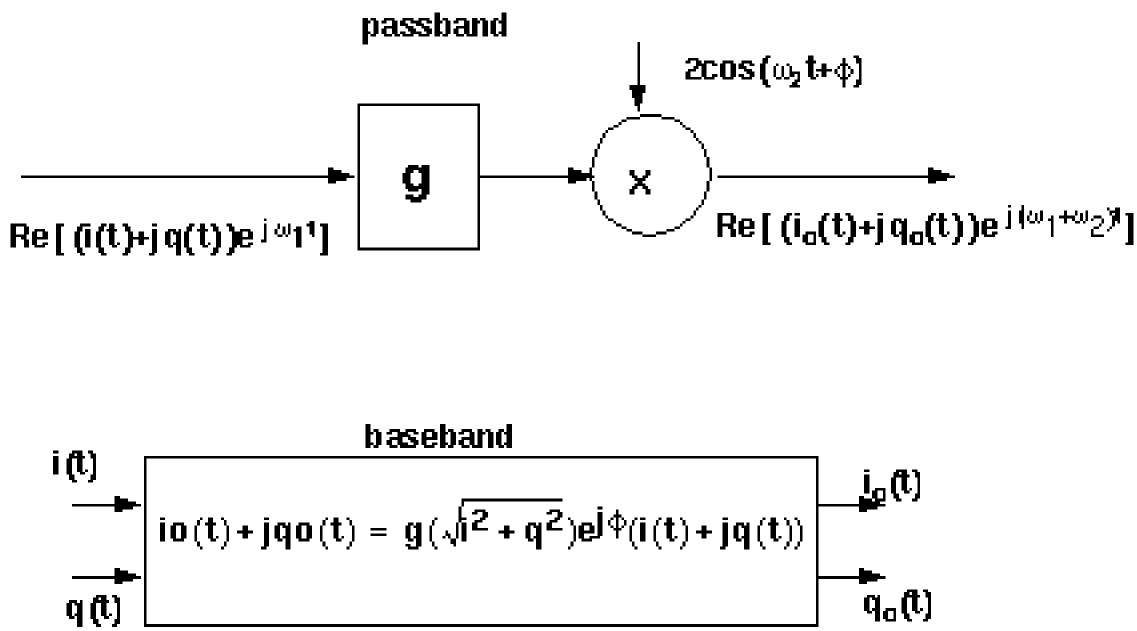
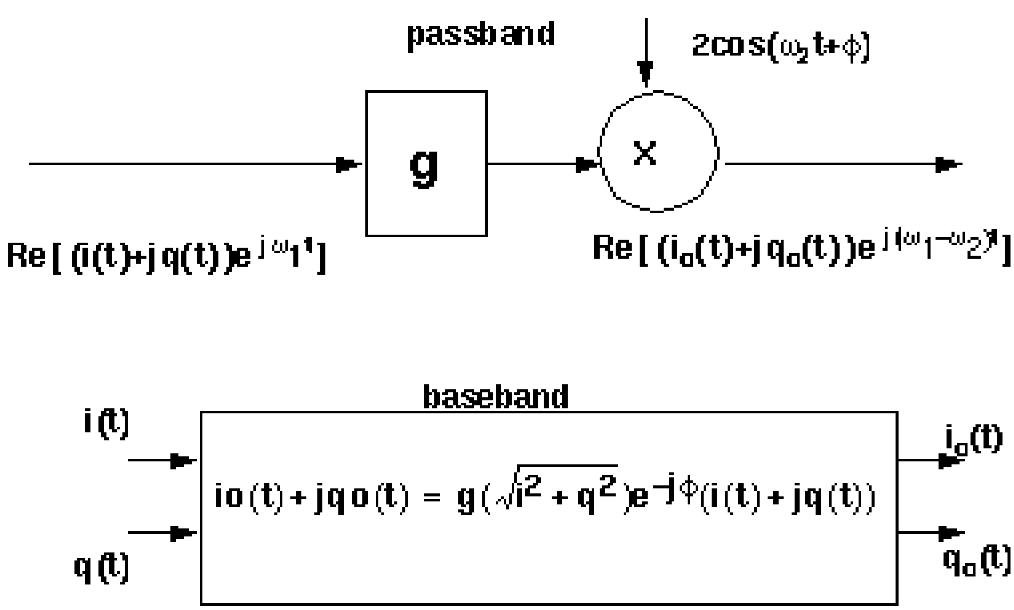


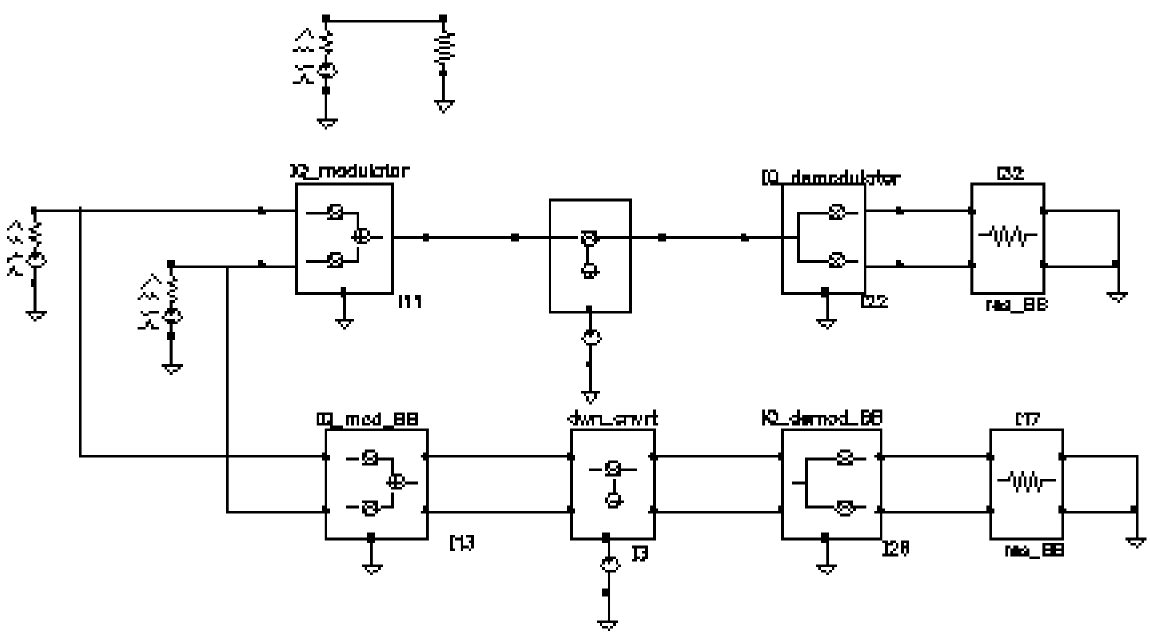
Figure D-61 Calculations for dwn_cnrt Mixer



The noise figure and IP3 parameters are defined in "Some Common Model Parameters" on page 1121. Unlike the IQ_demodulator, the IP3 test circuit can be used to define the available power gain because the gain is defined from the input frequency to just one sideband.

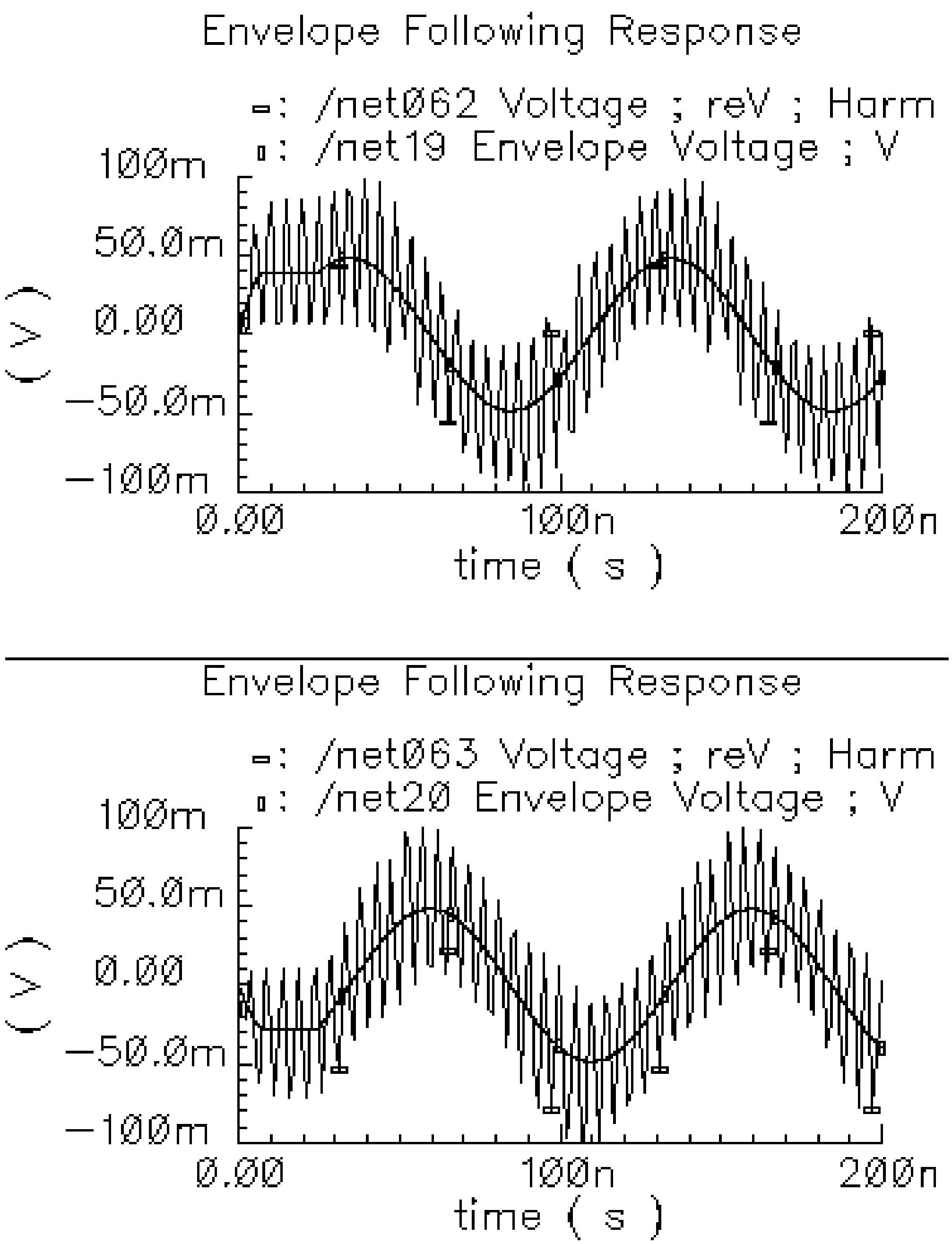
Typically the mixer would be used to create an IF stage. In that case, it is difficult to obtain a simple (i.e. filterless) envelope following analysis that overlays waveforms to show how well baseband and passband models agree. The test circuit shown in Figure D-62 on page 1164, which is listed as *dwn_cnvt_test* in the *testbenches* category of the rfLib, shows the relationship between baseband and passband models. The top branch of the circuit consists of passband models. The bottom branch consists of baseband models. To see the relationship:

Figure D-62 dwn_cnrt_test Circuit



1. Recall the circuit and set up a 200ns envelope following analysis with `fclck` as the *Clock Name*. Keep 1 harmonics1.
2. After the analysis completes, plot the "time" waveform at the I_out pin of the IQ_demod_BB model. Append to the plot, the harmonic-time, real part of the zero harmonic of the I_out pin on the IQ_demodulator model.
3. Open a subwindow and do the same for the Q outputs. You should now see a picture like the one in Figure D-63 on page 1165.

Figure D-63 Output from an Envelope Following Analysis



Let's trace the input signal through the passband branch to understand these results. A complex baseband 10MHz tone drives both branches. The modulator's local oscillator is 1GHz so that the IQ_modulator output is at 1.01GHz. There is no 990MHz sideband because the input baseband trajectory is a circle (= sin + jcos), which represents a complex tone. The mixer local oscillator is 900MHz, which when mixed with 1.01GHz, produces 110MHz and 1.91GHz. The IQ_demodulator local oscillator is 100MHz, which produces 10MHz, 210MHz, 2.01GHz, and 1.81GHz. The 10MHz and 210MHz terms dominate the zero harmonic at the demodulator outputs. The higher frequencies average out to nearly zero. The baseband output is the 10MHz term and that is what the baseband branch generates, as shown in Figure D-63 on page 1165. A Transient analysis actually runs about 13 times faster than envelope on this circuit. Figure D-65 on page 1167 compares the same outputs using a Transient analysis.The Transient analysis shows that the zero harmonic of the envelope analysis averaged out all frequencies above the envelope clock frequency (1GHz).

Testing the up_cnvrt Mixer

There is a test circuit for the up_cnvrt model similar to the test circuit containing the dwn_cnvrt model. The up_cnvrt model is called up_cnvt_test and is shown in Figure D-64. It is also in the testbenches category of rfLib. The steps parallel those for the dwn_cnvrt model.

Figure D-64 up_cnvt_test Circuit

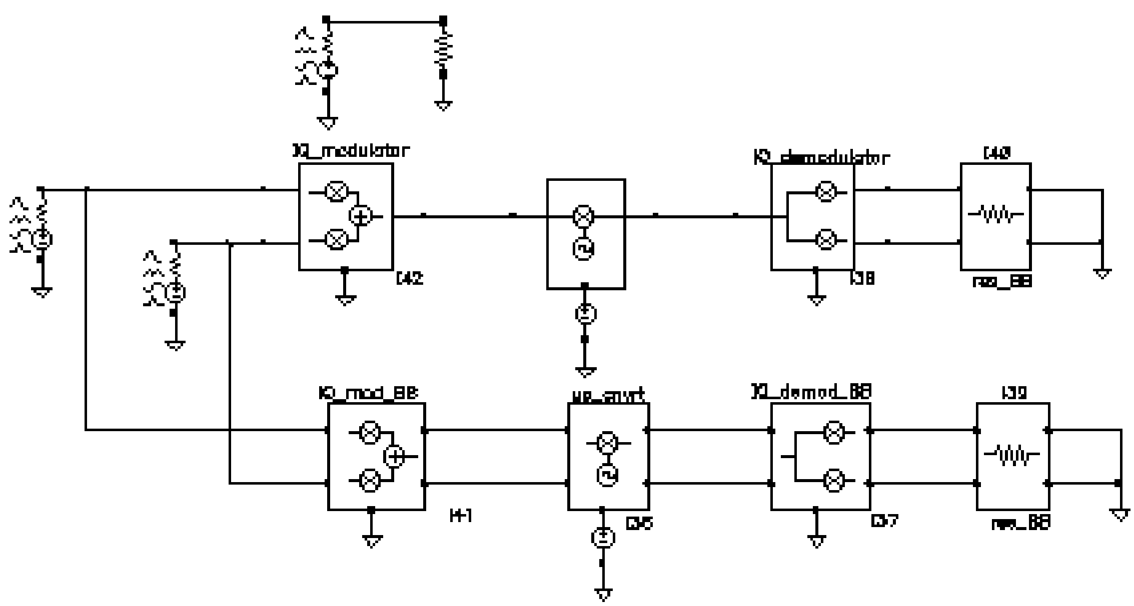
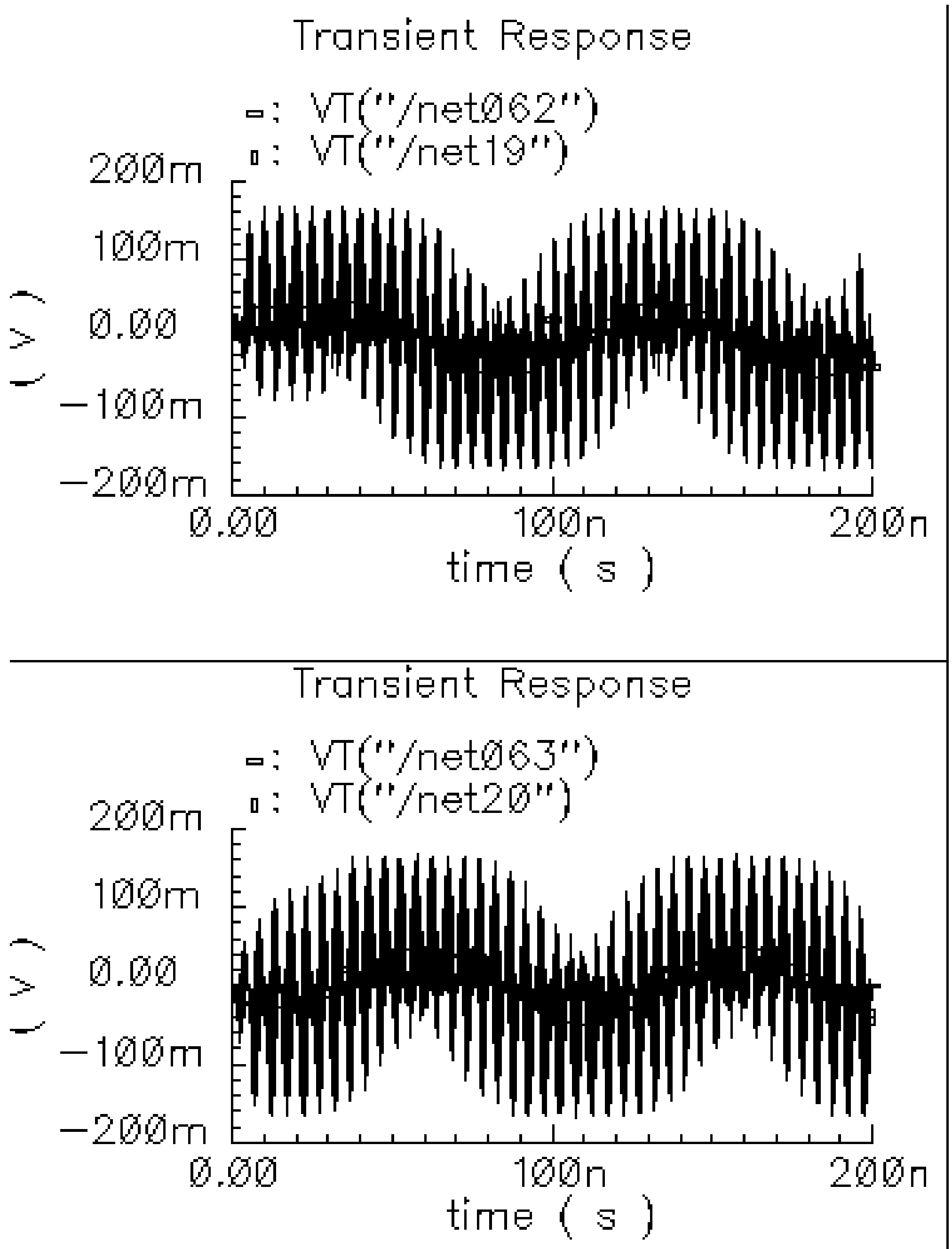


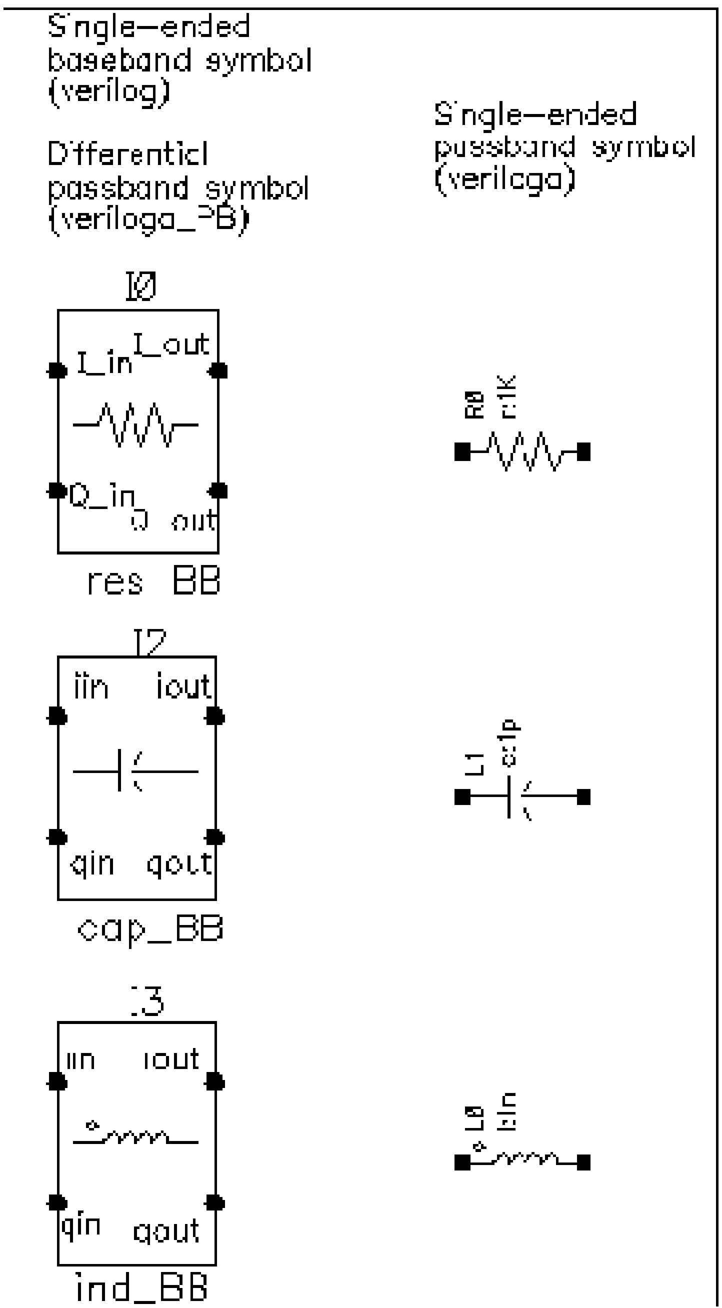
Figure D-65 Results from a Transient Analysis



Passive Devices

`(res_BB, cap_BB, ind_BB)`

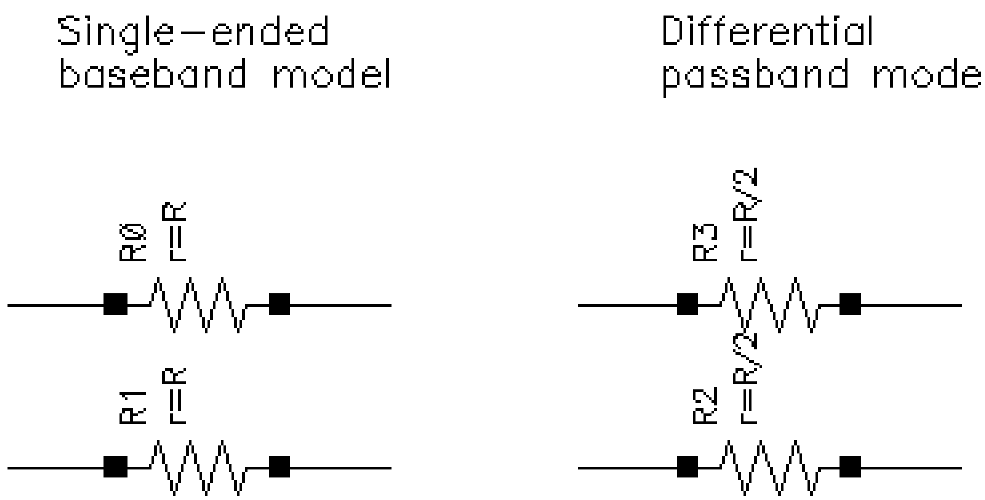
Figure D-66 Circuit



Resistors

Besides the resistance, the baseband resistor model has a parameter for turning its thermal noise on or off. The baseband resistor is intended for use at a passband node because it's noise is doubled. (This was discussed in the section entitled "Relationship between baseband and passband noise"). Figure D-67 on page 1169 shows the symbol, baseband, and passband models. The total noise in the differential passband resistor model equals the noise in one resistor of R Ohms.

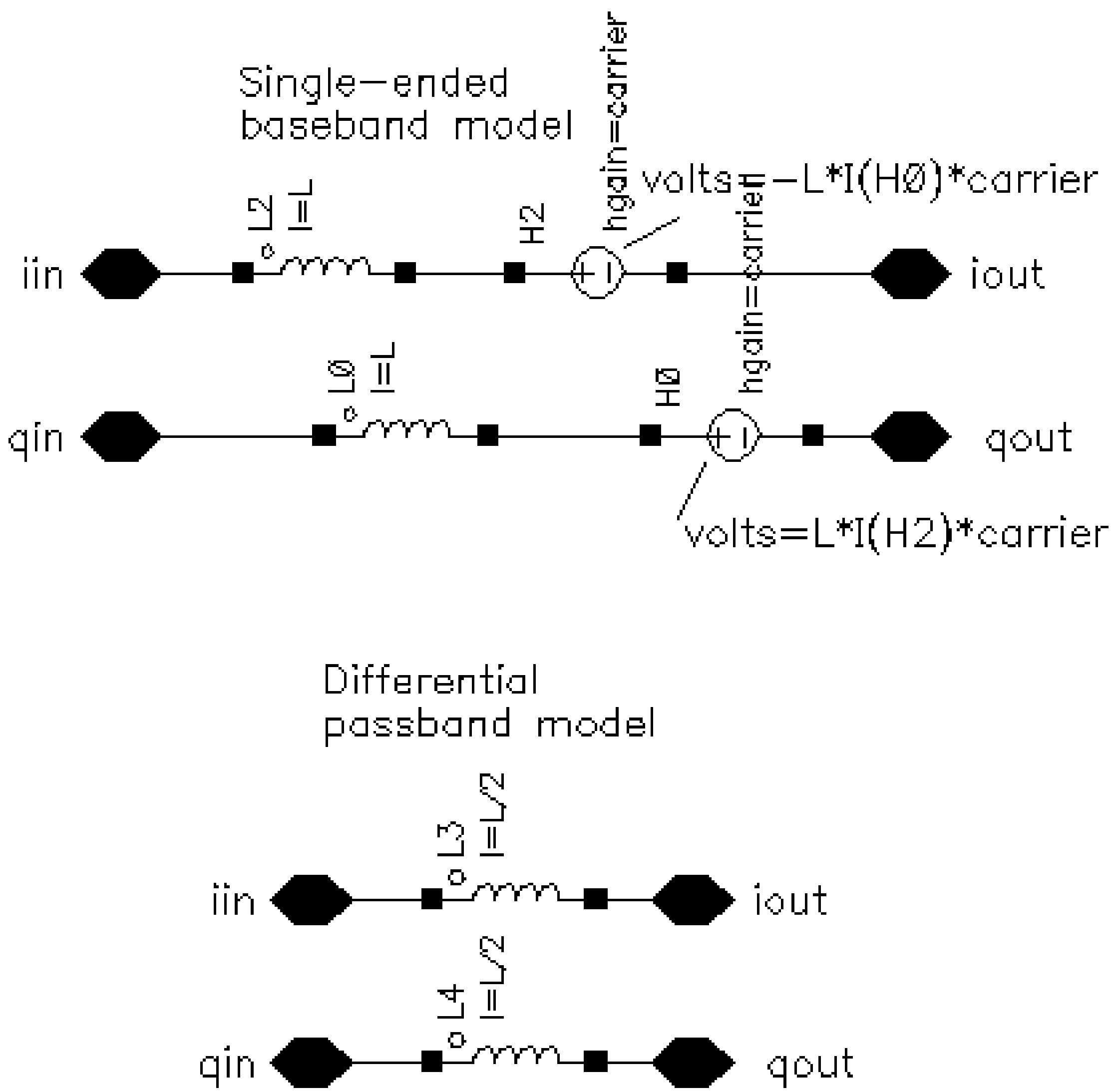
Figure D-67 Resistor model



Inductors

The library overview described the basic idea behind baseband modeling of inductors. The baseband inductor model requires one additional parameter besides the inductance, the carrier frequency. Figure D-68 on page 1170 shows equivalent schematics of the baseband and differential passband inductor models. The inductor models are noiseless.

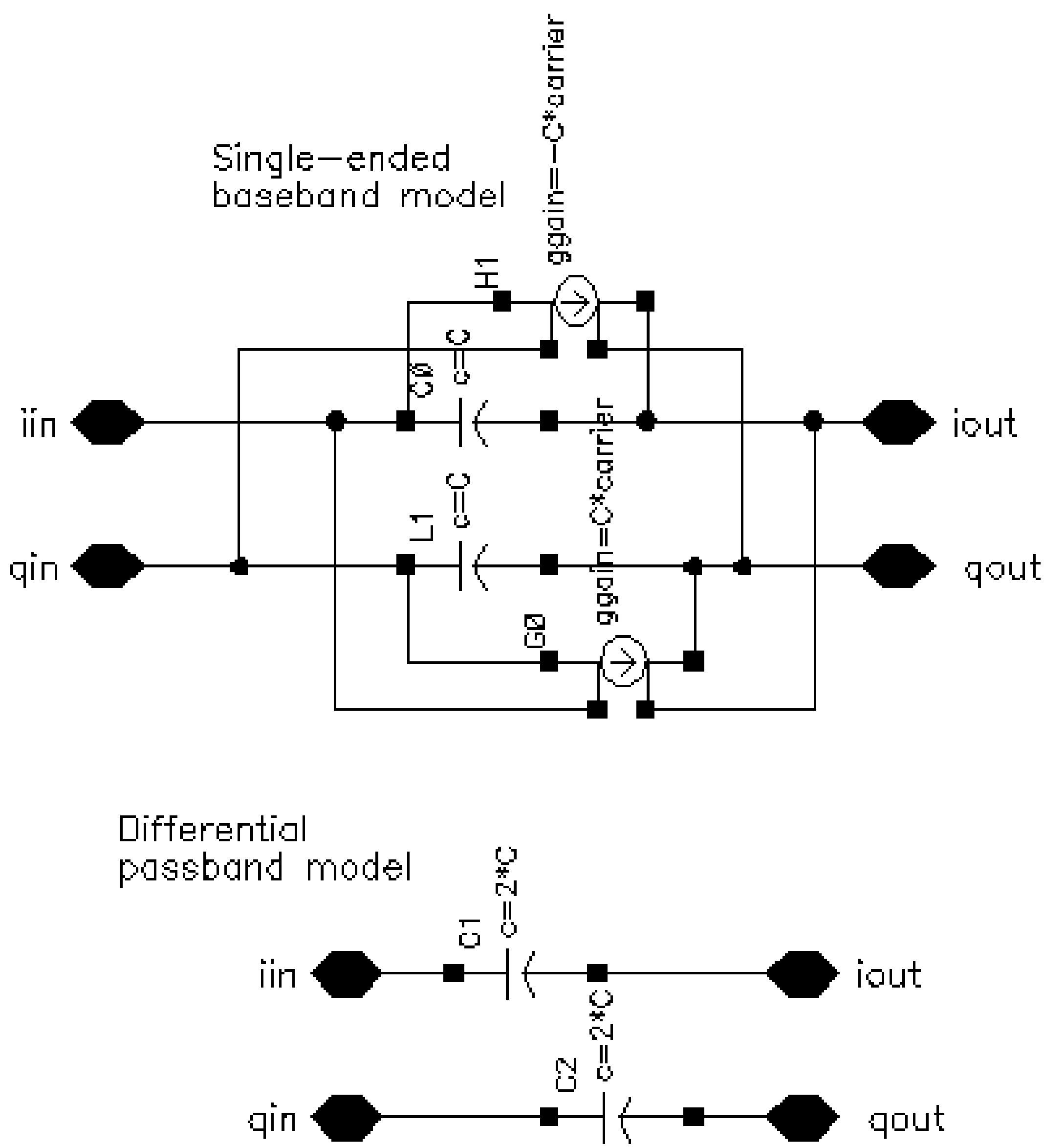
Figure D-68 Inductor model



Capacitors

The capacitor is the mathematical dual of the inductor. Figure D-69 on page 1171 shows the baseband and differential passband capacitor models.

Figure D-69 Capacitor model



RLC Test Circuits

The two circuits discussed below demonstrate how passband and baseband reactive elements are related. The circuit in Figure D-70 on page 1172 shows a simple passband RLC circuit driven by a modulated carrier. The circuit in Figure D-71 on page 1173 shows the associated baseband equivalent circuit model. The circuits are *PB_ind_cap_test* and *BB_ind_cap_test*. Both circuits reside in the rfLib under the *testbenches* category.

Figure D-70 Simple Passband RLC Circuit

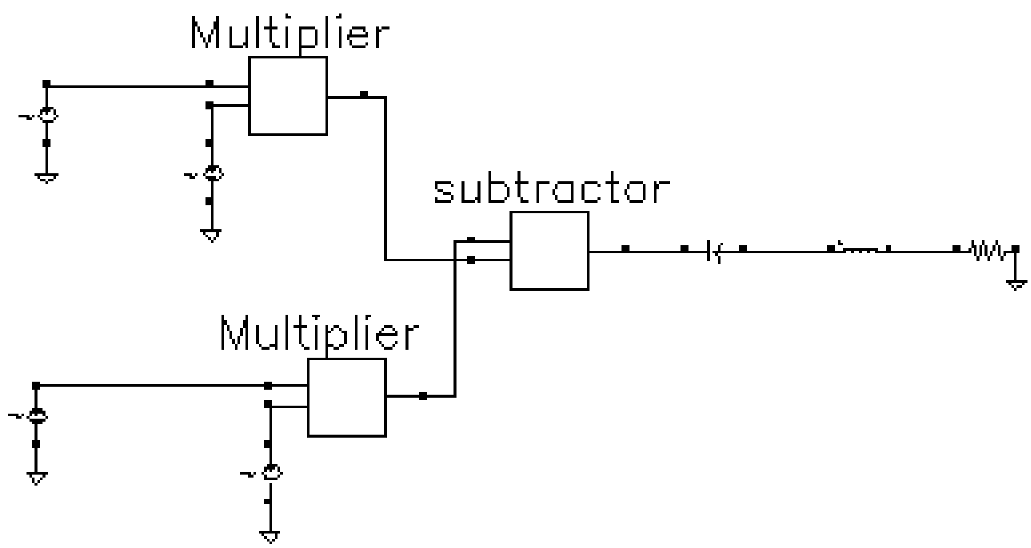
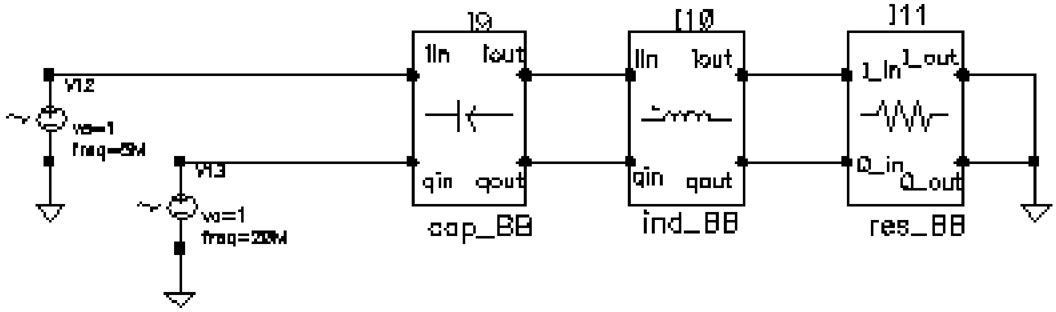


Figure D-71 Baseband Equivalent To Figure 1-63



The following steps explain how to simulate each circuit and overlay the results.

1. Recall the PB_ind_cap_test circuit and bring up an analog design environment window. Set up a 200ns Envelope analysis. Select `carrier` as the *Clock Name*. Set the Output Harmonics to 1.
2. Run the analysis and plot the real and imaginary parts of the harmonic-time voltage across the resistor. Use 1 for the harmonic number.
3. Recall the BB_ind_cap_test circuit and run a 200ns Transient analysis. Note the faster run time. That is the whole point to suppressing the carrier but it is only useful if the results match. Plot the *I_in* and *Q_in* voltages of the resistor model.
4. To overlay the results, bring up a waveform calculator.
5. Click the *wave* button on the calculator then click on one of the Envelope Following waveforms. If the waveform turns yellow you may have to hit the escape button a few times and click *clear* and *clst* a couple of times in the calculator then try again.
6. Make active the waveform display tool with the Transient results then click *Plot* in the calculator.
7. Repeat the last two steps for the other Envelope waveform. You should see the waveforms in Figure D-72 on page 1174. The two models agree very well. The resonant frequency of the series RLC branch is just over 500MHz. Only by riding on a carrier can the 5MHz and 20Mhz baseband signals propagate to the resistor at their original voltage levels. The baseband model accurately predicts the effects of the RLC circuit on the

baseband signal. There are two effects, one due to phase shift at the carrier frequency and one due to filtering of the baseband signal itself.

8. In the waveform display tool that overlays the results, change the x-axis to be one of the I-signals. You should get the picture shown in Figure D-73 on page 1175. The tilt in the resulting Lissajous plot indicates phase shift at the carrier frequency but not at the baseband frequencies. The aspect ratio of the Lissajous figure indicates the 20MHz component is attenuated more than the 5MHz component. The baseband model captures both effects well.

Figure D-72 Waveforms

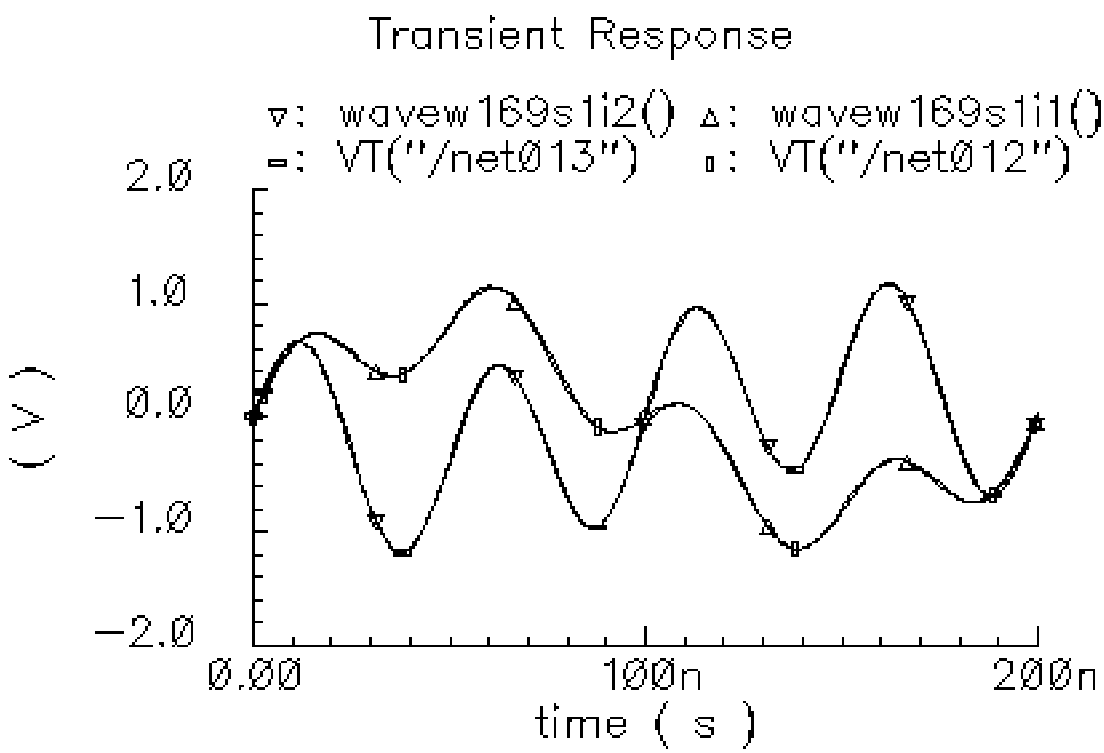
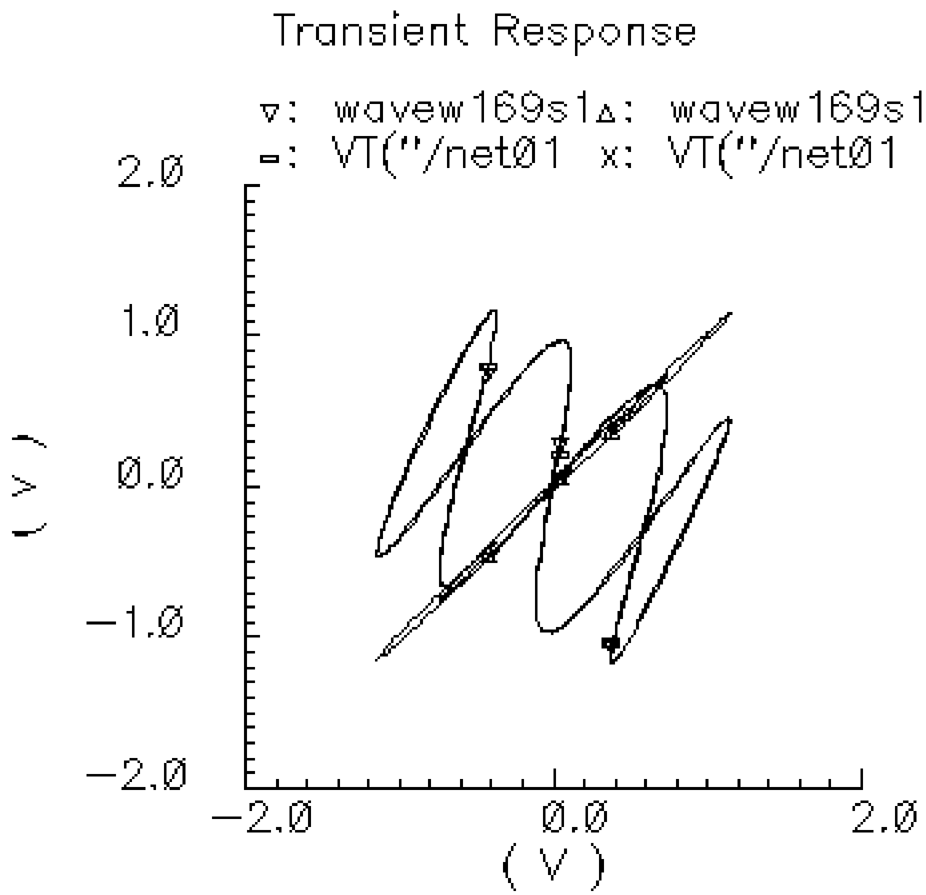


Figure D-73 Lissajous plot



Linear Time Invariant Filters

Figure D-74 Butterworth Filters

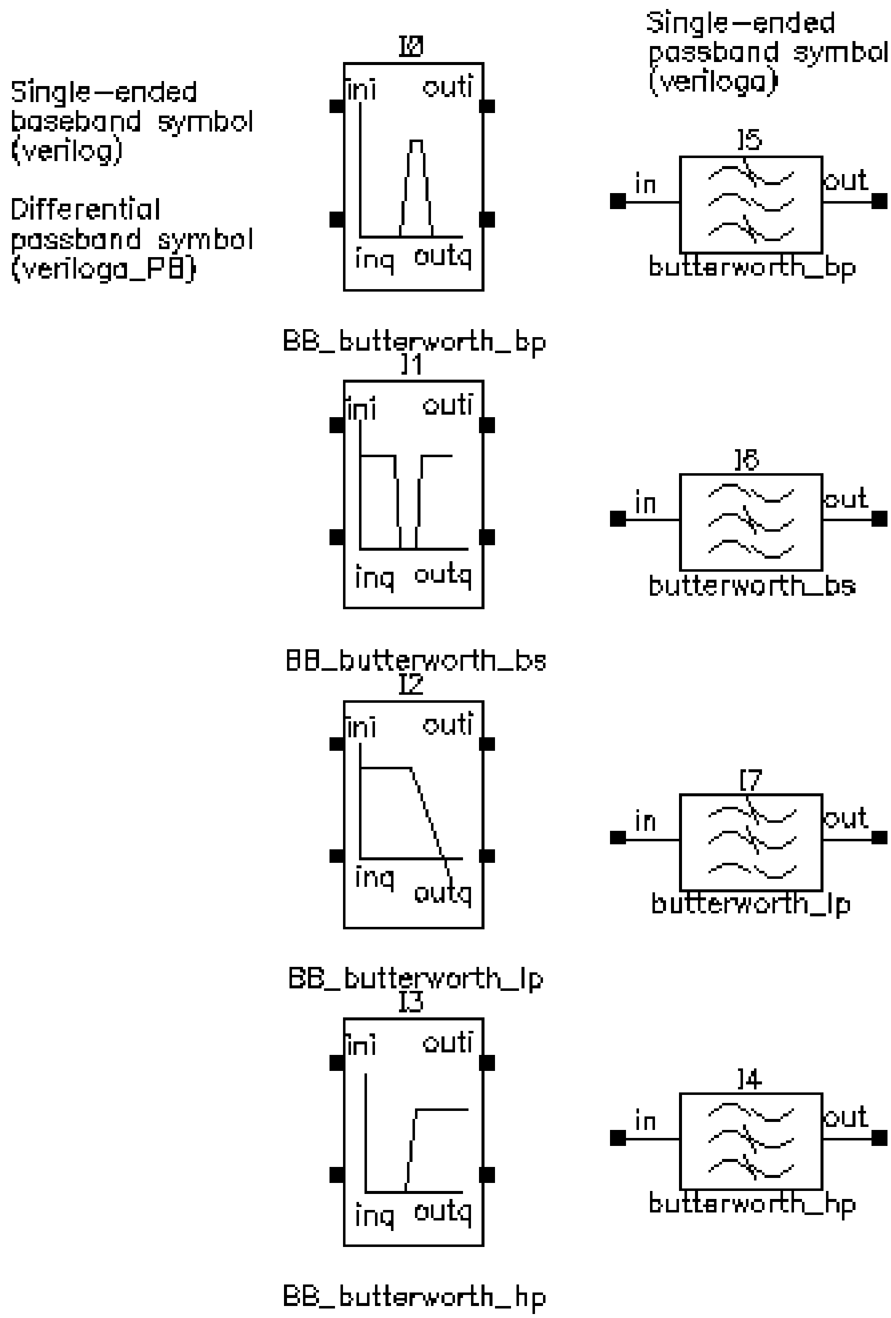
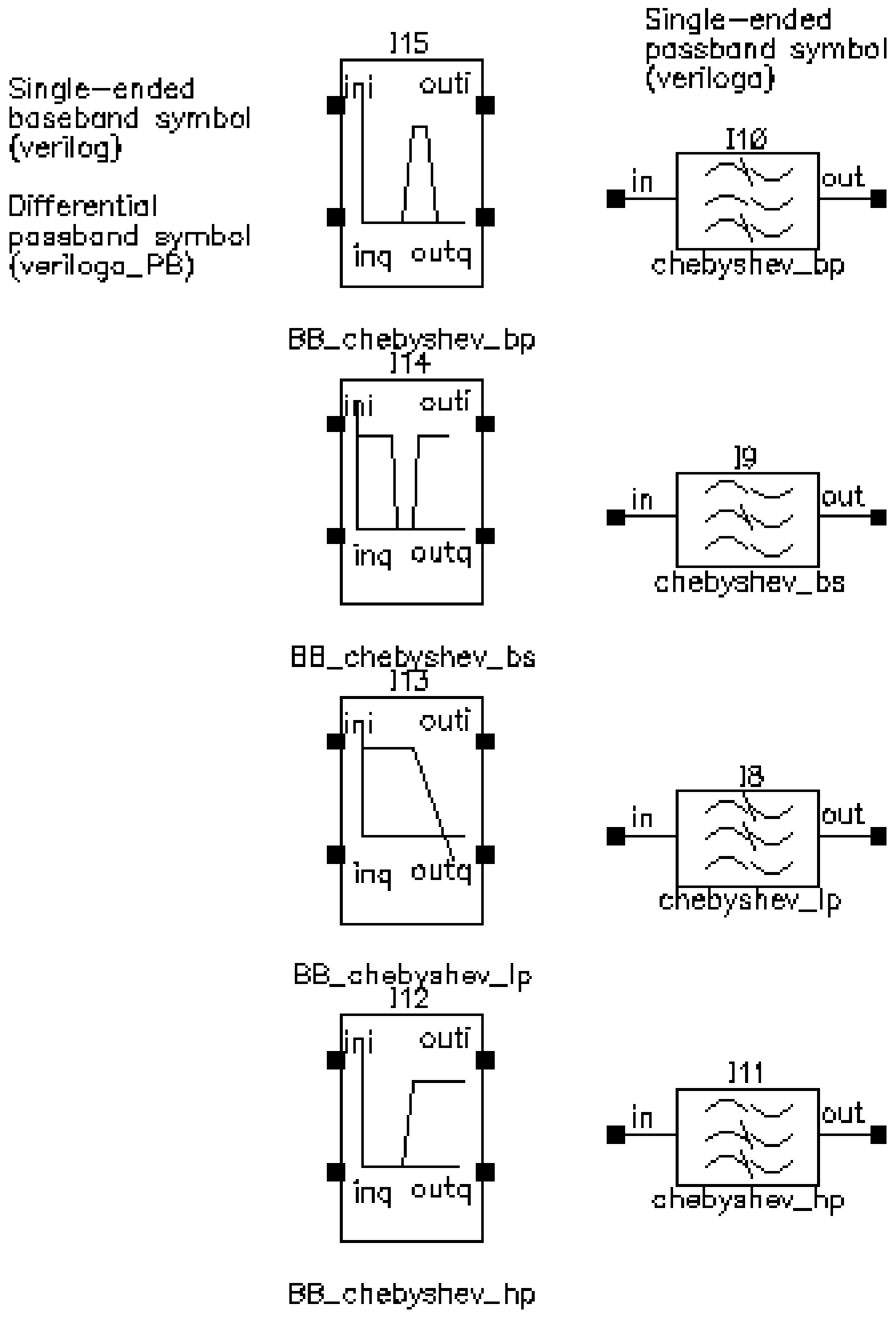


Figure D-75 Chebyshev Filters



The single-ended passband filters are the filters in the original RF AHDL library described in Appendix D. The single-ended baseband filters are baseband versions of the filters described in Appendix D. The passband views of the baseband equivalent filter models are differential versions of the filters described in Appendix D.

All filter models are LC ladder networks built up during the netlisting step. The only difference between the single-ended passband and baseband models is that the baseband models use baseband equivalent inductor and capacitor models, which means the baseband filter models have twice as many nodes as their passband counterparts. The only difference in the parameter list is that the baseband models require the carrier frequency. The carrier frequency is independent of all other parameters. It is the frequency at which the baseband signal is reference. If the input signal rides on a carrier that is offset from the reference carrier, the input baseband signal should be spun at the offset frequency before entering the filter model. This can be done at the preceding mixer stage by ramping the phase_err pin at a rate equal to the frequency offset.

Parameters (All parameters are defined in terms of passband models.)

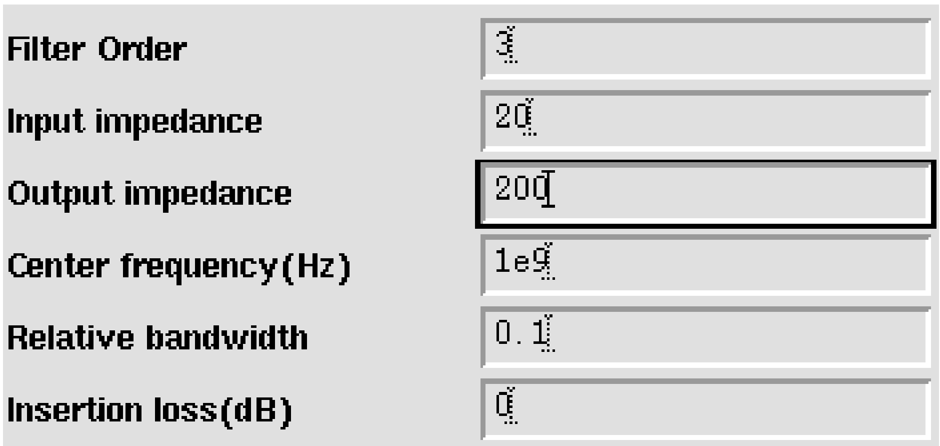
Note: The input impedance must be less than or equal to the output impedance to produce the desired filter response.

Input and Output impedance:

The input and output impedances are resistive reference impedances used to compute the inductors and capacitors that make up the filter. **The input and output impedances are not terminal impedances.** If you load one side of a filter from the library with a resistor of R Ohms and drive the other side with a frequency-swept current source of 1 Amp, you will see a voltage of R Volts across the current source at the passband frequency. (If the filter is a bandstop filter the passband is on either side of the stop band.) The following steps demonstrate this point with the four Butterworth filters.

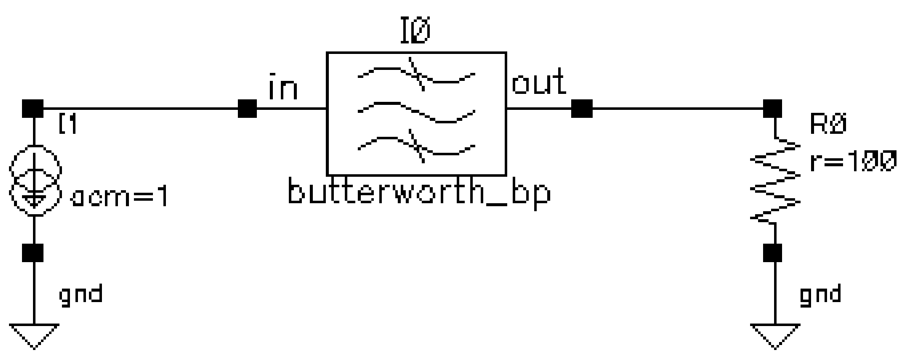
1. Bring up a new schematic in any library to which you can write.
2. Instantiate a butterworth_bp (bandpass) filter with the parameters as shown in

Figure D-76 Filter parameters



3. Drive the left terminal (the “input”) of the filter with a DC current source and set the source’s AC magnitude to 1 Amp.
4. Load the right terminal (the “output”) of the filter with a 100 Ohm resistor. The schematic should look like Figure D-77 on page 1179.

Figure D-77 Filter impedance test circuit



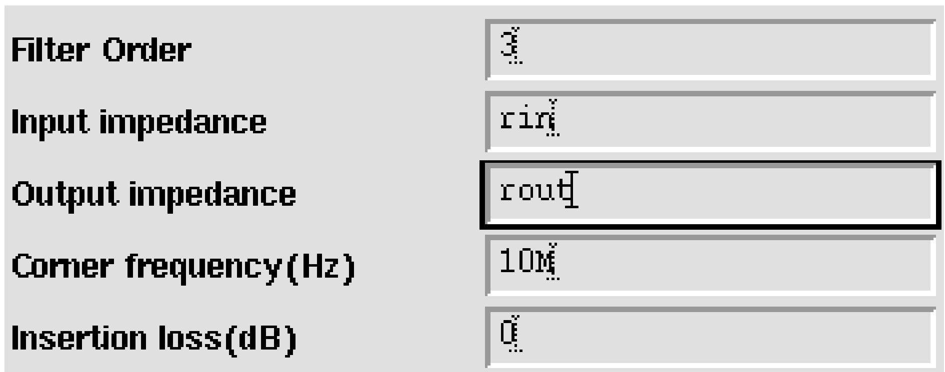
5. Set up an AC analysis. Sweep frequency from 950MHz to 1.05MHz in 100 linear steps.
6. Run the analysis.

7. Plot the voltage across the current source. You should see the voltage across the current source dip to 100 Volts right at the center frequency of 1GHz. At the center frequency, the impedance looking into a bandpass filter equals the impedance loading the other side, regardless of the filter's input and output impedance parameters.
8. Change the filter to butterworth_bs (bandstop filter).
9. Check and save the schematic.
10. In the AC analysis setup, change the AC analysis to sweep frequency from 1Hz to 100GHz in 1 linear step. Do not add any specific points.
11. Run the analysis.
12. Plot the voltage across the current source. You should see 100 Volts at either end point of the sweep. Far away from the center frequency of a bandstop filter, the impedance looking into the filter equals the impedance loading the other side.
13. Change the filter to butterworth_lp (low pass filter). Set the corner frequency to 10MHz.
14. Check and save the schematic.
15. Change the AC analysis to sweep from 1Hz to 1GHz logarithmically with 10 points per decade.
16. Run the analysis.
17. Plot the voltage across the current source. Near DC, the impedance looking into a low pass filter equals the impedance loading the other side.
18. Change the filter to butterworth_hp (highpass filter).
19. Change the AC analysis to sweep from 1GHz to 100GHz.
20. Run the Analysis.
21. Plot the voltage across the current source. As frequency goes to infinity, the impedance looking into a highpass filter equals the impedance loading the other side.

How are the input and output impedances used if they don't determine terminal impedances? Let the filter be specified with input and output resistances of rin and rout respectively. If you drive the filter with a Port of rin Ohms and load it with a Port of rout Ohms, an S-parameter analysis will produce an S21 with the shape associated with the filter type and associated specifications. The following steps demonstrate this point.

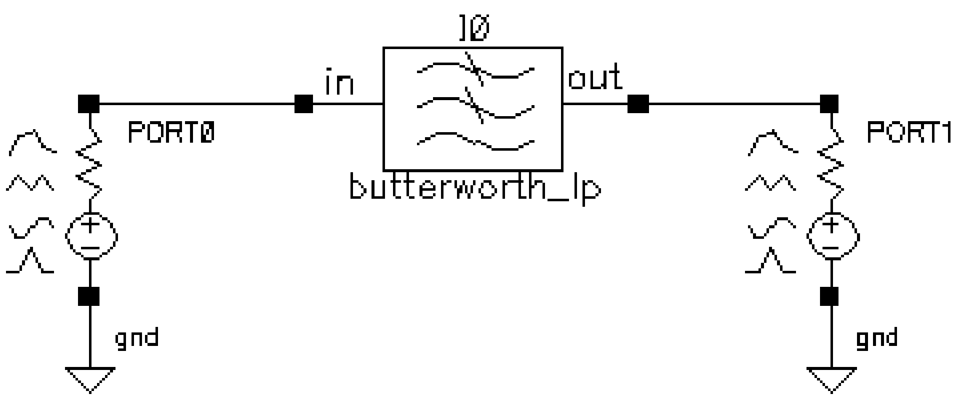
1. Bring up a new schematic window in a library you can write to.
2. Instantiate a butterworth_lp (low pass filter) and set the parameters as shown in Figure D-78 on page 1181.

Figure D-78 Filter parameters



3. Drive the filter a Port. Set the Port Resistance to rin Ohms and the Port number to 1.
4. Load the filter with another Port. Set the Port Resistance to rout Ohms and the Port number to 2. The schematic should look like Figure D-79 on page 1181.

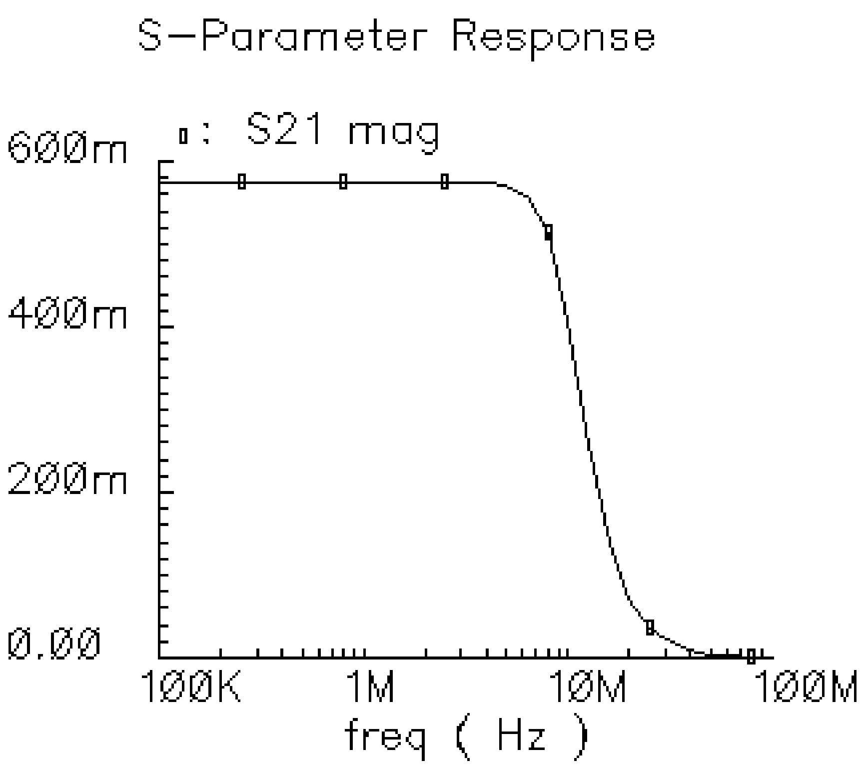
Figure D-79 S21 test circuit



5. Bring up an Analog Environment window. Copy variables from the cell view. Set rout to 200 Ohms and rin to 20 Ohms.
6. Set up an sp (s-parameter) analysis. Sweep frequency from 100KHz to 100MHz logarithmically with 10 points per decade.
7. Run the analysis.

8. Plot S21 and you should see the plot in Figure D-80 on page 1182. The corner frequency is where S21 drops to 70.71% of its maximum value. If you check, that point is right where it was specified, at 10MHz. Note that the maximum value of S21 is not 1. That is because the input and output reference resistances are not equal. In the passband, the test circuit becomes a voltage divider. Using the basic definition of S21, for this voltage divider one can show that S21 equals the geometric mean of the input and output reference resistances divided by their arithmetic mean; the maximum value of S21 should equal 2*Sqrt(200+20)/(200+20) = 0.575, just as the plot shows.

Figure D-80 S21 plot



Insertion Loss:

This is the voltage drop in dB, at the passband or center frequency, the filter would suffer if it were driven and loaded with the input and output resistances respectively.

Filter noise is computed from the insertion loss according to:

noise_current = 2*sqrt((‘db10_real(loss)-1)*1.380620e-23*$temperature/r1);

“r1” is the input impedance and “db10_real()” is a function that maps loss in dB into linear attenuation. The definition is as follows:

‘define db10_real(x) (pow(10, (x)/10))

Filter Order:

Low pass-This is total number of reactive elements in the filter. If the order is odd, there is one more capacitor than inductors. If the order is even, there are equal numbers of capacitors and inductors.

High pass-This is the total number of reactive elements in the filter. If the order is odd, there is one more inductor than capacitors. If the order if even there are equal numbers of capacitors and inductors.

Band pass and Band stop- The filter order is the number of inductor/capacitor pairs.

Corner frequency:

This is the frequency where S21 is 3dB below the peak response when driven and loaded with the input and output resistances respectively.

Center frequency:

For bandpass filters, the center frequency is where S21 is maximum. For bandstop filters, the center frequency is where S21 is a minimum.

Relative bandwidth:

This applies to band pass and band stop filters. It is the normalized frequency difference between points straddling the center frequency that correspond a 3dB drop (or rise) from the peak (dip) of S21. The normalization factor is the center frequency. In other words, the relative bandwidth is specified in terms of a fraction of the center frequency. A relative bandwidth of 0.1 and a carrier frequency of 1GHz means that in the case of a bandpass filter, the pass band goes from 950MHz to 1.05GHz with the band edges defined by those points lying 3dB below the peak response.

Passband ripple:

This applies only to Chebyshev filters. It is the peak-to-peak variation of the transfer function in the passband when the filter is driven and loaded with the input and output resistances.

BB_Loss

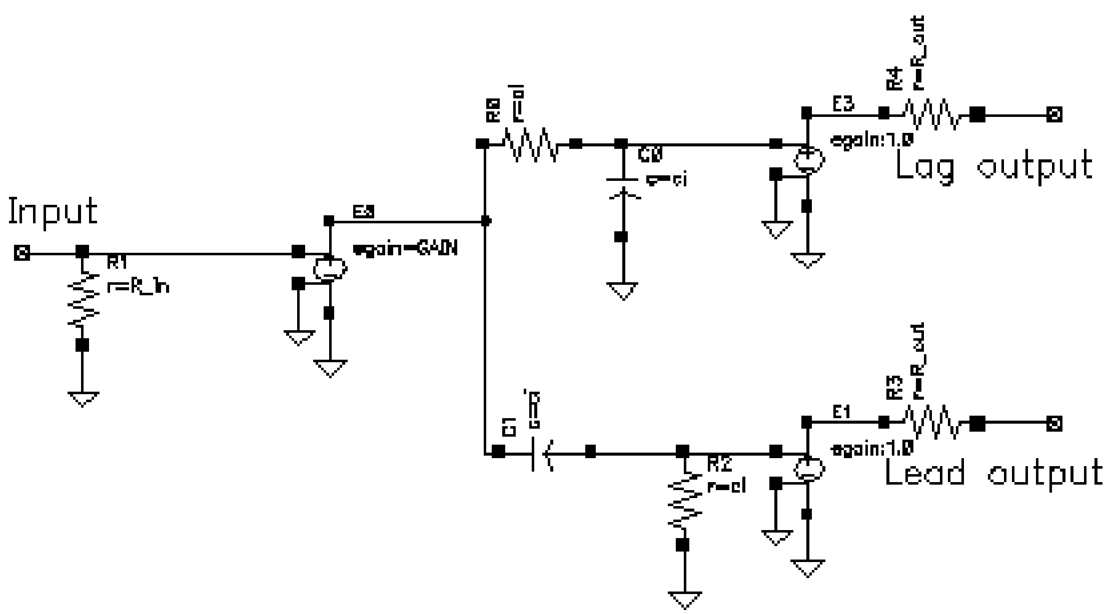
The BB_loss element is designed to be used with error vector magnitude (EVM) calculations. EVM is defined in terms of an ideal receiver or transmitter. If you want to remove a filter's response from the ideal receiver model while leaving only the passband attenuation, replace the filter with a BB_loss element and give it the same insertion loss as the filter. There is no passband view or counterpart for this model because a passband EVM analysis is not practical in this release.

Phase Shifter Splitter

`(shifter-splitter)`

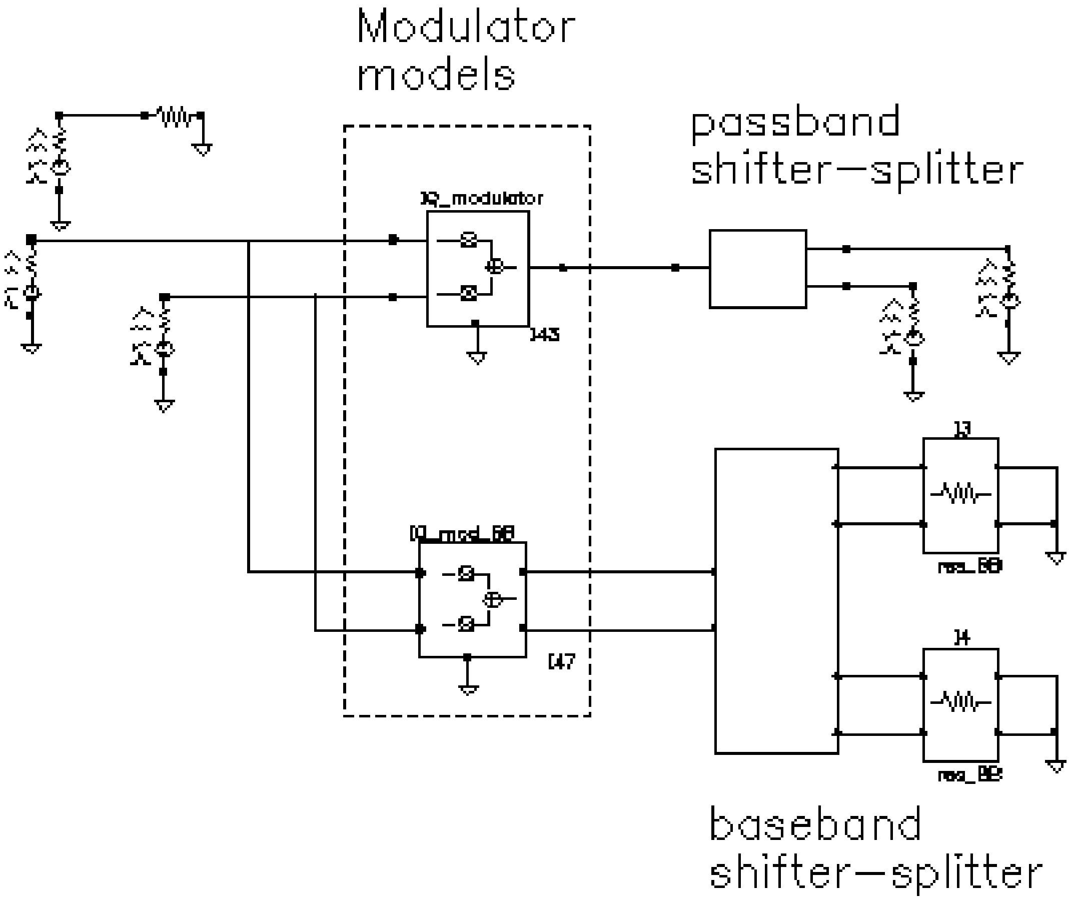
The phase shifter-splitter has one input and two outputs. The outputs are phase shifted versions of the input at the specified frequency. The phase difference between the two outputs is 90 degrees. The phase shifts are accomplished with VerilogA code that does the same thing as the circuit shown in Figure D-81 on page 1185. The right-most voltage-controlled-voltage-sources (vcvs) are unity gain buffers. The left-most vcvs is also a buffer but the gain is a user-defined parameter. The input resistance, output resistance, intended operating frequency, and internal resistance are also user-defined. The internal resistance and operating frequency are used to calculate the capacitance necessary to provide +-45 degrees of phase shifts at the operating frequency. The baseband view requires the carrier frequency.

Figure D-81 Phase Shift Circuit



The test circuit in Figure D-82 on page 1186 is for comparing the baseband responses of the passband and baseband equivalent models of the shifter-splitter. The circuit can be found in rfLib under the *testbenches* category. It is listed as *shifter_splitter_test*.

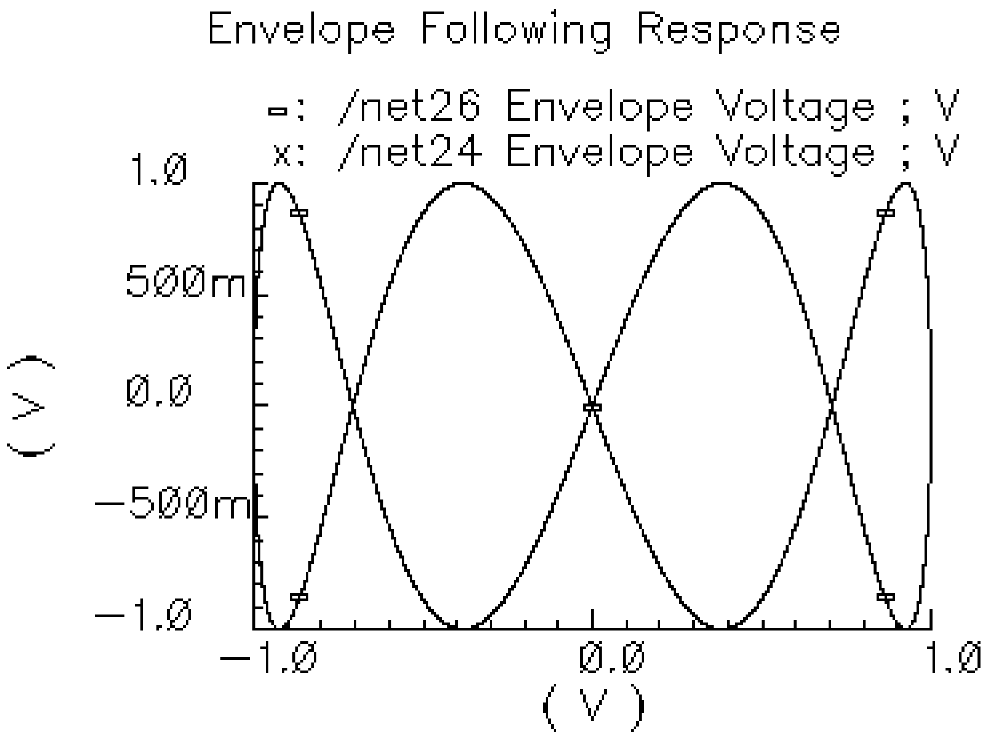
Figure D-82 shifter_splitter_test Circuit



The following steps produce a set of Lissajous plots that show what the shifter-splitter does. We will observe phase shift in the carrier by observing the tilt of the output Lissajous figures generated by the equivalent baseband signals.

1. Bring up the test circuit in Figure D-82 on page 1186 and an Analog Environment window.
2. Set up a 200ns Envelope Following analysis with “carrier” as the Clock Name. Set the Number of harmonics to 1. Set the Envelope Following analysis option called *modulationBW* equal to 100MHz.
3. Run the analysis.
4. Plot the “time” waveforms of the two input baseband signals. Change the x-axis to be the I-signal. You should see the Lissajous plot in Figure D-83 on page 1187.

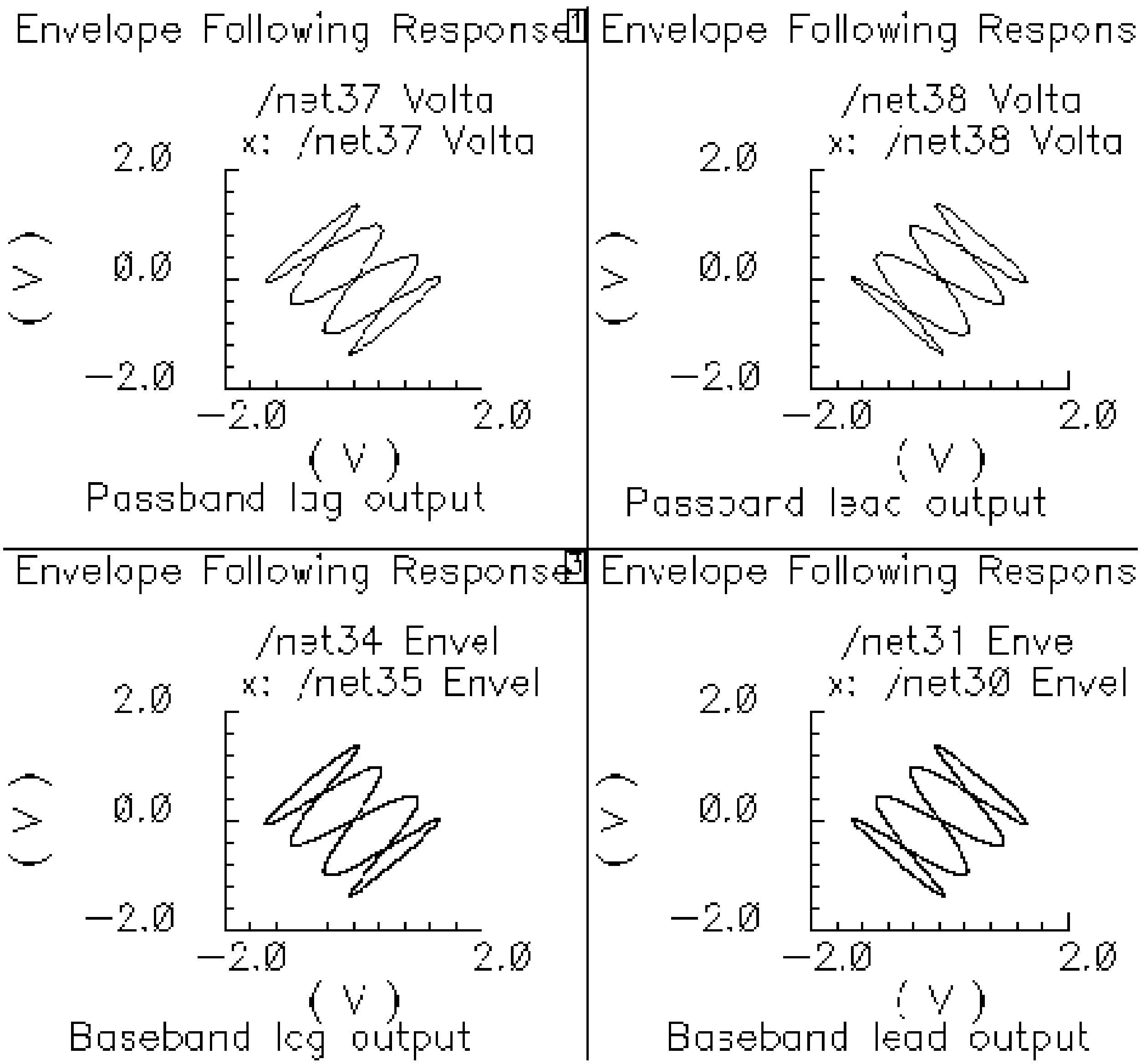
Figure D-83 Lissajous Plot for Baseband Input Signals



5. Reset the Waveform Display window and plot the harmonic time, 1 harmonic, real and imaginary parts of the voltage across Port2. Set the x-axis to be the real part. Note that the Lissajous plot is tilted -45 degrees from the one in Figure D-83 on page 1187.
6. Add a subwindow.
7. Repeat step 5 for the voltage across Port1. Notice that the Lissajous plot is tilted +45 degrees with respect to the Lissajous plot in Figure D-83 on page 1187.
8. Add another subwindow.
9. Plot the “time” waveforms at the lag_I and lag_Q outputs of the BB_shifter_splitter model. Set the x-axis to be the lag_I waveform. The Lissajous plot should match the one produced in step 5.
10. Add another subwindow.
11. Repeat step 9 for the “lead” outputs of the BB_shifter_splitter model. The Lissajous plot should match the one produced in step 7. Aside from the labels, your Waveform Display tool should look like Figure D-84 on page 1188. The time-results of the baseband model

faithfully duplicates the passband results but without simulating the carrier. The baseband model can be run with Spectre Transient analysis.

Figure D-84 Comparison of Lag and Lead times for Passband and Baseband Models

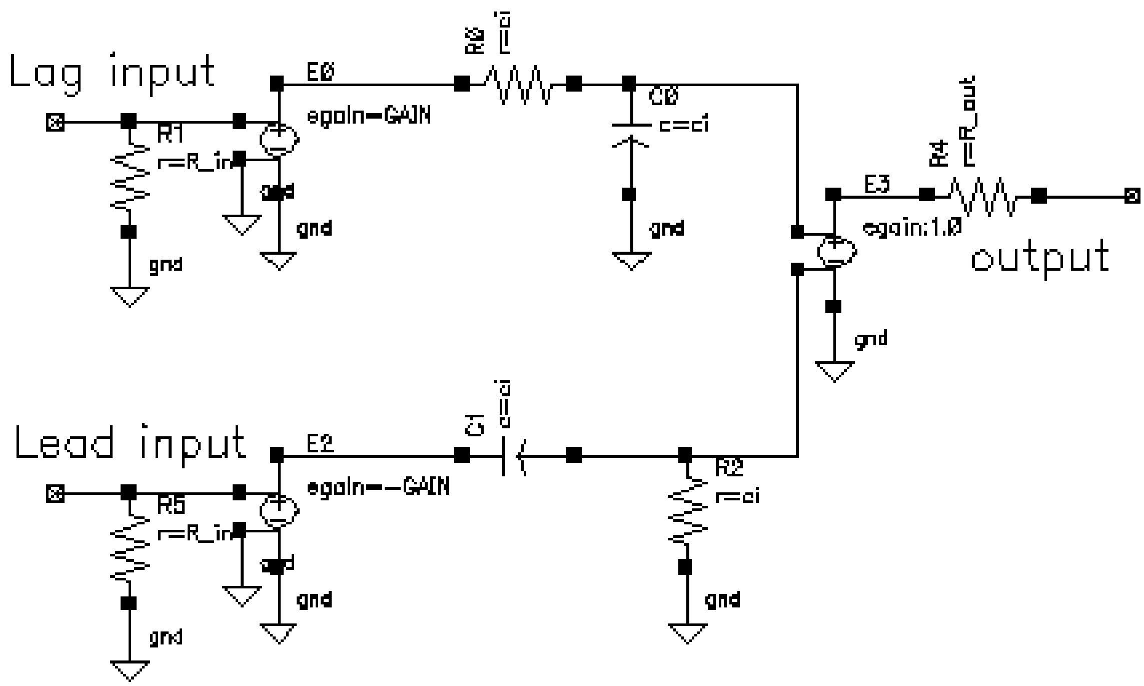


Phase Shifter Combiner

`(shifter-combiner)`

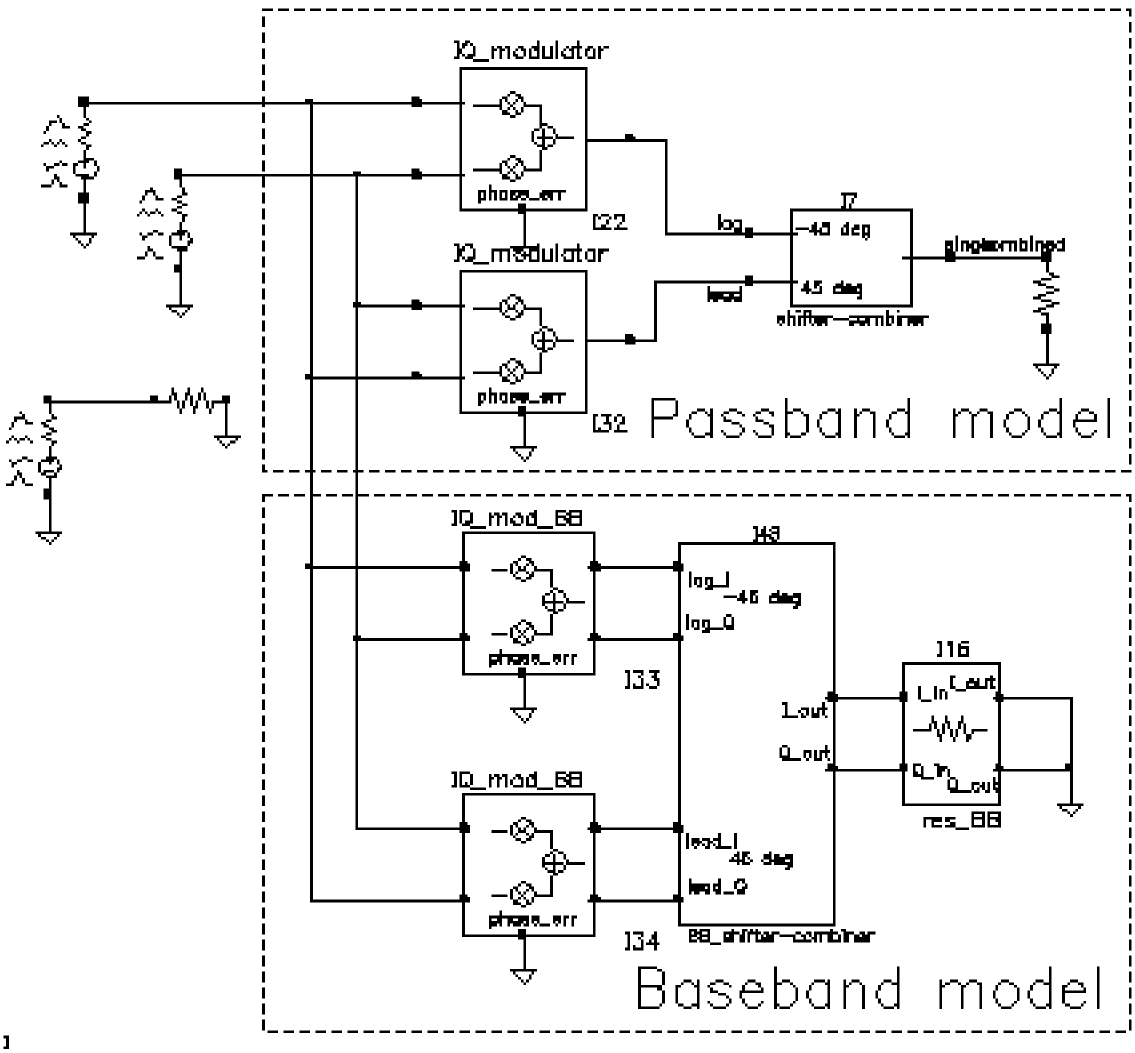
The phase shifter-combiner has two inputs and one output. The inputs are phase shifted by +/- 45 degrees then added together to form the output. All terminals are buffered and have the specified terminal resistances. The phase shifts are accomplished with VerilogA code that does the same thing as the circuit shown in Figure D-85 on page 1189. The gains of the left-most voltage-controlled-voltage sources are user-defined. The input resistance, output resistance, intended operating frequency, and internal resistance are also user-defined. The internal resistance and operating frequency are used to calculate the capacitance necessary to provide 45 degrees of phase shifts at the operating frequency. The baseband view requires that the carrier frequency be specified.

Figure D-85 Phase Shift Circuit



The shifter-combiner can be used to eliminate one phase of the carrier. The test circuit in Figure D-86 on page 1190 shows a simple test to demonstrate the idea. The circuit is in the rfLib under the *testbenches* category and listed as *shifter_combiner_test*. The top circuit is a passband model and the bottom circuit is the baseband equivalent. Baseband input signals are mixed up tot 1GHz then passed into the shifter-combiner. The baseband signal contains 10MHz and 20Mhz components. The modulators and shifter-combiner are arranged to produce only a 20MHz signal riding on the carrier.

Figure D-86 shifter_combiner_test Circuit

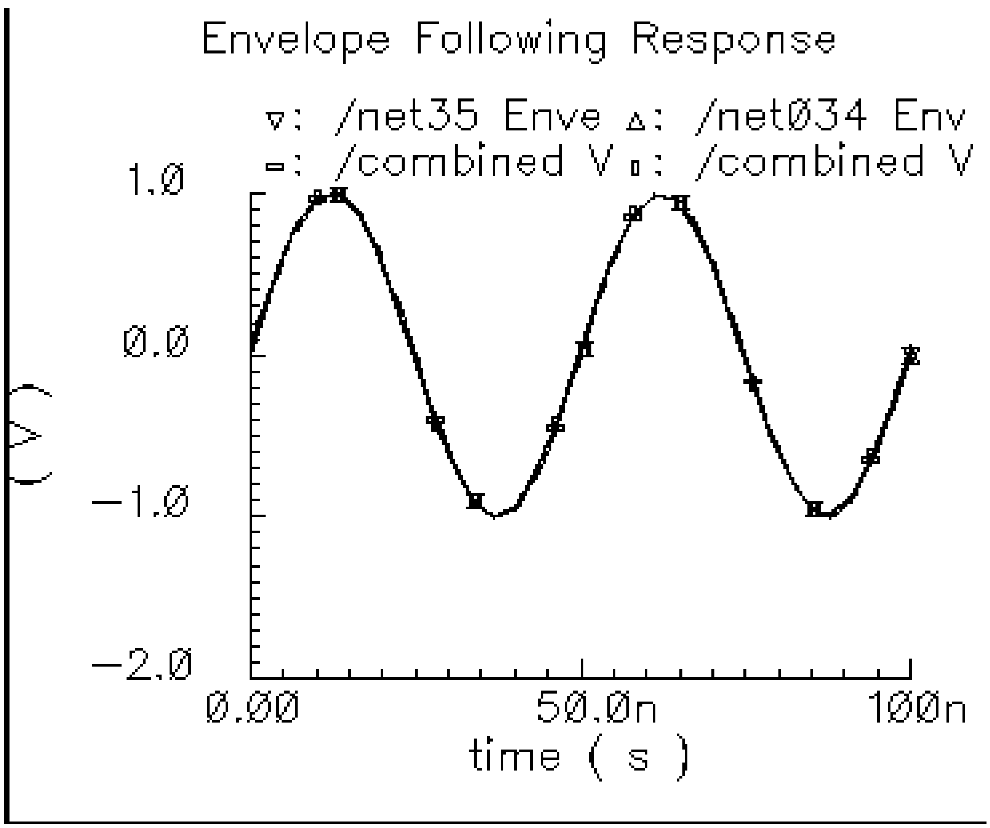


To check this assertion:

1. Bring up the test circuit and an Analog Environment window.
2. Set up a 100ns Envelope Following analysis on the circuit with the Clock Name set to “carrier” and the modulationbw option set to 40MHz. Set the Harmonic number to 1.
3. Run the analysis.

4. Plot the harmonic time, 1 Harmonic, real and imaginary parts of the passband shifter combiner output.
5. Append to the plot, the “time” waveforms at I_out and Q_out pins of the BB_shifter _combiner model. Figure D-87 on page 1191 shows what you should now see in the Waveform Display window. All waveforms are the same and they contain only the 20Mhz baseband signal. The 10Mhz baseband input signal does not propagate to the output.

Figure D-87 shifter_combiner_test results



One application of the shifter-combiner is an image rejection receiver. Figure D-88 on page 1192 shows a very simple example of an image rejection receiver. Figure D-89 on page 1192 shows the baseband equivalent model of the receiver. Both examples are in the *rfExamples* directory and are listed as *image_reject_rcvr_PB* and *image_reject_rcvr_BB*. The local oscillator runs at 1GHz and the RF carrier is 1.1GHz, which places the image at 900Mhz. This example shows one of the limitations of the baseband equivalent models.

Figure D-88 A Simple Image Rejection Receiver

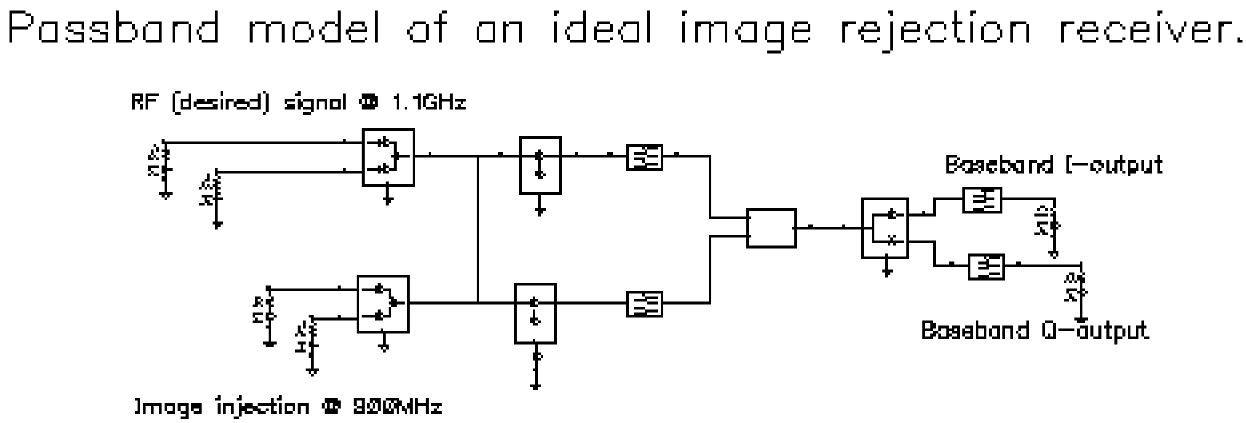
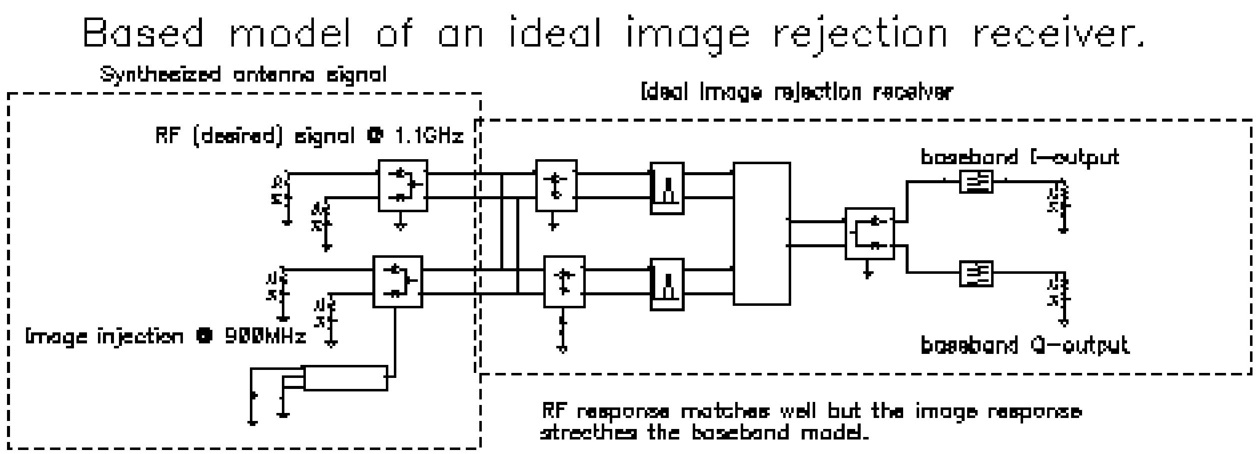


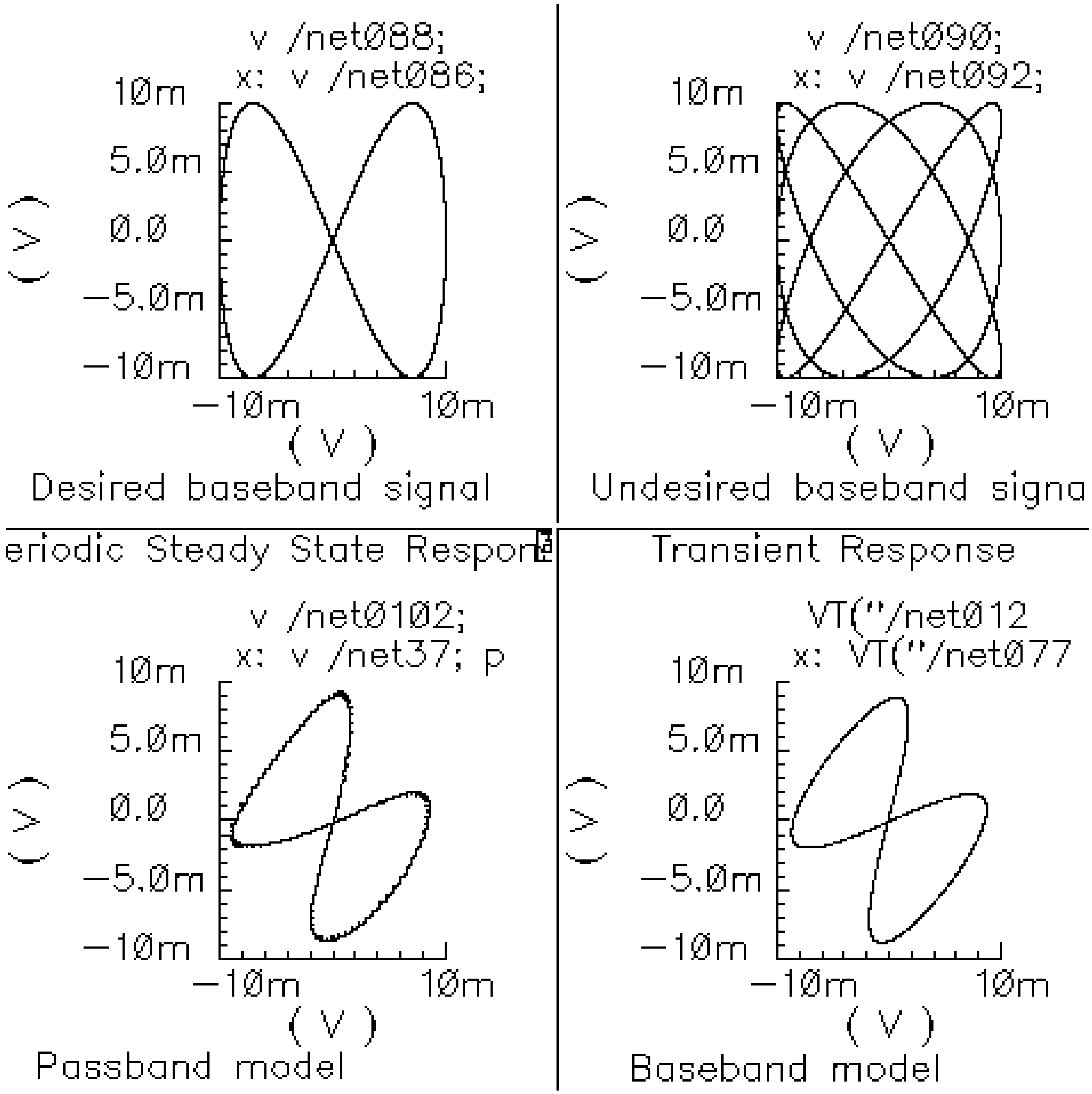
Figure D-89 Baseband Equivalent Model of the Image Rejection Receiver



1. Bring up the passband test circuit and an Analog Environment tool.
2. Set up a PSS analysis. You will need to add the 1.1GHz, 1GHz, and 900Mhz fundamental tones. Give them arbitrary but distinct names. AutoCalculate the Beat Frequency, which should be 2MHz. You need not save more than the 1st harmonic. Set the PSS maxstep option to 20ps so that it accurately simulates the oscillators hidden inside the Verilog-A modules. (In a future release, the modules will have Boundstep lines to take care of this problem.)
3. Run the analysis.

4. Plot the voltages across Ports 5 and 6. Set the x-axis to be the voltage across Port 6. This is a Lissajous plot of the desired baseband signal, the one riding on the 1.1GHz carrier.
5. Add a subwindow.
6. Plot the voltages across Ports 8 and 7. Set the x-axis to be the voltage across Port 8. This is a Lissajous plot of the undesired baseband signal, the signal riding on the image of the carrier at 900MHz.
7. Add another subwindow.
8. Plot the I and Q- baseband outputs. Set the x-axis to be the I-output. The Lissajous plot is a tilted version of the desired baseband signal, indicating that most of the image was successfully rejected.
9. Bring up the baseband equivalent receiver model and another Analog Environment tool.
10. Run a 10us Transient analysis with 9.5us as the output start in the analysis options and maxstep set to 250ps. The phase_err pin on the image signal generator is being driven to spin the output at 200MHz, the frequency difference between the desired frequency and image frequency.
11. Add another subwindow to the Waveform Display tool showing the passband results and make sure it is active.
12. Plot the I and Q baseband outputs from the baseband equivalent receiver model. Set the x-axis to be the I-output. You may need to make the scales on the last two plots are the same. Aside from the labels, the Waveform Display tool should look like Figure D-90 on page 1194.

Figure D-90 Lissajous Plots for Baseband Signals



The baseband equivalent receiver model indeed rejects the image but the rejection is over-estimated. If you look closely, the baseband output of the passband model contains more ripple from the image. The over-attenuated ripple in the baseband model is explained as follows.

Recall the rotating reference frame analogy for baseband modeling. With respect to the rotating 1.1GHz reference frame, the image signals rotate counter-clockwise at twice the IF, 200MHz in this case. The lower left block in the baseband receiver model spins the modulator output at -200Mhz by ramping the phase error pin. The -200MHz signal propagates through the IF bandpass filters, as it should, because the response of the baseband model of the filter peaks at DC and at minus 200MHz. The trouble occurs in the final downconversion to baseband. In the baseband model, the final low pass filters severely attenuate the -200Mhz image signal. However, in the passband model, image power at minus 100MHz contributes to the baseband signal through the low pass filters with less attenuation.

This example highlights one of the limitations of baseband equivalent models: at any point in the system, the signal should not have a bandwidth larger than any carrier (RF or IF) of the system. For this example, the baseband model is only valid for input RF signals between 1GHz and 1.2GHz.

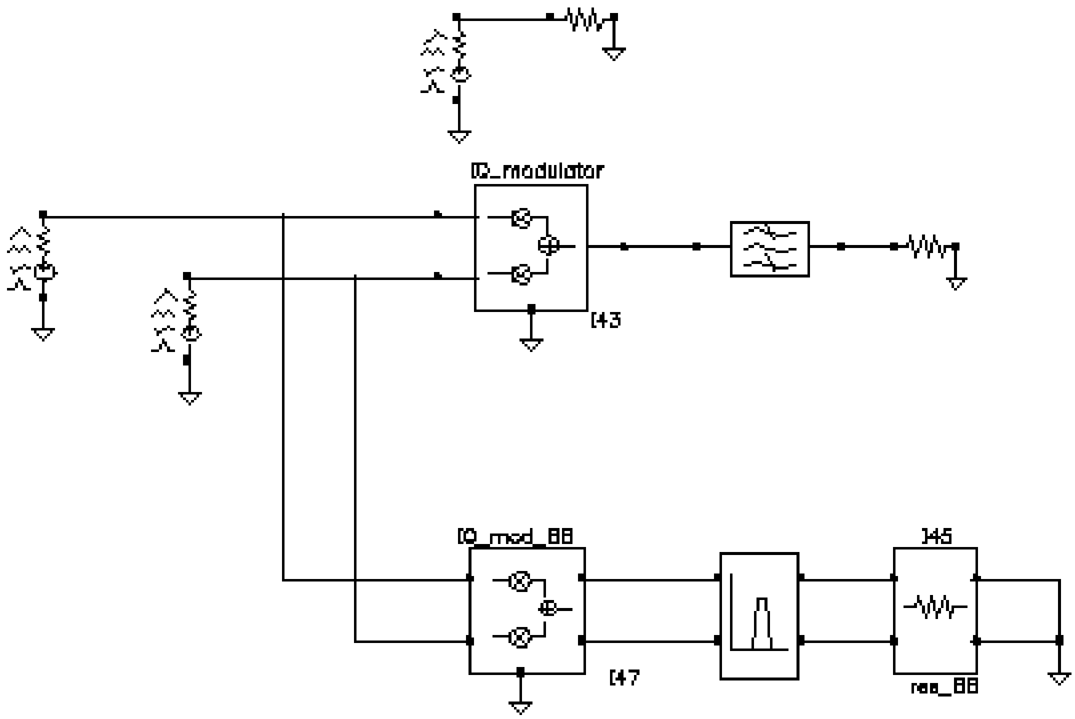
The limitation is somewhat moot because the idea behind a baseband equivalent model is to suppress all carriers. To simulate the image response with the baseband model we had to include a 200MHz source! We would have been better off simply not suppressing the 100MHz IF carrier, i.e. using baseband models for the RF stages but passband models for the IF stages.

In summary, an all-baseband equivalent model of an image rejection receiver is only good for simulating the response to the desired RF signal, not the image response.

Comparison of Baseband and Passband Models

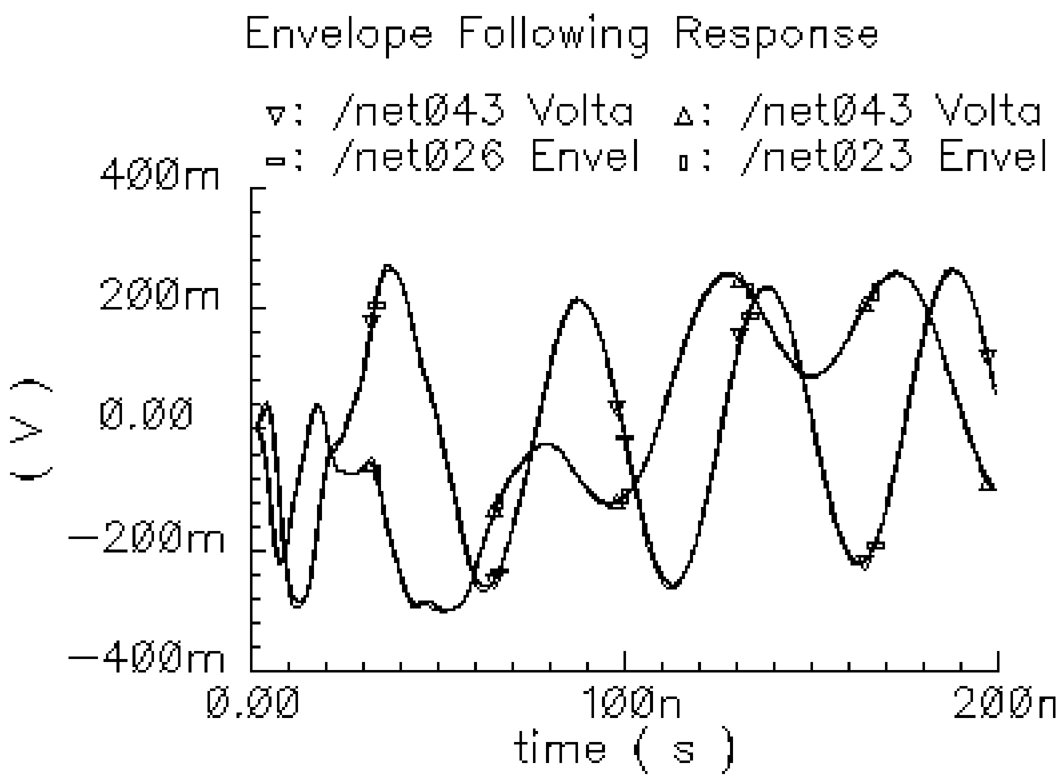
The circuit in Figure D-91 on page 1196 shows how well the baseband and passband filters agree. The I-input is a 5MHz 1 volt peak sinusoid and the Q-input signal is a 20MHz 1 volt peak signal. The filter has a center frequency of 1.1GHz and a relative bandwidth of 0.1. The modulator LO is 1GHz. To make the analysis more interesting the carrier is not exactly aligned with the filter center frequency and the terminals are not matched. The circuit is listed as *PB_BB_filter_comparison* in the testbenches category of the *rfLib*.

Figure D-91 PB_BB_filter_comparison Circuit



1. Bring up the test circuit and an Analog Environment window.
2. Set up an Envelope Following analysis with “carrier” as the Clock Name. Set reltol in the analog options to 1e-5. You can use the default reltol of 1e-3 but you will not get the waveforms close to the baseband results.
3. Plot the “time” waveforms of the BB_butterworth_bp outputs. These waveforms are the response of the baseband equivalent model.
4. Plot the “harmonic time”, 1 harmonic, real and imaginary waveforms at the butterworth_lp output. These waveforms are the baseband waveforms extracted from a passband model. Figure D-92 on page 1197 overlays the baseband and passband results. The baseband and passband filter models produce identical equivalent baseband waveforms. The slight offset in time is due to the ambiguity associated with deciding whether to plot a time-varying Fourier coefficient at the beginning or at the end of a clock cycle.

Figure D-92 I and Q Baseband Equivalent Outputs

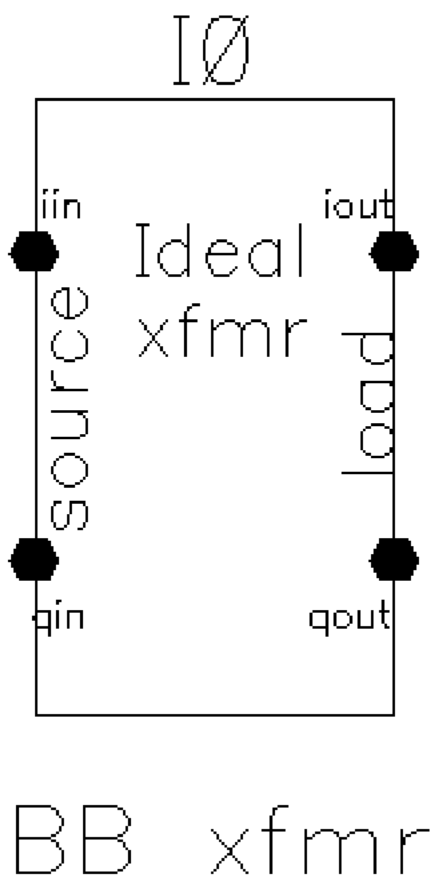


Measurement Blocks

Ideal Transformer

(`BB_xfmr`)

Figure D-93 Ideal Transformer Block Symbol



The ideal transformer is in the *measurement* category of the *rfLib*. It's purpose is to help designers transform between different resistances.

- Parameters: source resistance *Rs* and load resistance *Rl*.
- Inputs = i and q input voltages, i and q output currents.

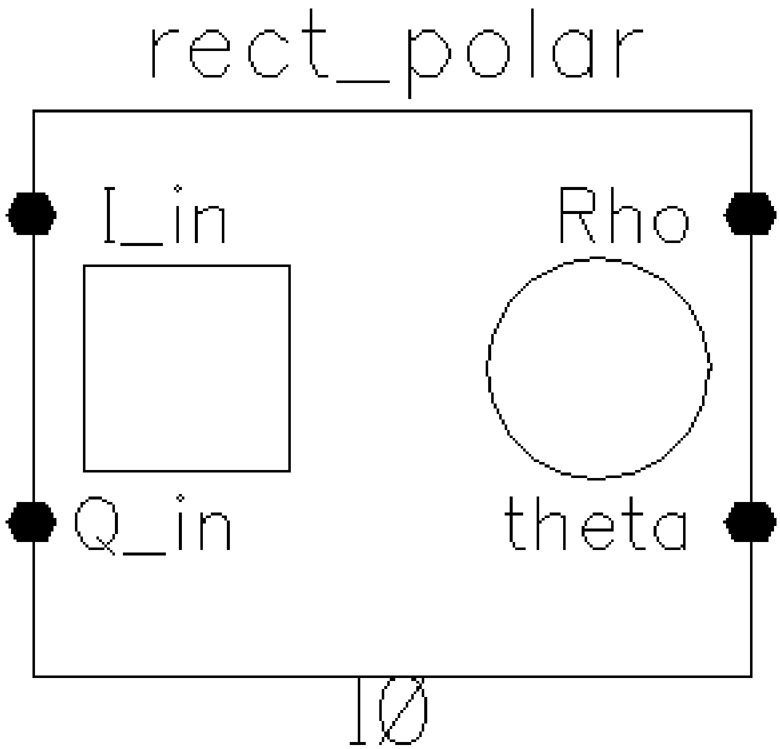
- Outputs = i and q output voltages, i and q input currents defined as follows:

$$\begin{bmatrix} i_{iin} \\ i_{qin} \end{bmatrix} = \sqrt{\frac{Rl}{Rs}} \begin{bmatrix} i_{iout} \\ i_{qout} \end{bmatrix}$$

$$\begin{bmatrix} v_{iout} \\ v_{qout} \end{bmatrix} = \sqrt{\frac{Rl}{Rs}} \begin{bmatrix} v_{iin} \\ v_{qin} \end{bmatrix}$$

Rectangular-to-Polar Transformation

Figure D-94 Rectangular-to-Polar Block Symbol



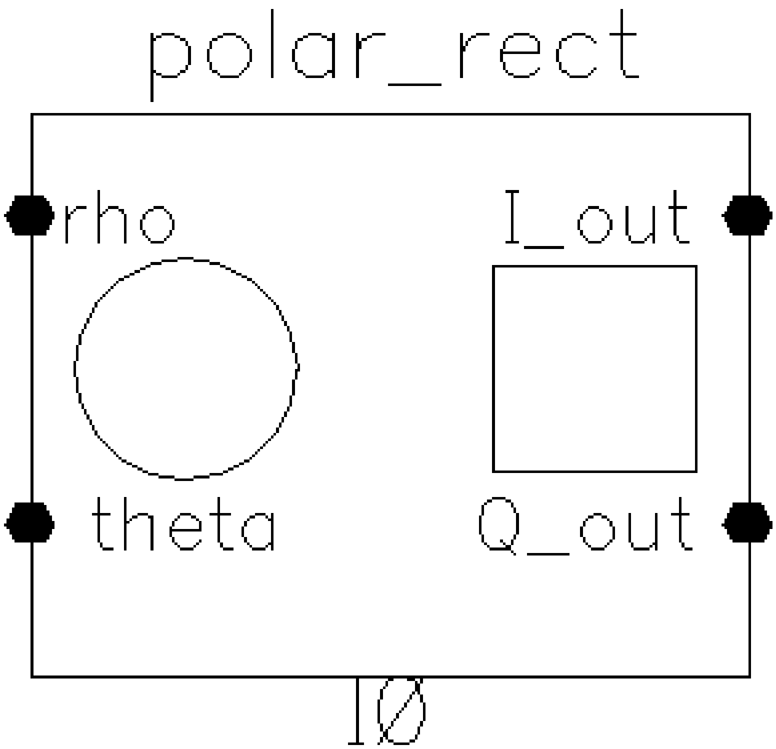
The rectangular-to-polar block is in the “measurement” category. The only parameters are input and output resistances. The inputs are the baseband signal in Cartesian coordinates, the outputs are the baseband signal in polar coordinates.

Parameters: Input and output resistances.

- Inputs = i and q volts.
- Outputs = ρ and θ volts such that
- ρ = Sqrt[i*i+q*q] and
- θ = arctan[q/i] + π*(1+sgn[i])/2 radians, with appropriate checks for the i=0 case.

Polar-to-Rectangular Transformation

Figure D-95 Polar-to-Rectangular Block Symbol



The polar-to-rectangular block is in the “measurement” category. The only parameters are input and output resistances. The inputs are the baseband signal in polar coordinates, the outputs are the baseband signal in rectangular coordinates.

- Parameters: Input and output resistances.
- Inputs = ρ and θ volts.
- Outputs = i and q volts such that
- i = ρ*cos(θ) and
- q = ρ*sin(θ).

Instrumentation and Terminating Blocks

comms_instr

offset_comms_instr

instr_term

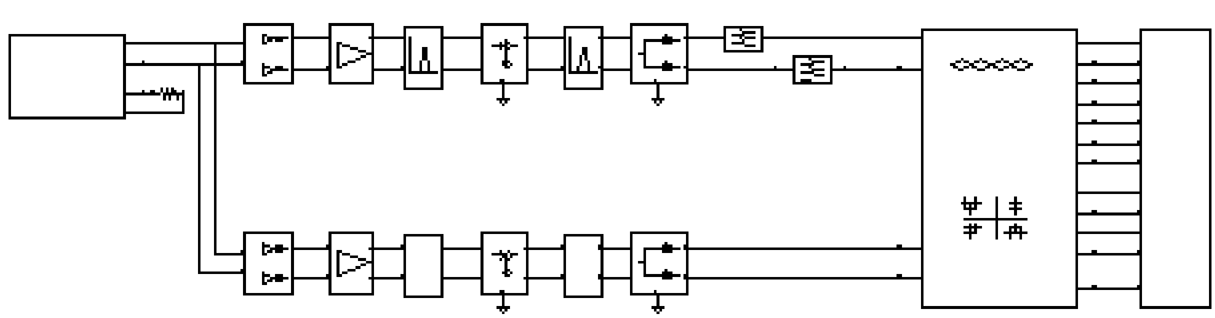
The instrumentation block is in the testbench category and generates waveforms that can be used to create eye-diagrams, eye-diagram statistics, scatter plots, and rms error-vector-magnitude. An example was given in "Use Model and Design Example" on page 926. This section summarizes the parameters and outputs.

There are two kinds of instrumentation blocks, one called *comms_instr* and one called *offset_comms_instr*. The offset block is identical to the first one except the sampling time for scatter plots and eye-diagram statistics are delayed by half a symbol period. The delay makes it possible to plot symbols in an offset QPSK modulation scheme.

There is one other block in the library associated with the two instrumentation blocks, the *instr_term* model. The instr_term block simply loads all instrumentation output pins with 50 Ohms. The *instr_block* simply keeps the schematic editor from complaining about unconnected pins, nothing more.

Figure D-96 on page 1202 shows how the offset_comms_instr and instr_term blocks should be used. The comms_instr block is used similarly. The circuit consists of two branches driven from a single baseband signal generator. The top branch is the non-ideal receiver model, the bottom branch is an ideal version of the top branch. The ideal version is ideal as you like. The ideal branch computes ideal symbol locations in the complex plane. The instrumentation block compares ideal and non-ideal symbols to compute error-vector-magnitude.

Figure D-96 EVM setup



Parameters:

symbols per second. This parameter is necessary for generating the sawtooth which will be used as the x-axis to generate eye-diagrams. It also determines the rate at which the input waveforms will be sampled.

I-sampling delay. This parameter sets the phase of the symbol sampler. It is referenced to the eye-diagram of the I-output. Estimate the optimal delay by doing one simulation just to get the eye-diagram. The optimal delay is the time from the leftmost part of the eye-diagram, which should be zero, to the time at which they eye is widest.

number of symbols. This is the number of symbols to sweep in the eye-diagram. Sweeping two symbols ensures that you will see at least one continuous eye, if it is open.

max, min eye-diagram volts, and number of hstgm bins. These parameters are used to compute the bins which define the eye-diagram histogram. The bin width equals (max voltage-min voltage)/(number of bins). The histogram shows the distribution of the I_in voltage at the sampling instant.

I-noise and Q-noise(volts^2). These parameters set the variance of Gaussian random variables which can be added to the received symbols before anything is computed or plotted.

statistics start time. This parameter delays the start of any statistical computations. The purpose is to exclude start-up transients from the statistics.

input resistance. This parameter is the input resistance of the input terminals of the instrumentation block.

Eye-diagrams. To generate an eye-diagram of the I-input signal, plot the sawtooth and I-eye outputs in one waveform display tool. Change the x-axis to be the sawtooth. This is done through the x-axis menu in the waveform display tool. The procedure for generating an eye-diagram of the Q-output is the same except you use the Q-eye output.

Histograms. You can only generate a histogram of the I-input signal. The histogram shows the distribution of the I-input voltage at the sampling instant. To generate a histogram, plot the eye_hist and eye_count_hist outputs in the same waveform display window. Change the x-axis to be the eye-hist signal then change the plot to *bar*.

Eye-diagram statistics. The ave_eye output is the average absolute value of the I-input signal at the sampling instant. The root_var_eye output is the square root of the variance of the absolute value of the I-input voltage at the sampling instant. The voltages at these output pins represent running estimates of the associated statistics.

Scatter plots. A scatter plot is the I-input and Q-input samples plotted against each other. The scatter plot shows the locations of the received symbols. To generate a scatter plot, plot I_scatter and Q_scatter in the same waveform display window then change the x-axis to be the I_scatter signal. Finally, change the plot to plot data points only. A scatter plot of the reference model can be generated similarly by replacing I_scatter and Q_scatter with Iref_scatter and Qref_scatter.

rms Error Vector Magnitude. The rms EVM is defined as the square root of the sum of the squares of the vectorial differences between the ideal and non-ideal received symbols, normalized to the rms value of the magnitude of the ideal received symbols. The output voltage at this pin is represents a running calculation of the rms EVM. The normalized EVM is in percent.

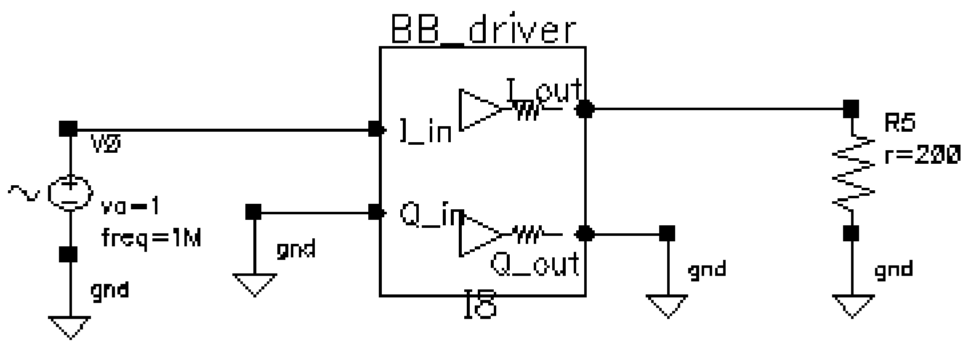
Baseband Drive Signals

Baseband signals can be imported into a Spectre Transient simulation through the Ports by setting the Source type to pwl and specifying the path to a file containing the data. You can also use the ppwlf sources. The ppwlf sources recognize SPW format.

You can also use one of the three baseband signal generators in the "measurement" category, which are described below. The baseband signal generators can drive J-models directly but should be used with a BB_driver model when driving top-down models. The BB_driver model is described first.

BB_driver

Figure D-97 BB_driver



The BB_driver converts peak volts into dBm. It is intended for use with the baseband signal generators. The BB_driver is nothing more than an identical pair of voltage-controlled current sources with identical output resistances. The I_out pin is independent of the Q_in pin and the Q_out pin is independent of the I_in pin. The following example describes what this model does. Figure D-97 on page 1205 shows the set up.

1. Build the schematic. Set the Output resistance parameter to 200 Ohms and the "dBm-out@1v peak in" parameter t0 10.
2. Load one output with 200 Ohms and drive the associated input with a sinusoidal voltage that has a peak voltage of 1 volt.
3. Simulate a few cycles with a Transient analysis. The power delivered to the 200 Ohm load will be 10 dBm, which corresponds to a peak voltage of 2 Volts across the 200 Ohm load.

CDMA_reverse_xmit

Figure D-98 on page 1206 shows the symbol for the CDMA_reverse_xmit generator with the BB_driver. Figure D-99 on page 1207shows the signal trajectory for the CDMA_reverse_xmit generator.

Figure D-98 CDMA_reverse_xmit generator

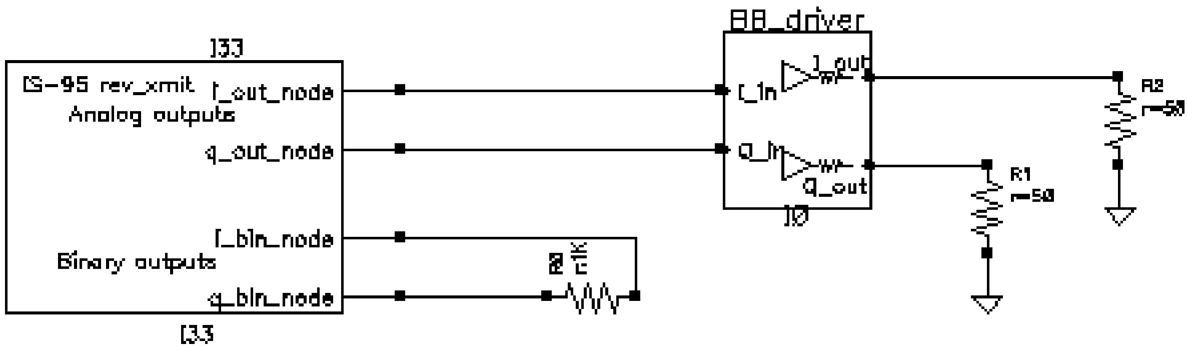
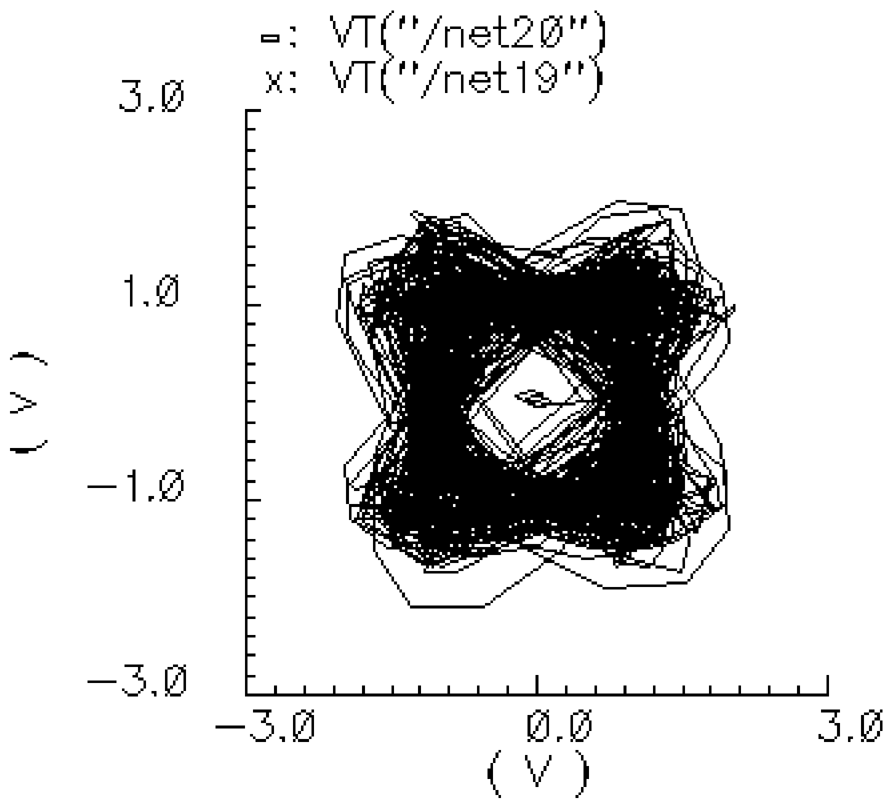


Figure D-99 Signal Trajectory for CDMA_reverse_xmit



This model generates an IS-95 reverse link transmitter signal. The modulation is offset QPSK. A single stream of random bits is spread with two different spreading sequences to generate the I and Q outputs. The spreading sequences are generated with pseudo noise generators that have 15 states. The resulting chip rate is 1.2288 Megachips per second on each output. (A chip is a piece of a bit. It is the product of a bit in the spreading sequence and the original bit that is being spread.) Each chip is over sampled at a 4:1 rate. Each stream of samples drives its own 48-tap FIR filter. The Q-output is delayed by 2 samples, half a chip, to convert the QPSK signal to offset QPSK.

The i_out_node and q_out_node pins produce voltages equal to the digitally filtered I and Q outputs.

The i_bin_node and q_bin_nodes are the binary outputs. The binary outputs are the raw unfiltered chips.

The model has three parameters:

seed: This is the seed for the random number generator that generates the bits.

amplitude: This sets the peak amplitude of the i_out_node and q_out_node pins.

t-rise_fall: The digitally filtered outputs jump from one value to the next in a time specified by this parameter. To generate outputs consisting of data points connected by straight lines, set t-rise_fall to the chip (or symbol) period. For more of a staircase waveform, make this parameter a small fraction of the symbol period.

The spreading sequences and FIR filters can be redefined through the Modelwriter.

GSM_xmtr

Figure D-100 on page 1208 shows the symbol for the GSM_xmtr generator with the BB_driver. Figure D-101 on page 1209 shows the signal trajectory for the GSM_xmtr generator.

Figure D-100 GSM_xmtr generator

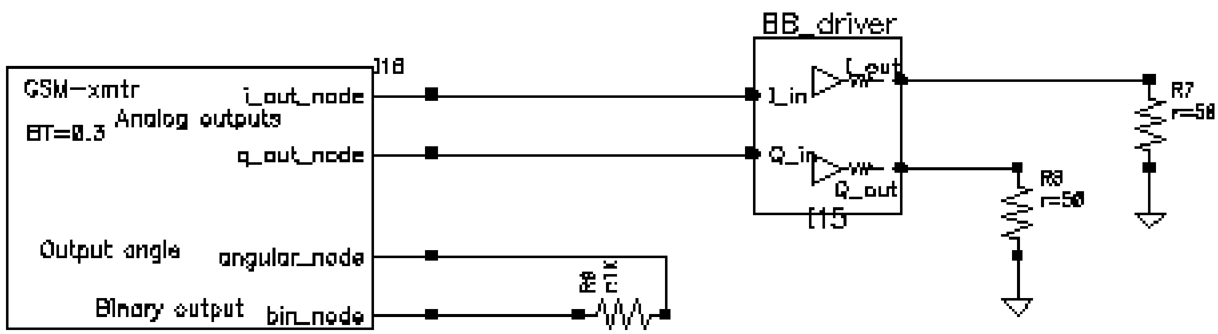
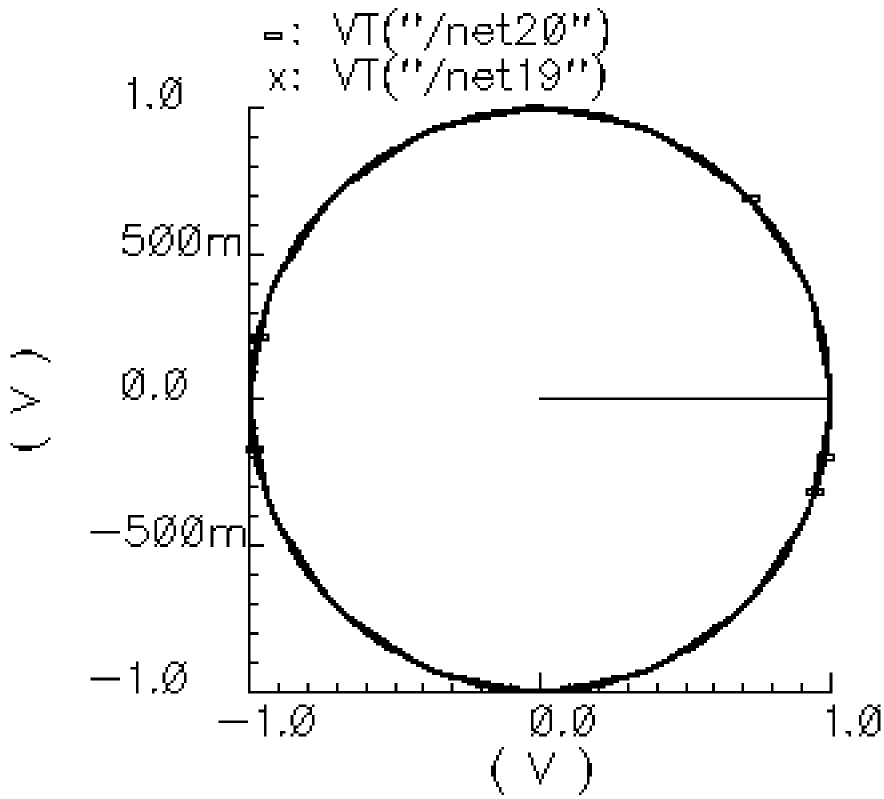


Figure D-101 Signal Trajectory for the GSM_xmtr



The GSM signal generator starts with a bit sequence of three zeros, 8.25 ones, and three zeros. GSM modulation is 0.3 GMSK [11]. This pattern models 3 stop bits, 8.25 guard bits, and 3 start bits. The fractional guard bits are important in eliminating two sharp dips in the power spectral density. The “frame” concludes with 142 random bits. The bit rate is 270833.333 bits per second. Each bit is over sampled at a 4:1 rate then filtered with a Gaussian FIR filter that has 32 taps. The filtered bits modulate the frequency of a VCO. The gain of the VCO equals 0.25*bit_rate*oversample_rate Hz/volt.

The voltage on the i_out_node pin equals the amplitude, a user-specified parameter, scaled by the cosine of the VCO phase.

The voltage on the q_out_node pin equals the amplitude scaled by the sine of the VCO phase.

The angular_node pin produces a voltage numerically equal to the phase of the VCO in radians.

The bin_node is a voltage representing the bits that drive the FIR filter.

Aside from the amplitude parameter, all parameters are used as they are in the CDMA_reverse_xmit model.

The FIR filter can be modified through the Model Writer.

pi_over4_dqpsk

Figure D-102 on page 1210 shows the symbol for the pi_over4_dqpsk with the BB_driver. Figure D-103 on page 1211 shows the signal trajectory for the pi_over4_dqpsk.

Figure D-102 pi_over4_dqpsk generator

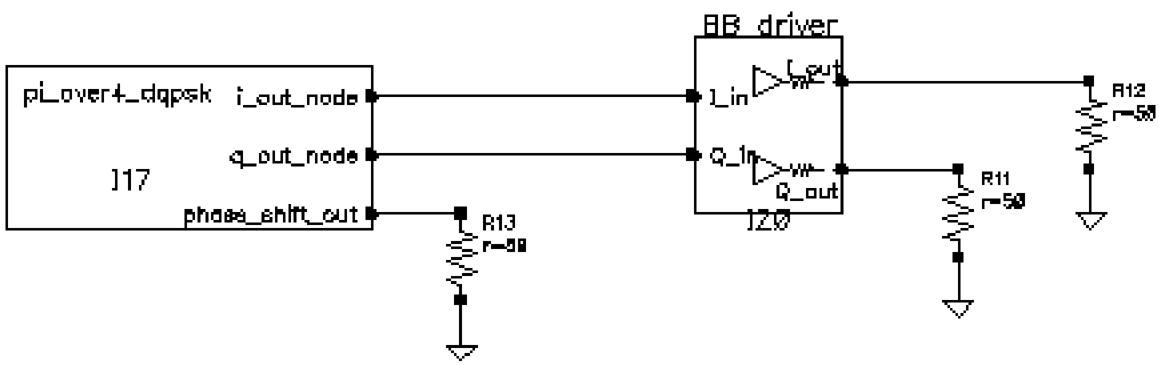
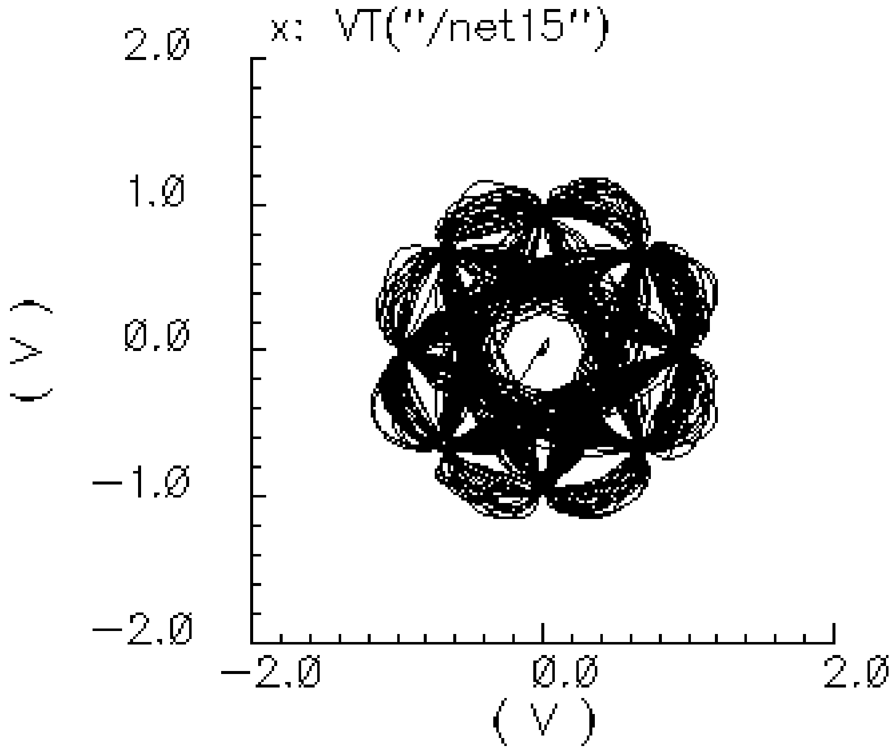


Figure D-103 Signal Trajectory for the pi_over4_dqpsk



In π/4 DQPSK modulation, information rides on one of four possible phase shifts in the carrier. The possible phase shifts are +-45 degrees and +-135 degrees. The model selects one of the four phase shifts at random then computes I and Q pulses that will drive FIR filters. The I-pulse equals the amplitude parameter, scaled by the cosine of the resulting absolute phase. Absolute phase equals the sum of all past phase shifts plus the present one. The initial phase is zero.

The Q-pulse equals the amplitude parameter, scaled by the sine of the absolute phase. The I and Q pulses drive their own 64-tap FIR filters. The I and Q filters are identical. The filters are raised cosine filters with a 0.35 roll off factor. The rate at which the phase shifts occur is 24300 shifts per second. The FIR filters operate at 8 times that rate.

The voltages on the i_out_node and q_out_nodes pins are the filter outputs.

The phase_shift_out pin is a voltage numerically equal to the phase shift.

Aside from the amplitude parameter, all parameters are used as they are in the CDMA_reverse_xmit model.

The FIR filter can be modified through the Model Writer.

References

1. M. Jeruchim, P. Balaban, K. Shanmugan, “Simulation of Communication Systems”, Plenum Press, 1992.
2. E. Lee, D. Messerschmitt, “Digital Communication”, Kluwer Academic Publishers, 1994
3. A. Fitzgerald, C. Kingsley Jr.,S Umans, "Electric Machinery", Fifth edition, McGraw Hill
4. R. Stein and W. Hunt Jr., “Electric Power System Components”, Van Nostrand Reinhold Company, 1979.
5. W. Leonhard, “Control of Electrical Drives”, Springer-Verlag 1990.
6. P. Kraus and O. Wasynchuk, “Electromechanical Motion Device”s, McGraw-Hill Book Company, 1989.
7. P. Vas, “Vector Control of AC Machines”, Oxford University Press, 1990
8. G. Kron, “A Short Course in Tensor Analysis for Electrical Engineers” , John Wiley and Sons, Inc, 1942.
9. J. Chen, “Rotor Reference Frame Models of a Multiloop 2-phase Motor Drive in Brushless DC and MicroStepping Modes”, 1995 Intersociety Energy Conversion Engineering Conference, Orlando, FLA.
10. A. Harrysson, M. Ziren, “System Level Simulations of a Receiver for W-CDMA”, Master of Science Thesis, Department of Applied Electronics, Lund Institute of Technology, 12/99.
11. T. Rappaport, “Wireless Communications, Principles and Practice”. 1996. Prentice Hall.

E

Plotting Spectre S-Parameter Simulation Data

This appendix describes the equations used by the Waveform Calculator to plot data generated by Spectre S-parameter simulations.

Using the Waveform Calculator in the analog design environment, you can plot the following S-parameter data:

- Network Parameters
- Two-Port Scalar Quantities
- Two-Port Gain Quantities
- Two-Port Network Circles
- Equation for VSWR (Voltage Standing Wave Ratio)
- Equation for GD (group delay)

Use the buttons located in the *Function* area of the S-parameter Direct Plot form to specify the type of data to plot.

Network Parameters

You can plot S, Y, Z, and H network parameters.

S-parameters, Y-parameters, and Z-parameters (denoted as SP, YP, and ZP on the Direct Plot form) are defined for circuits with any number of ports. H-parameters (denoted as HP on the Direct Plot form) are defined only for two-port circuits.

You can plot parameters on polar charts, Smith charts, or on rectangular plots after applying a *Modifier* option. The dB conversion uses $20 \log_{10} X$ because the parameters represent scalar ratios (for example, voltage).

Equations for Network Parameters

For the ZP, YP, and HP parameters, Spectre returns S-parameters to the analog design environment. The environment converts them, as needed, to the equivalent Z, Y, and H matrixes using standard published methods. Spectre calculates S-parameter values.

SP (S-parameter) values

SP (S-parameter) values are calculated by Spectre.

ZP (Z-parameter) equation

The Z-parameter equation is as follows

$$Z_m = [Z_{ref}][I + S_m][I - S_m]^{-1}[Z_{ref}]$$

Where

- S_m is the N-port S-parameter matrix
- I is the N x N identity matrix
- Z_{ref} is the characteristic impedance of the port
- Z_m is the resulting Z-parameter matrix

YP (Y-parameter) equations

The Y-parameter equations are as follows

$$Y_m = [Y_{ref}][I - S_m][I + S_m]^{-1}[Y_{ref}]$$

Where

- S_m is the N-port S-parameter matrix
- I is the N x N identity matrix
- Y_{ref} is a diagonal matrix defined as

$$Y_{ref} = \frac{1}{\sqrt{\Re e\{Z_i\}}}$$

Where

- ❍ Z_i is the terminating impedance at port i
- ❍ Y_m is the resulting Y-parameter matrix

The HP (H-parameter) equations

The HP (H-parameter) equations only apply to two-port circuits.

D is

$$D = (1 - S_{11})(1 + S_{22}) + S_{21}S_{12}$$

H_{11} is

$$H_{11} = \frac{[(1 + S_{11})(1 + S_{22}) - S_{21}S_{12}]Z_{ref1}^{2}}{D}$$

H_{21} is

$$H_{21} = \left(\frac{-2S_{21}}{D}\right)\frac{Z_{ref1}}{Z_{ref2}}$$

H_{12} is

$$H_{12} = \left(\frac{2S_{21}}{D}\right)\frac{Z_{ref1}}{Z_{ref2}}$$

and H_{22} is

$$H_{22} = \frac{(1 - S_{11})(1 - S_{22}) - S_{21}S_{12}Z_{ref}^{2}}{D}$$

Where

- Z_{ref1} is the terminating impedance at port 1
- Z_{ref2} is the terminating impedance at port 2

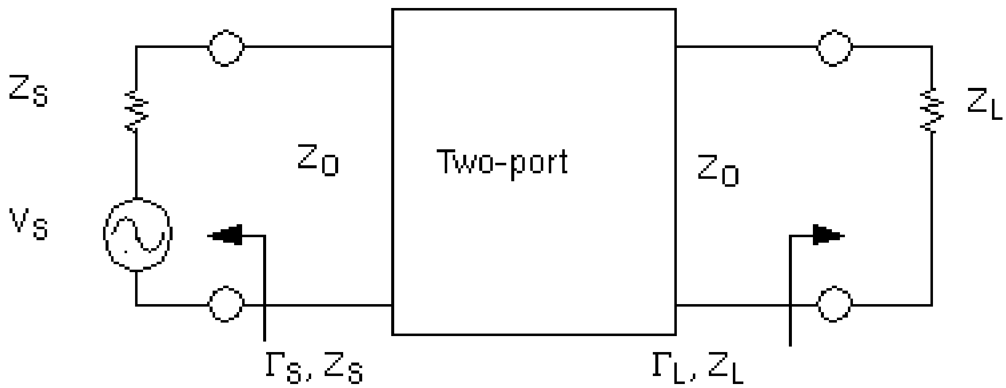
Two-Port Scalar Quantities

The two-port scalar quantities include

- NF_{min} is minimum noise figure
- NF is noise figure
- R_n is equivalent noise resistance
- G_{min} is Optimum Noise reflection coefficient
- B_{1f} is alternative stability factor
- K_f is stability factor

You can plot two-port scalar quantities only against frequency. In addition, you can plot them only on rectangular charts. Figure E-1 illustrates a generic two-port circuit that defines impedances and reflection coefficients.

Figure E-1 A Generic Two-Port Circuit



In Figure E-1

- Z_s is the source impedance
- Γ_s is the input reflection coefficient
- Z_L is the load impedance
- Γ_L is the load reflection coefficient
- Z_O is the characteristic impedance

Equations for Two-Port Scalar Quantities

G_{min} (Optimum Noise Reflection Coefficient) Equation

G_{min} (also known as Γ_{min} in the literature) is the reflection coefficient associated with minimum noise figure. You can plot G_{min} on a Smith chart or, by using the `Modifier` field, on a rectangular chart.

$$G_{min} = \Gamma_{min} = \frac{Z_{on} - Z_o}{Z_{on} + Z_o}$$

Where

- Z_o is the characteristic impedance
- Z_{on} is the source impedance associated with minimum noise figure (NF_{min})

NF_{min} (Minimum Noise Figure) and NF (Noise Figure) Equations

The NF_{min} and NF plots are controlled by the *Modifier* option in the Direct Plot form.

- Plot noise figure (NF) in dB by setting *Modifier* to *dB10*
- Plot noise factor (F) by setting *Magnitude*

Use NF_{min} and NF only for two-port circuits.

$$NF = 10\log_{10}F$$

Where

- F is the noise factor
- NF_{min} is the minimum noise figure

The NF_{min} values are calculated by Spectre

The NF (noise figure) equation is calculated by the analog design environment from NF_{min}, G_{min} (Γ_{min}), and r_n. You can specify the optional source reflection coefficient Γ_S as an argument if you use the analog design environment Waveform Calculator. From the Direct Plot form, the analog design environment assumes Γ_S to be 0 (input matched to reference termination).

$$NF = NF_{min} + \frac{4r_n\left|r_s - r_{min}\right|^2}{\left(1 - \left|r_s\right|^2\left|1 + r_{min}\right|^2\right)}$$

Where

$$r_n = \frac{R_n}{Z_o}$$

Here r_n is the *normalized* equivalent noise resistance.

Rn (equivalent noise resistance)

R_n plots equivalent noise resistance. The R_n values are calculated by Spectre.

Stability Factors

K_f and B_{1f} plot the Rollet stability factor and its intermediate term. Use these parameters only for two-port circuits.

D is

$$D = S_{11}S_{22} - S_{21}S_{12}$$

K_f is

$$K_f = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |D|^2}{2|S_{21}||S_{12}|}$$

and B_{1f} is

$$B_{1f} = 1 + |S11|^2 - |S22|^2 - |D|^2$$

Two-Port Gain Quantities

The following gain quantities are valid only for two-port circuits

G_A (available gain) is the power gain obtained by optimally matching the output of the network.

G_P (power gain) is the power gain obtained by optimally matching the input of the network.

G_T (transducer gain) shows the insertion effect of a two-port circuit. This quantity is used in amplifier design.

G_{umx} (maximum unilateral transducer power gain)

G_{max} (maximum available gain) shows the transducer power gain when there exists a simultaneous conjugate match at both ports.

G_{msg} (maximum stable gain) shows the gain that can be achieved by resistively loading the two-port such that $k = 1$ and then simultaneously conjugately matching the input and output ports. For conditionally stable two-ports, you can approach the maximum stable gain as you reduce the input and output mismatch. If you attempt a simultaneous conjugate match and $k < 1$, the two-port will oscillate.

Equations for Two-Port Gain Calculations

G_A (Available Gain) Equations

Available gain equations for output conjugately matched.

$$G_A = \frac{|S_{21}|^2 \left(1 - |\Gamma_S|^2\right)}{|1 - S_{11}\Gamma_S|^2 \left(1 - |\Gamma_2|^2\right)}$$

Where

$$\Gamma_2 = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1 - S_{11}\Gamma_S}$$

Note: When you use the S-parameter Direct Plot form, G_S is set to zero, and therefore available gain (G_A) is plotted as

$$G_A = \frac{|S_{21}|^2}{1 - |S_{22}|^2}$$

To plot G_A for nonzero values of G_S, use the analog design environment Waveform Calculator.

G_P (Power Gain) Equations

Power gain equations for input conjugately matched.

$$G_P = \frac{|S_{21}|^2\left(1-|\Gamma_L|^2\right)}{|1-S_{22}\Gamma_L|^2\left(1-|\Gamma_1|^2\right)}$$

Where

$$\Gamma_1 = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1-S_{22}\Gamma_L}$$

Note: When you use the S-parameter Direct Plot form, G_L is set to zero, and therefore the power gain, G_P, is plotted as

$$G_P = \frac{|S_{21}|^2}{1-|S_{11}|^2}$$

To plot G_P for nonzero values of Γ_L, use the analog design environment Waveform Calculator.

G_T (Transducer Gain) Equations

$$G_T = \frac{\left(1-|\Gamma_S|^2\right)|S_{21}|^2\left(1-|\Gamma_L|^2\right)}{|(1-S_{11}\Gamma_S)(1-S_{22}\Gamma_L)-S_{12}S_{21}\Gamma_S\Gamma_L|^2}$$

Note: When using the S-parameter Direct Plot form, the analog design environment assumes that the source (Γ_S) and load (Γ_L) reflection coefficients are zero. G_T, therefore, plots the insertion gain.

$$G_T = |S_{21}|^2$$

Using the Waveform Calculator, you can plot G_T and specify the source and load terminations.

G_{max} (Maximum Available Gain) Equations

For K >1

$$G_{max} = \left|\frac{S_{21}}{S_{12}}\right|\left[K - \sqrt{K^2 - 1}\right]$$

For K<=1

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}$$

Where K is the stability factor, K_f

G_{umx} (Maximum Unilateral Power Gain) Equation

$$G_{umx} = \frac{|S_{21}|^2}{(1 - |S_{11}|^2)(1 - |S_{22}|^2)}$$

G_{msg} (Maximum Stable Power Gain) Equation

$$G_{msg} = \frac{|S_{21}|}{|S_{12}|}$$

Two-Port Network Circles

NC plots constant noise contours at the input of a two-port circuit. GAC plots constant gain contours at the input port, and GPC plots constant gain contours at the output port. Gain contour values reflect an optimum match at the opposing port.

Noise and Gain circles can be plotted at a single dB value for a range of frequencies or at a single frequency for a range of dB values. If you do not enter values for the frequency range, a circle is plotted for every simulated frequency for which a circle with the specified value exists.

SSB plots stability circles at the input port, and LSB plots stability circles at the output port. You can also specify a limited frequency range for these contours.

Equations for Two-Port Network Circle

NC (Noise Circle) Equations

$$N_i = \frac{(F_i - F_{min})\left|1 + \Gamma_{min}\right|^2}{4r_n}$$

where

$$\Gamma_{min} = G_{min}$$

The center is calculated using

$$C_N = \frac{\Gamma_{min}}{1 + N_i}$$

The radius is calculated using

$$r_N = \sqrt{\frac{N_i^2 + N_i\left(1 - \left|\Gamma_{min}\right|^2\right)}{1 + N_i}}$$

Where i is the index number

GAC (Available Gain Circle) Equations

The center is calculated using

$$C_A = \frac{g_a(S_{11}{}^* - D^* S_{22})}{1 + g_a\left(\left|S_{11}\right|^2 - |D|^2\right)}$$

The radius is calculated using

$$r_A = \frac{\sqrt{1 - 2K_f\left|S_{21}S_{12}\right|g_a + \left|S_{12}S_{21}\right|^2 g_a^2}}{\left|1 + g_a\left(\left|S_{11}\right|^2 - |D|\right)^2\right|}$$

Where

$$G_a = \frac{G_A}{\left|S_{21}\right|^2}$$

And

$$D = S_{11}S_{22} - S_{12}S_{21}$$

GPC (Power Gain Circle) Equations

The center is calculated

$$C_P = \frac{g_p(S_{22}^* - D^* S_{11})}{1 + g_p\left(|S_{22}|^2 - |D|^2\right)}$$

The radius is calculated using

$$r_P = \frac{\sqrt{1 - 2K_f|S_{21}S_{12}|g_p + |S_{12}S_{21}|^2 g_p^2}}{1 + g_p\left(|S_{22}|^2 - |D|^2\right)}$$

Where

$$g_p = \frac{G_P}{|S_{21}|^2}$$

And

$$D = S_{11}S_{22} - S_{12}S_{21}$$

LSB (Load Stability Circle) Equations

The center is calculated using

$$C_L = \frac{S_{11}D^* - S_{22}^*}{|D|^2 - |S_{22}|^2}$$

The radius is calculated using

$$r_L = \left|\frac{S_{12}S_{21}}{|D|^2 - |S_{22}|^2}\right|$$

Where

$$D = S_{11}S_{22} - S_{12}S_{21}$$

SSB (Source Stability Circle) Equations

The center is calculated using

$$C_S = \frac{S_{22}D^* - S_{11}^*}{|D|^2 - |S_{11}|^2}$$

The radius is calculated using

$$R_S = \left|\frac{S_{12}S_{21}}{|D|^2 - |S_{11}|^2}\right|$$

Where

$$D = S_{11}S_{22} - S_{12}S_{21}$$

Equation for VSWR (Voltage Standing Wave Ratio)

VSWR is calculated from the S-parameters. You can plot the VSWR at any port in the circuit on a rectangular chart.

$$VSWR_i = \frac{1 + |S_{ii}|}{1 - |S_{ii}|}$$

Where i is the port number.

Equation for ZM (Input Impedance)

You can plot input impedance if all other ports are matched

$$Z_m = \frac{1 + S_{ii}}{1 - S_{ii}} R$$

Where R the reference impedance of the port of interest and i is the port number

Equation for GD (group delay)

GD (group delay) approximates the derivative of the phase with respect to frequency, normalized to 360 degrees. Units for group delay are in seconds.

$$G_d \cong \frac{-d\phi}{d\omega}$$

Group delay is calculated from the phase of the corresponding S-parameter (for example, GD_{21} corresponds to S_{21}).

F

Using QPSS Analysis Effectively

Quasi-Periodic Steady-State (QPSS) analysis computes the quasi-periodic steady-state responses of circuits with multiple periodic inputs of differing frequencies. The QPSS analysis is a prerequisite for all quasi-periodic small-signal analyses such as the quasi-periodic AC (QPAC), quasi-periodic transfer function (QPXF), quasi-periodic S-Parameter (QPSP), and quasi-periodic noise (QPnoise) analyses provided by the SpectreRF circuit simulator.

A quasi-periodic signal has multiple fundamental frequencies. Closely spaced or incommensurate fundamentals cannot be efficiently resolved by PSS. QPSS allows you to compute responses to several moderately large input signals in addition to a strongly nonlinear tone which represents the LO or clock signal. The circuit is assumed to respond in a strongly nonlinear fashion to the large tone and in a weakly nonlinear fashion to the moderate tones. You might use a QPSS analysis, for example, to model intermodulation distortion with two moderate input signals. QPSS treats one of the input signals (usually the one that causes the most nonlinearity or the largest response) as the large signal, and the others as moderate signals.

The QPSS analysis employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. When input signals are stiff, that is, when they have large time constants, the QPSS analysis with its MFT algorithm is more efficient than transient circuit simulation. MFT also performs better than traditional algorithms such as harmonic balance when solving highly nonlinear problems.

A typical application of QPSS analysis is to predict the harmonic distortion of switched-capacitor filters that operate under widely separated fundamentals. See an example in “Switched Capacitor Filter Example” on page 1240.

QPSS can also predict intermodulation distortion of a narrowband circuit that is driven by a local oscillator (LO) and two high-frequency input fundamentals that are closely spaced in frequency, such as in the down conversion stage of a receiver. See an example in “High-Performance Receiver Example” on page 1241.

This appendix contains the following:

- A discussion of when to use the QPSS analysis

- A brief description of the MFT method
- A comparison of the QPSS and PSS analyses
- A comparison of the QPSS and PAC analyses
- Two examples of typical use: a switched capacitor filter and a high-performance receiver
- Procedures for setting up and running QPSS analysis

When Should You Use QPSS Analysis

The increasing demand for low-cost, mobile communication systems increases the need for efficient and accurate simulation algorithms for RF communication circuits. These circuits are difficult to simulate because they process modulated carrier signals that consist of a high-frequency carrier and a low-frequency modulation signal. Typically, the carrier frequency ranges from 1-5 GHz while the modulation frequency ranges from 10 kHz to 1 MHz. If you were to apply a standard transient analysis to such a circuit, it might require simulating the detailed response of the circuit over hundreds of thousands of clock cycles (or millions of timepoints), a generally impractical approach.

Fortunately, many RF circuits of interest operate near a time-varying, but periodic, operating point. You can analyze some of these circuits if you assume that one of the circuit inputs produces a periodic response that you can calculate using a PSS analysis. You can treat other time-varying circuit inputs as small signals and linearize the circuit around the periodic operating point. You can then analyze small signals efficiently using the SpectreRF Periodic AC (PAC) analysis [8]. This approach lets you avoid long simulation times with transient analysis [7]. See "QPSS and PSS/PAC Analyses Compared" on page 1238 for more information.

However, many RF circuits cannot be analyzed efficiently with the periodic-operating-point plus small-signal approach. For example, predicting the intermodulation distortion of a narrowband circuit, such as a receiver down converter (a mixer followed by a filter), requires calculating the nonlinear response of the mixer circuit, driven by a LO, to two closely spaced high-frequency inputs. The response to both inputs is within the band width of the filter. The steady-state response of such a circuit is quasi-periodic.

As a further complication, many multi-timescale circuits, such as mixers and switched-capacitor filters, have a highly nonlinear response with respect to one or more of the exciting inputs. Consequently, steady-state approaches such as multi frequency harmonic balance do not perform well.

The mixed frequency-time (MFT) approach used for QPSS analysis avoids these difficulties. MFT methods assume that many circuits of engineering interest have a strongly nonlinear response to only one input, such as the clock in a switched-capacitor circuit or the LO in a mixer, and respond in a weakly nonlinear manner to other inputs.

Compared to previous MFT methods [2, 4], the MFT algorithm used for QPSS analysis has the following advantages.

The MFT algorithm used for QPSS analysis

- Avoids the ill-conditioning caused by poorly chosen boundary conditions found in previous algorithms
- Uses a multi-dimensional discrete Fourier transform (DFT) scheme for cycle placement
- Uses a continuation method to enhance the global convergence of Newton's method
- Uses a matrix-implicit, Krylov-subspace-based iterative scheme that enables MFT methods to solve large problems
- Uses a preconditioning strategy that permits the iterative solver to converge rapidly

If you are unfamiliar with terminology such as *matrix-implicit*, *Krylov-subspace*, and *preconditioning,* you can find detailed descriptions in reference [6]. You can find an introduction to *continuation methods* in reference [1].

Essentials of the MFT Method

Circuit behavior is usually described by a set of nonlinear differential-algebraic equations (DAEs) that can be written as,

(F-1) $$\frac{d}{dt}Q(v(t)) + I(v(t)) + u(t) = 0$$

Where

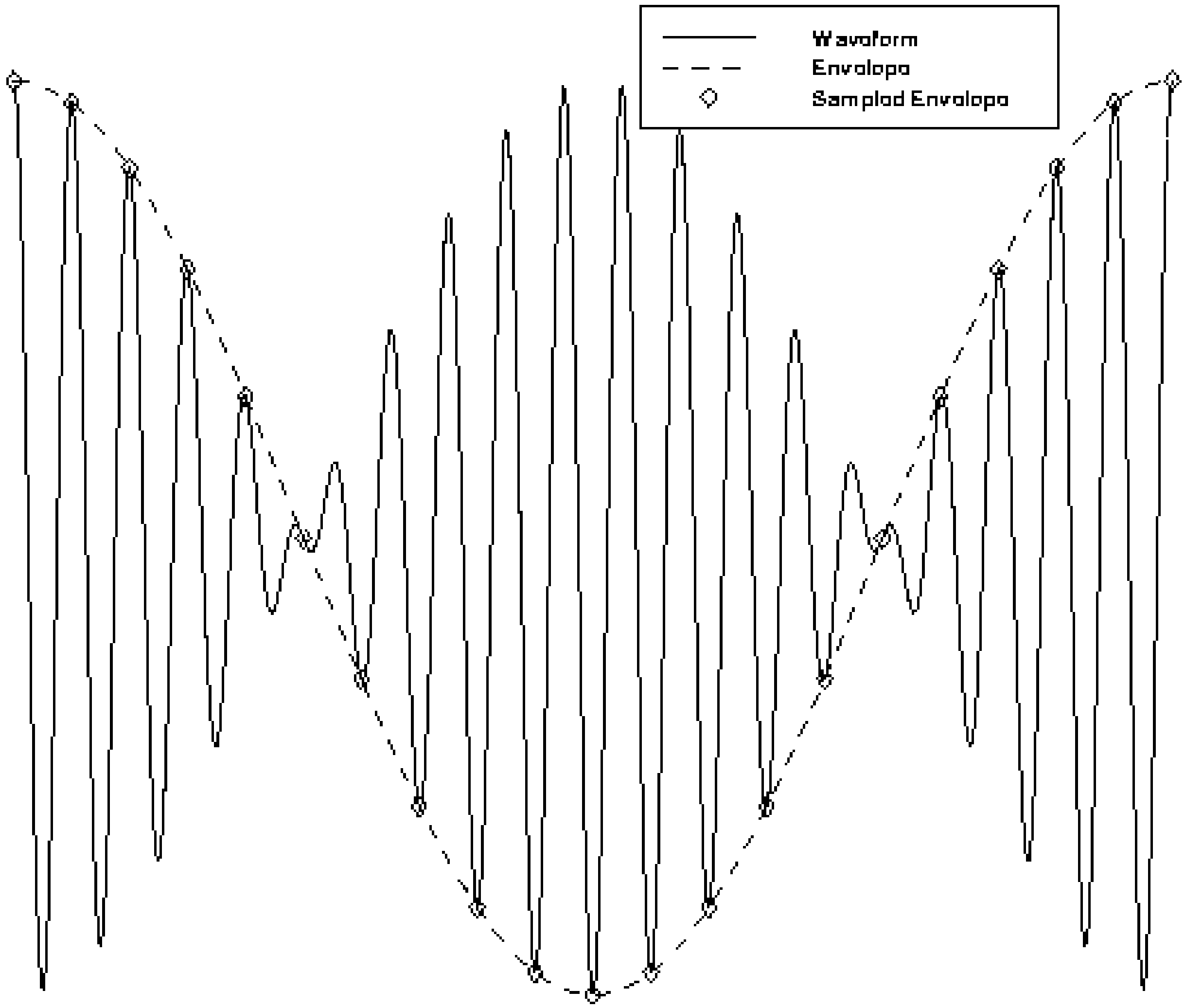
- $Q(v(t)) \in \Re^N$ is typically the vector of sums of capacitor charges at each node
- $I(v(t)) \in \Re^N$ is the vector of sums of resistive currents at each node
- $u(t) \in \Re^N$ is the vector of inputs
- $v(t) \in \Re^N$ is the vector of node voltages
- N is the number of circuit nodes

The MFT algorithm assumes that the circuit is in quasi-periodic steady-state; that is, that the signals can be represented as,

(F-2) $$v(t) = \sum_k \sum_l V_{kl} e^{j2\pi(lf_0 + kf_1)t}$$

where, for simplicity, there are only two fundamental frequencies, f_0 and f_1. The signal $v(t)$ is then sampled at one of the fundamental frequencies, f_0, which is called the *clock* signal. This is shown in Figure F-1 on page 1233, where sampling a two-fundamental quasi-periodic signal at one of the fundamental frequencies creates a sampled waveform that is one-fundamental quasi-periodic, or simply-periodic. MFT directly finds the solution that, when sampled at f_0, is periodic in f_1.

Figure F-1 Sample Envelope Shown Sampled at the Waveform Peaks



The sample envelope shown in Figure F-1 is the waveform traced out when the signal is sampled with the clock period. The envelope is shown sampled at the peaks but this is not necessary.

The sampled waveform is,

(F-3) $$\bar{v}_n = v(nT_0) = \sum_{k=-\infty}^{\infty} \bar{V}_k e^{j2\pi k f_1 t}$$

Where

$T_0 = 1/f_0$

Alternatively, you can also write,

(F-4) $\bar{v}_n = F^{-1}\bar{V}$

which states that

$\bar{v}$

is the inverse Fourier transform of

$\bar{V}$

Recall that v is a solution of the circuit equations and that

$\bar{v}$

is simply v uniformly sampled, so given

$\bar{v}_n$

you can compute a subsequent sample point

$\bar{v}_{n+1}$

using Equation F-5,

(F-5) $\bar{v}_{n+1} = \phi(\bar{v}_n, nT_0, (n+1)T_0)$

In Equation F-5,

- ϕ is the state transition function for the circuit

- $\phi(v_{0,} t_{0,} t_1)$ is the solution for the circuit equations at t_1, given that it starts from the initial condition v_0 at t_0.

Consider the n^{th} sample interval. Then let

$$x_n = \bar{v}_n$$

be the solution at the start of the interval and let

$$y_n = \bar{v}_{n+1} = x_{n+1}$$

be the solution at the end of the interval. Equation F-5 uses the circuit equations to relate the solution at both ends of the interval as shown in Equation F-6.

(F-6) $$y_n = \phi(\bar{v}_n, nT_0, (n+1)T_0)$$

Let $X = F_{\text{x}}$ and $Y = F_{\text{y}}$ (X and Y are the Fourier transforms of x and y). Then, from Equation F-3 and because $y_{\text{n}} = x_{\text{n+1}}$,

(F-7) $$X_k = e^{-j2\pi k f_1 T_0} Y_k$$

Or

(F-8) $$X = D_{T_0} Y$$

Where

$$D_{T_0}$$

is a diagonal delay matrix with

$$e^{-j2\pi k f_1 T_0}$$

being the k^{th} diagonal element.

Together, Equations F-6 and F-8 make up the MFT method.

In practice,

$$\bar{V} = X$$

is band-limited, so only a finite number of harmonics are needed. In addition, if the circuit is driven with one large high-frequency signal at f_0, which is called the *clock* signal, and one moderately-sized sinusoid at f_1, only K harmonics are needed and the method is efficient. With only K harmonics, Equation F-6 on page 1235 is evaluated over $2K + 1$ distinct intervals that are spread evenly over one period of the lowest beat tone. Therefore, the total simulation time is proportional to the number of harmonics needed to represent the sampled waveform. The simulation time is independent of the period of the lowest-frequency beat tone or the harmonics needed to represent the clock signal.

Equation F-8 on page 1235, along with the associated Fourier transforms, relate the starting and ending points of the solution of the circuit equations over each interval, and consequently represent a boundary-value constraint on Equation F-6 on page 1235.

Shooting methods are the most common method for solving boundary-value problems. Their use of transient analysis to solve the circuit equations over an interval brings two important benefits.

1. Transient analysis handles abruptly discontinuous signals efficiently because the timestep shrinks to follow rapid transitions.
2. Transient analysis easily handles the strongly nonlinear behavior of the circuit as it responds to the large clock signal.

QPSS and PSS Analyses Compared

Like PSS analysis, QPSS analysis uses a shooting Newton method. However, instead of doing a single transient integration, each Newton iteration does transient integrations over a number of nonadjacent clock periods. Each of the integrations differs by a phase-shift in each moderate input signal. The number of integrations is determined by the number~~s~~ of harmonics of moderate fundamentals that you specify. Given `maxharms` = [`k1`, `k2`, ..., `kS`], the total number of integrations is

$$\prod_{s=1}^{S} (2k_s + 1)$$

Consequently, the efficiency of the algorithm depends significantly on the number of harmonics required to model the responses of moderate fundamentals.

Fortunately, the number of harmonics of the clock does not significantly affect the efficiency of the shooting algorithm. The boundary conditions of a shooting interval are such that the time-domain integrations are consistent with a frequency-domain transformation with a shift of one large-signal period.

As a SpectreRF user, you might need to run a PSS analysis to calculate the steady-state operating point of circuits with multiple input frequencies. You can do this by making the PSS beat frequency the greatest common factor of all the input frequencies.

When the greatest common factor is relatively close to the clock frequency; for example, within a factor of `10`, PSS analysis might be preferable to QPSS analysis for two reasons.

1. Each nonlinear iteration would not require excessive integrations of clock period.
2. PSS analysis solves a much smaller linear system in each nonlinear iteration.

The size of the linear system resulting from QPSS analysis is

$$K = \prod_{s=1}^{S} (2k_s + 1)$$

times as big as that resulting from PSS analysis.

In general,

- If the ratio of *clock frequency/beat frequency* for a PSS analysis is smaller than K, a PSS analysis might be preferable to a QPSS analysis.
- For circuits such as switched-capacitor filters that operate on wide timescales, where the ratio of *clock frequency/beat frequency* for PSS analysis can be greater than 1000 — QPSS is clearly the analysis of choice.

Additional differences between the PSS and QPSS analyses are that PSS analysis does not have any restrictions on input sources, whereas QPSS analysis requires that all nonclock inputs must be sinusoidal. Also the QPSS analysis cannot be applied to autonomous circuits.

Like the PSS analysis and the periodic small-signal analyses, you must run the QPSS analysis to determine the quasi-periodic operating point before you can run the quasi-periodic small-signal analyses.

QPSS and PSS/PAC Analyses Compared

Like PAC analysis, the QPSS analysis calculates responses of a circuit that exhibits frequency translations. However, instead of having small-signal linear behavior, QPSS models the response as having components of a few harmonics of input-signal frequencies. This permits computing responses to moderately large input signals.

PAC analysis assumes that only the clock is generating harmonics. For example, for a clock frequency f_c, and a small-signal frequency f_s, amplitudes of circuit response are generated at $f_s \quad k_c f_c$ where k_c is bonded by the parameter `harms` in PSS analysis. In contrast, QPSS also permits nonclock fundamentals to generate harmonics. In the same situation, a spectrum at frequencies $k_s f_s \quad k_c f_c$ is generated, where k_s and k_c are bonded by the QPSS parameter `maxharms`.

QPSS Analysis Parameters

While QPSS analysis inherits most PSS parameters directly, the QPSS analysis adds two new parameters and extends the meaning of a few parameters. The two new parameters are the most important QPSS parameters.

- `funds`
- `maxharms`

They replace the PSS parameters, `fund` (or `period`) and `harms`, respectively.

The `funds` parameter accepts a list of names of fundamentals that are present in the sources. You specify these names in the sources using the `fundname` parameter.The simulator figures out the frequencies associated with the fundamental names.

An important feature of the `funds` parameter is that each input signal can be composed of more than one source. However, these sources must all have the same fundamental name. For each fundamental name, its fundamental frequency is the greatest common factor of all frequencies associated with the name.

The first fundamental is considered as the large signal. You can use a few heuristics to pick the large fundamental.

- Pick the fundamental which is not sinusoidal.

- Pick the fundamental which causes the most nonlinearity.
- Pick the fundamental which causes the largest response.

The `maxharms` parameter accepts a list of numbers of harmonics needed for each fundamental.

If you do not list all the fundamental names using the `funds` parameter, the current analysis is skipped. However, if you do not specify `maxharms`, a warning message is issued, and the number of harmonics defaults to 1 for each fundamentals.

QPSS analysis expands the role of two PSS parameters.

- `maxperiods`
- `tstab`

The maxperiods parameter that controls the maximum number of shooting iterations for PSS analysis also controls the maximum number of shooting iterations for QPSS analysis.

The tstab parameter controls both the length of the initial transient integration, while only the clock tone is active, and the number of stabilizing iterations, while both the clock tone and the moderate tones are active. The stabilizing iterations run before the Newton iterations begin.

The remaining QPSS analysis parameters are inherited directly from PSS analysis, and their meanings remain essentially unchanged.

The `errpreset` parameter quickly adjusts several simulation parameters. In most cases, `errpreset` should be the only parameter you need to adjust. See “The errpreset Parameter in QPSS Analysis” on page 78 for information about the `errpreset` parameter.

Application Examples

QPSS analysis is particularly efficient for circuits operating on widely separated timescales. This is demonstrated by the following two examples

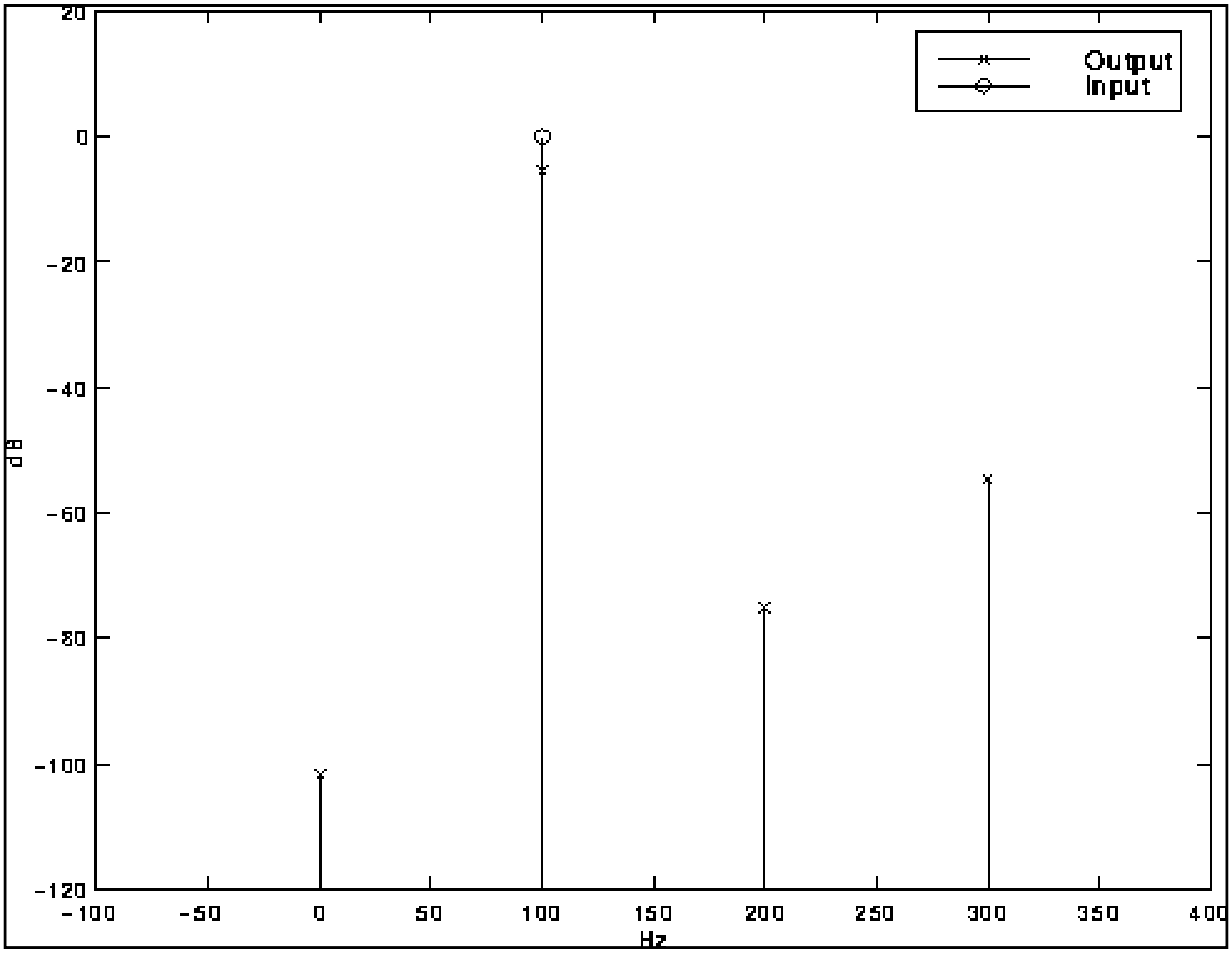
- A switched capacitor filter
- A high-performance receiver

Switched Capacitor Filter Example

The low-pass switched-capacitor filter example has a 4 kHz bandwidth and 238 nodes, resulting in 337 equations. To analyze this circuit, the QPSS analysis was performed with an 8-phase 100 kHz clock and a 1*V* sinusoidal input at 100 Hz.

The 1000-to-1 clock-to-signal ratio makes this circuit difficult for traditional circuit simulators to analyze. Three harmonics were used to model the input signal. The eight-phase clock required about 1250 timepoints for each transient integration. The total number of variables solved by the analysis is 337 X (2 X 3 +1) X 1250 = 2,948,750, slightly less than three million. The simulation completes in less than 20 minutes on a Sun UltraSparc1 workstation with 128 Megabyte memory and a 167 MHz CPU clock. A swap file is used because the analysis cannot be finished in core. For more information, see “Memory Management” on page 1248. Figure F-2 on page 1241 shows the harmonic distortion.

Figure F-2 Harmonic Distortion of a Switched Capacitor Filter



High-Performance Receiver Example

The high-performance image rejection receiver example consists of a low-noise amplifier, a splitting network, two double-balanced mixers, and two broad-band Hilbert transform output filters combined with a summing network that suppresses the unwanted sideband. A limiter in the LO path controls the amplitude of the LO. It is a rather large RF circuit that contains 167 bipolar transistors and uses 378 nodes. This circuit generated 987 equations in the simulator.

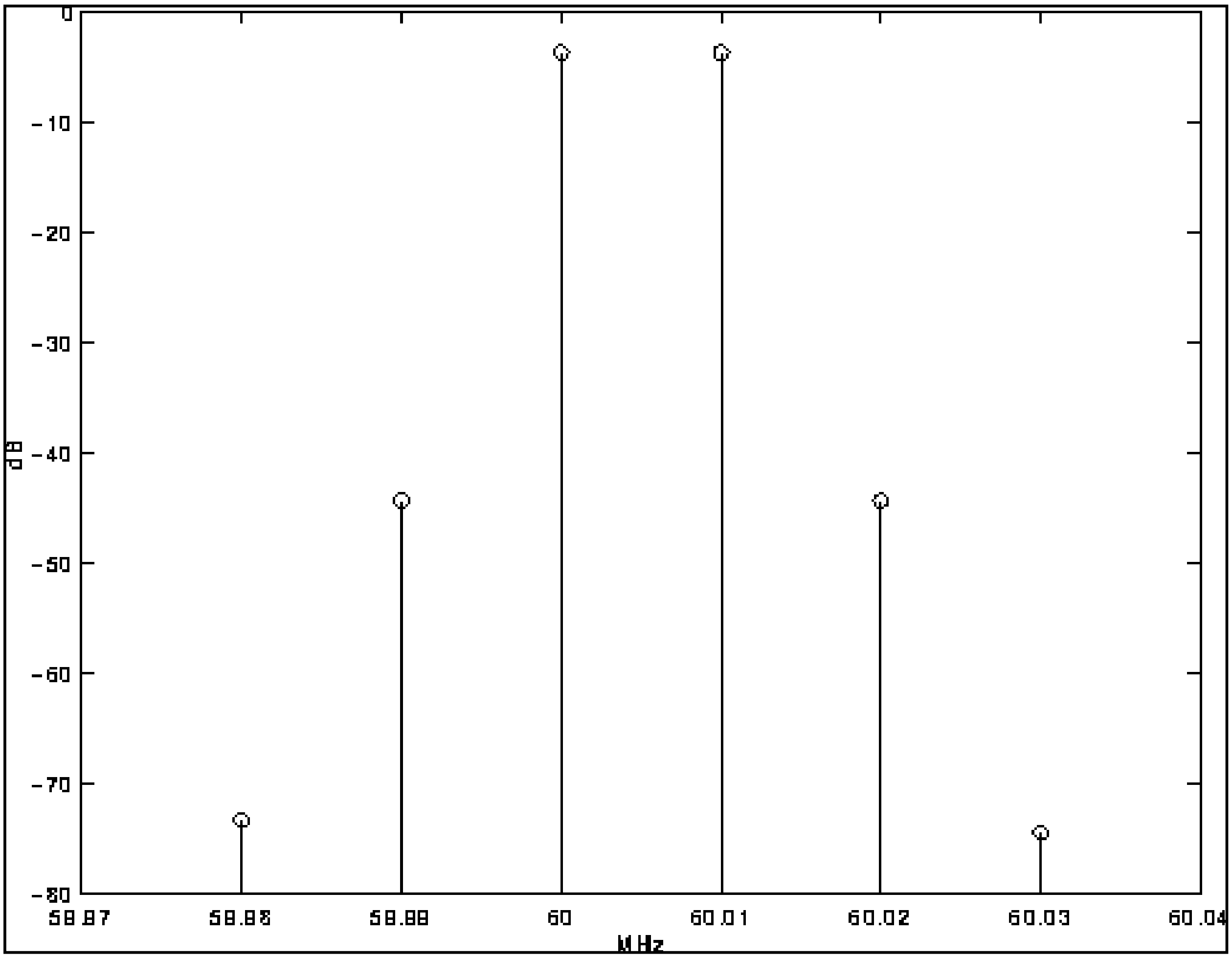
To determine the intermodulation distortion characteristics, the circuit is driven by a 780 MHz LO and two 50 mV closely placed RF inputs, at 840 MHz and 840 MHz+10 KHz, respectively. Three harmonics are used to model each of the RF signals, and 200 time points are used in each transient clock-cycle integration, considered to be a conservative accuracy specification for this circuit. As a consequence, about 10 million unknowns are generated

$$987 \times (2 \times 3 + 1)^2 \times 200 = 9{,}672{,}600$$

The simulation requires 55 CPU minutes on a Sun UltraSparc10 workstation with 128 megabytes of physical memory and a 300 MHz CPU clock. A swap file is used because the analysis cannot be finished in core. For more information, see "Memory Management" on page 1248. Figure F-3 on page 1243 shows the 3rd and 5th order distortion products.

To appreciate the efficiency of the MFT method, consider that traditional transient analysis needs at least 80,000 cycles of the LO to compute the distortion, a simulation time of over two days. Additionally, the results might be inaccurate because of the large numerical error accumulated by integrating over so many cycles. In contrast, the MFT method is able to resolve very small signal levels, such as the 5th order distortion products shown in Figure F-3.

Figure F-3 Intermodulation Distortion of a High-Performance Receiver



Running a QPSS Analysis

The following sections describe how to set up and run a QPSS analysis. They also present ways to promote convergence.

Picking the Large Fundamental

Your first task is to select the large fundamental, called the clock or the LO. Below are a few guidelines for selecting the large fundamental.

- Choose the one that is not sinusoidal.

- Choose the one that causes the most nonlinearity.
- Choose the one that causes the largest response.
- Choose the one that has the highest frequency.

Setting Up Sources

You can specify the clock input using any type of source. However, other fundamentals can only be sinusoidal sources.

In addition to specifying the waveform parameters such as `type`, and `ampl`, you must also use the parameter `fundname` to specify a name for each non-DC source. Each fundamental can be a composition of several input sources with the same name. This is a difference between QPSS and other analyses.

Example F-1 shows how to set up the sources for the "Switched Capacitor Filter Example" on page 1240.

An eight-phase clock:

Example F-1 Setting Up the Sources for the Switched Capacitor Filter

```
//*****************BEGIN NETLIST**********************************
            .
            .
            .
// Clocks
Phi1  (phi1  gnd)  vsource type=pulse period=2.5us delay=0.25us\ width=1us val0=-
VDD val1=VDD rise=10ns fundname="Clock"
Phi2  (phi2  gnd)  vsource type=pulse period=2.5us delay=1.5us\ width=1us val0=-
VDD val1=VDD rise=10ns fundname="Clock"
Phi8  (phi8  gnd)  vsource type=pulse period=5.0us delay=1.5us\ width=2.25us val0=-
VDD val1=VDD rise=10ns fundname="Clock"
Phi9  (phi9  gnd)  vsource type=pulse period=5.0us delay=1.25us\ width=2.75us
val0=VDD val1=-VDD rise=10ns fundname="Clock"
Phi10 (phi10 gnd)  vsource type=pulse period=10.0us delay=3.75us\ width=5.25us
val0=VDD val1=-VDD rise=10ns fundname="Clock"
Phi11 (phi11 gnd)  vsource type=pulse period=10.0us delay=4.0us\ width=4.75us
val0=-VDD val1=VDD rise=10ns fundname="Clock"

// Input source
Vin   (pin   gnd)  vsource type=sine freq=100_Hz ampl=1 sinephase=0\
fundname="Input"
// QPSS Analysis
harmDisto QPSS funds=["Clock" "Input"] maxharms=[3 3] +swapfile="SomeFileName"
//*****************END NETLIST************************************
```

Example F-2 shows how to set up the sources and analysis for the "High-Performance Receiver Example" on page 1241.

Example F-2 Setting Up the Sources for the High Performance Receiver

```
//*****************BEGIN NETLIST**********************************
             .
             .
             .
// LO input
Vlo (8 7) vsource type=sine ampl=400mV freq=780MegHz fundname="LO"
//RF inputs
Prf1 (15 44) port  type=sine ampl=50mV freq=840MegHz fundname="F1"
+ ampl2=50mV freq2=840.01MegHz fundname2="F2"
//Analysis
InterModDisto QPSS funds=["LO" "F1" "F2"] maxharms=[5 3 3]
+ swapfile="SomeFileName"
//*****************END NETLIST**********************************
```

Sweeping a QPSS Analysis

You can combine a QPSS analysis with a Spectre circuit simulator Sweep analysis to create a powerful tool for a wide variety of applications, such as IP3 and IP5 calculations. For example, you might want to calculate the distortion for input power ranging from -60 dBm to 0 dBm. The netlist for this task is shown in Example F-3.

Example F-3 Calculating the Input Power Distortion from -60 to 0 dBm

```
//*****************BEGIN NETLIST*********************************
             .
             .
             .
parameters inputpower=-60
// Clock

Vlo (clock 0) vsource type=pulse val0=-1 val1=1 period=1n delay=0\ rise=10p
fall=10p width=460.00p fundname="carrier"
//  Input port
Prf (input 0)  port  type=sine r=50.0 num=1.0
+freq=+9.0E+08  dbm=inputpower fundname="RF1"
+freq2=+9.05E+08 dbm2=inputpower fundname2="RF2"
//  Analysis

swp sweep param=inputpower start=-60 stop=0 lin=6 {
distortion QPSS funds=["LO" "RF" "RF2"] maxharms=[5 3 3]
}
//*****************END NETLIST**********************************
```

Always arrange the sweep values so that analyses that converge more easily are performed first. When you sweep QPSS, it automatically uses the converged steady-state solution of the previous QPSS analysis. As discussed in “Convergence Aids” on page 1246, this practice can also aid convergence.

Convergence Aids

Normally QPSS analysis converges with default parameter settings, but occasionally you might need to adjust some parameter settings in order to achieve convergence.

Normally, giving a sufficiently large `tstab` parameter value or a looser `steadyratio` value resolves convergence problems during the initial QPSS stages. For convergence problems during the QPSS iterations, try the following procedures.

- Increase the tstab parameter value.

  The tstab parameter controls both the length of the initial transient integration, with only the clock tone activated, and the number of stabilizing iterations, with the moderate tones activated. The stable iterations are run before Newton iterations begin.

- Increase the `steadyratio` parameter value.

  The `steadyratio` parameter guards against false convergence. Its default values, `1.0` for `liberal`, `0.1` for `moderate`, and `0.01` for `conservative`, are derived from the `errpreset` parameter. Tighten `steadyratio` only if you suspect false convergence.

  Sometimes `steadyratio` must be loosened (for example to `steadyratio =1`) particularly with a tight `reltol` setting. Also loosen `steadyratio` when convergence stagnates. An indication of stagnation is that the `convNorm` value, which you can see on the screen, fluctuates within a certain range and never decreases further.

  The convergence tolerance of QPSS is determined by the product of `steadyratio` and `reltol`. Normally, you do not set `steadyratio` to a value higher than `10`.

- Severe trapezoidal rule ringing can prevent convergence.

  If you suspect trapezoidal rule ringing, use `method = gear2` or `method = gear2only`.

- Avoid using unnecessarily tight `reltol` settings.

  Excessively tight `reltol` significantly reduces efficiency besides causing convergence problems.

- Do a continuation on a parameter, such as input power, of moderate fundamentals.

  When the circuit is behaving in a highly nonlinear manner at a certain input power level, the PSS plus PAC approach might not compute a good enough estimate of the initial condition. One effective strategy is to ramp up the input power gradually by carefully arranging a sweep as described in “Sweeping a QPSS Analysis” on page 1245. Because it is usually much easier to achieve convergence at a low power input level where the circuit behaves in a more linear manner, start with a low input power. Once the simulation converges, save the steady-state solution as the initial condition for the next input power level. This process repeats automatically as QPSS is swept. You can achieve convergence at the desired input power level if the sweep steps are sufficiently small.

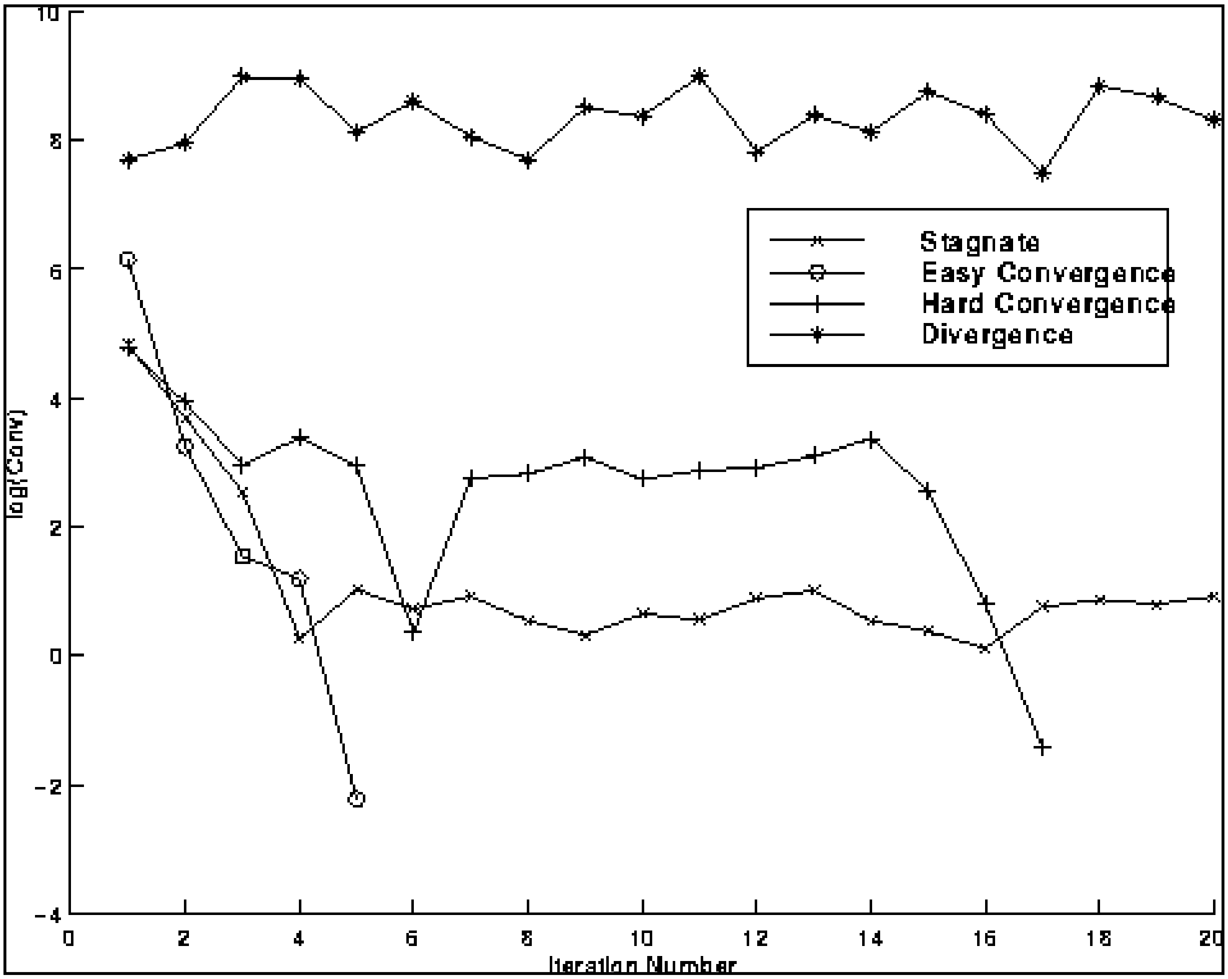
- In general, avoid using an excessively high number of harmonics for the moderate fundamentals.

  Using too many harmonics lengthens the simulation time and uses a lot of memory. However, if the number of harmonics you use is not high enough for a particular fundamental to adequately model its nonlinear effects, convergence problems might also occur.

An important indication of convergence or divergence is the `Conv` value printed to the screen. There are a few typical scenarios shown in Figure F-4 on page 1248.

- For most QPSS runs, the *Easy Convergence* scenario occurs.
- A few simulations follow the *Hard Convergence* scenario.
- If QPSS iterations *Stagnate* (the `Conv` value fluctuates close to but above `1.0`), loosen `steadyratio`. Loosening `steadyratio` solves the stagnation problem.
- If the iterations show *Divergence*, you usually must improve the initial condition. Do a continuation on input power level as described within the preceding list.

Figure F-4 QPSS Convergence Scenarios



Memory Management

QPSS is a memory-intensive analysis. If QPSS cannot be finished in core with real physical memory, use a swap file residing on a local disk. The simulator manages swapping much more efficiently than the operating system. The examples found in "Setting Up Sources" on page 1244 depict how to set up the `swapfile` parameter. Typically, 80% CPU utilization is achieved.

Dealing with Sub-harmonics

One advantage of QPSS analysis over PSS is that you need only provide a name for each fundamental frequency. The actual beat frequency value associated with the name is calculated automatically by the simulator. For each unique fundamental name, the simulator first finds all the frequencies associated with it. Then the greatest common factor is calculated among these frequencies, which is used as the beat frequency associated with the fundamental name.

However, if the fundamental frequency has sub-harmonics (circuit responses at some fraction of a driven frequency, typically 1/2 or 1/3), as with a divider, for example, the simulator currently cannot detect them. As a workaround, add a dummy source that tells the simulator of the existence of a sub-harmonic associated with a fundamental frequency.

Understanding the Narration from the QPSS Analysis

The examples in this section describe information that is typically printed to the screen during a QPSS analysis run.

After initialization, the SpectreRF circuit simulator confirms the fundamental tone names that were read and their beat frequencies. For this example, this circuit has

- A 1 GHz fundamental tone named `flo` as the large or clock signal.
- Two fundamental tones named `frf` and `fund2`, as moderate input signals. The frequency for `frf` is 900 MHz and the frequency for `fund2` is 920 MHz.

```
     .
     .
     .
Fundamental flo:  period = 1 ns, freq = 1 GHz.
Fundamental fund2:  period = 1.08696 ns, freq = 920 MHz.
Fundamental frf:  period = 1.11111 ns, freq = 900 MHz.
***************************************************************
Quasi-Periodic Steady State Analysis ‘qpss’: largefund = 1 GHz
*****************************************************************

=============================
‘qpss’: time = (1 ns -> 2 ns)
================================
     .
     .
     .
```

QPSS prints the following message and begins the initial transient iteration.

```
Starting qpss analysis iterations.
```

The initial transient iteration runs with only the clock tone (the large signal) active and all moderate input signals suppressed. Unlike PSS analysis (where each iteration performs a single transient integration), each QPSS iteration performs a number of transient integrations. In this example, 25 transient integrations are performed. The `tstab` parameter controls the length of the initial transient integration.

For the 1st transient integration, QPSS prints data as it steps through the integration. For the 2nd through 25th integrations, QPSS prints a countdown timer for each integration.

At the end of the 1st transient iteration, QPSS prints out a summary that includes `tstab`, the iteration number, its convergence norm, the node with the maximum deviation, and the amount of CPU time spent in the iteration.

```
     .
     .
     .
1st_Transient_Integration:
    qpss: time = 2.026 ns    (2.59 %), step = 1.792 ps     (179 m%)
    qpss: time = 2.077 ns    (7.66 %), step = 1.64 ps      (164 m%)
    qpss: time = 2.127 ns    (12.7 %), step = 2.572 ps     (257 m%)
    qpss: time = 2.176 ns    (17.6 %), step = 1.714 ps     (171 m%)
    qpss: time = 2.227 ns    (22.7 %), step = 2.492 ps     (249 m%)
    qpss: time = 2.277 ns    (27.7 %), step = 2.869 ps     (287 m%)
    qpss: time = 2.325 ns    (32.5 %), step = 3.514 ps     (351 m%)
    qpss: time = 2.375 ns    (37.5 %), step = 3.535 ps     (353 m%)
    qpss: time = 2.428 ns    (42.8 %), step = 3.566 ps     (357 m%)
    qpss: time = 2.477 ns    (47.7 %), step = 4.151 ps     (415 m%)
    qpss: time = 2.526 ns    (52.6 %), step = 4.464 ps     (446 m%)
    qpss: time = 2.576 ns    (57.6 %), step = 2.36 ps      (236 m%)
    qpss: time = 2.626 ns    (62.6 %), step = 2.378 ps     (238 m%)
    qpss: time = 2.675 ns    (67.5 %), step = 3.07 ps      (307 m%)
    qpss: time = 2.726 ns    (72.6 %), step = 4.936 ps     (494 m%)
    qpss: time = 2.777 ns    (77.7 %), step = 2.705 ps     (271 m%)
    qpss: time = 2.826 ns    (82.6 %), step = 3.1 ps       (310 m%)
    qpss: time = 2.875 ns    (87.5 %), step = 3.366 ps     (337 m%)
    qpss: time = 2.928 ns    (92.8 %), step = 3.661 ps     (366 m%)
    qpss: time = 2.976 ns    (97.6 %), step = 4.372 ps     (437 m%)

2nd_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
'tstab' iter = 1, convNorm = 64.3, maximum dI(rif:p) = 51.5933 uA, took 5.49 s.
     .
     .
     .
```

One or more stabilizing transient iterations run with all signals active; the clock tone (the large signal) and all moderate input signals. As for the first transient iteration, each QPSS iteration performs a number of transient integrations. This example performs 25 transient integrations. The `tstab` parameter controls the number of stabilizing iterations run when all tones are active.

At the end of each stabilizing transient iteration, QPSS prints out a summary that includes `tstab`, the iteration number, its convergence norm, the node with the maximum deviation, and the amount of CPU time spent in the iteration.

```
     .
     .
     .
1st_Transient_Integration:
    qpss: time = 2.026 ns    (2.59 %), step = 1.792 ps     (179 m%)
    qpss: time = 2.077 ns    (7.66 %), step = 1.64 ps      (164 m%)
    qpss: time = 2.127 ns    (12.7 %), step = 2.572 ps     (257 m%)
    qpss: time = 2.176 ns    (17.6 %), step = 1.714 ps     (171 m%)
    qpss: time = 2.227 ns    (22.7 %), step = 2.492 ps     (249 m%)
    qpss: time = 2.277 ns    (27.7 %), step = 2.869 ps     (287 m%)
    qpss: time = 2.325 ns    (32.5 %), step = 3.514 ps     (351 m%)
    qpss: time = 2.375 ns    (37.5 %), step = 3.535 ps     (353 m%)
    qpss: time = 2.428 ns    (42.8 %), step = 3.566 ps     (357 m%)
    qpss: time = 2.477 ns    (47.7 %), step = 4.151 ps     (415 m%)
    qpss: time = 2.526 ns    (52.6 %), step = 4.464 ps     (446 m%)
    qpss: time = 2.576 ns    (57.6 %), step = 2.36 ps      (236 m%)
    qpss: time = 2.626 ns    (62.6 %), step = 2.378 ps     (238 m%)
    qpss: time = 2.675 ns    (67.5 %), step = 3.07 ps      (307 m%)
    qpss: time = 2.726 ns    (72.6 %), step = 4.936 ps     (494 m%)
    qpss: time = 2.777 ns    (77.7 %), step = 2.705 ps     (271 m%)
    qpss: time = 2.826 ns    (82.6 %), step = 3.1 ps       (310 m%)
    qpss: time = 2.875 ns    (87.5 %), step = 3.366 ps     (337 m%)
    qpss: time = 2.928 ns    (92.8 %), step = 3.661 ps     (366 m%)
    qpss: time = 2.976 ns    (97.6 %), step = 4.372 ps     (437 m%)

2nd_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
'tstab' iter = 2, convNorm = 15.2, maximum dI(rif:p) = -43.8125 uA, took 5.41 s.
     .
     .
     .
```

The Newton iterations run after the stabilizing iterations.

The QPSS analysis employs the Mixed Frequency Time (MFT) algorithm extended to multiple fundamental frequencies. The large tone is resolved in the time domain and the moderate tones are resolved in the frequency domain (hence the name mixed frequency time algorithm). The QPSS analysis uses the shooting Newton method as its backbone. However, unlike PSS analysis where each Newton iteration performs a single transient integration, for each Newton iteration the QPSS analysis performs a number of transient integrations.

When you set up a QPSS analysis, you determine the number of integrations performed by the number of moderate fundamental harmonics you select. The efficiency of the shooting Newton algorithm depends significantly on the number of harmonics required to model the responses of moderate fundamentals. The number of harmonics of the large fundamental does not significantly affect the efficiency of the Newton algorithm. The boundary conditions of a shooting Newton interval are such that the time domain integrations are consistent with a frequency domain transformation with a shift of one large signal period.

The shooting Newton iterations run with all signals active. As for the stabilizing transient iterations, each shooting Newton iteration performs a number of transient integrations. This example performs 25 transient integrations.

At the end of each shooting Newton iteration, QPSS prints out a summary that includes `Newton iter`, the iteration number, its convergence norm, the node with the maximum deviation, and the amount of CPU time spent in the iteration.

```
     .
     .
     .
1st_Transient_Integration:
    qpss: time = 2.026 ns    (2.59 %), step = 1.792 ps     (179 m%)
    qpss: time = 2.077 ns    (7.66 %), step = 1.64 ps      (164 m%)
    qpss: time = 2.127 ns    (12.7 %), step = 2.572 ps     (257 m%)
    qpss: time = 2.176 ns    (17.6 %), step = 1.714 ps     (171 m%)
    qpss: time = 2.227 ns    (22.7 %), step = 2.492 ps     (249 m%)
    qpss: time = 2.277 ns    (27.7 %), step = 2.869 ps     (287 m%)
    qpss: time = 2.325 ns    (32.5 %), step = 3.514 ps     (351 m%)
    qpss: time = 2.375 ns    (37.5 %), step = 3.535 ps     (353 m%)
    qpss: time = 2.428 ns    (42.8 %), step = 3.566 ps     (357 m%)
    qpss: time = 2.477 ns    (47.7 %), step = 4.151 ps     (415 m%)
    qpss: time = 2.526 ns    (52.6 %), step = 4.464 ps     (446 m%)
    qpss: time = 2.576 ns    (57.6 %), step = 2.36 ps      (236 m%)
    qpss: time = 2.626 ns    (62.6 %), step = 2.378 ps     (238 m%)
    qpss: time = 2.675 ns    (67.5 %), step = 3.07 ps      (307 m%)
    qpss: time = 2.726 ns    (72.6 %), step = 4.936 ps     (494 m%)
    qpss: time = 2.777 ns    (77.7 %), step = 2.705 ps     (271 m%)
    qpss: time = 2.826 ns    (82.6 %), step = 3.1 ps       (310 m%)
    qpss: time = 2.875 ns    (87.5 %), step = 3.366 ps     (337 m%)
    qpss: time = 2.928 ns    (92.8 %), step = 3.661 ps     (366 m%)
    qpss: time = 2.976 ns    (97.6 %), step = 4.372 ps     (437 m%)
2nd_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
     .
     .
     .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 1, convNorm = 28.3, maximum dI(rif:p) = 9.16928 uA, took 7.58 s.
```

```
        .
        .
        .
```

In this example, four Newton iterations were performed to reach the steady-state solution. Information about the QPSS analysis including the steady-state solution print at the end.

```
        .
        .
        .
1st_Transient_Integration:
        .
        .
        .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 2, convNorm = 1.35, maximum dI(L1:1) = 927.427 nA, took 8.29 s.
1st_Transient_Integration:
        .
        .
        .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 3, convNorm = 4.21e-03, maximum dI(q56:i_extra) = -20.204 nA,
took 8.24 s.
1st_Transient_Integration:
        .
        .
        .
25th_Transient_Integration:
......9......8......7......6......5......4......3......2......1......0
Newton iter  = 4, convNorm = 1.05e-06, maximum dI(q56:i_excess) = -478.108 fA,
took 5.78 s.
qpss: The steady-state solution was achieved in 6 iterations.
Number of accepted qpss steps = 360 each in 25 time intervals.
Starting spectrum calculation.
Total time required for qpss analysis ‘qpss’ was 42.83 s.
        .
        .
        .
```

Occasionally, you might see warning messages such as the following

```
Minimum time step used.  Solution might be in error.
```

or

```
Junction current exceeds ‘imelt’.  The results computed by Spectre are now incorrect
because the junction current model has been linearized.
```

You can ignore these warning messages if they appear in the early stage of QPSS iterations. They might be caused by bad starting integration conditions and do not affect the final solution. However, if they appear in the final iteration, the solution might be in error.

References

[1] A. Allgower and K. Georg, *Numerical Continuation Methods*, Springer-Verlag, New York, 1990.

[2] L. O. Chua and A. Ushida, “Algorithms for computing almost periodic steady-state response of nonlinear systems to multiple input frequencies,” *IEEE Transactions on Circuits and Systems*, vol. 28, pp. 953-971, 1981.

[3] D. Feng, J. Phillips, K. Nabors, K. Kundert, and J. White, “Efficient computation of quasi-periodic circuit operating conditions via a mixed frequency/time approach,” Submitted to *Proceedings of the 36th Design Automation Conference*, June 1999.

[4] K. Kundert, J. White, and A. SangiovanniI-Vincentelli, “A mixed frequency-time approach for distortion analysis of switching filter circuits,” *IEEE Journal of Solid State Circuits*, vol 24, pp. 443-451, 1989.

[5] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, second ed., 1985.

[6] Y. Saad, *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.

[7] R. Telichevesky, J. White, and K. Kundert, “Efficient steady-state analysis based on matrix-free krylov-subspace methods,” in *Proceedings of 32rd Design Automation Conference*, June 1995.

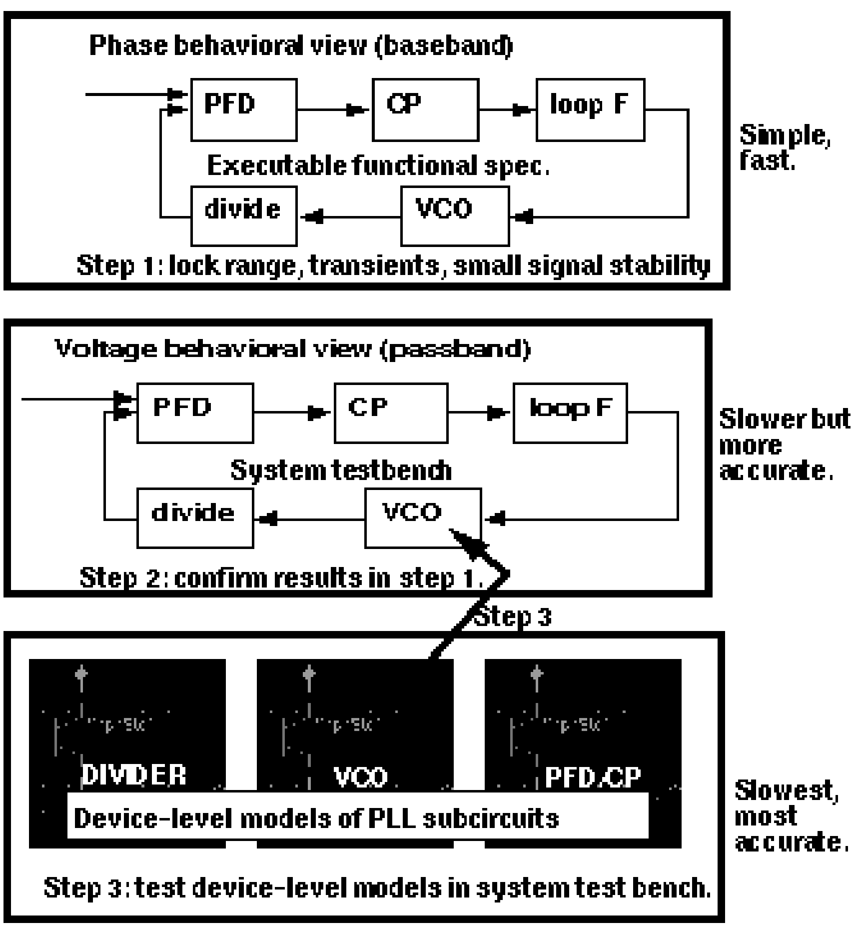
[8] _____, “Efficient AC and noise analysis of two-tone RF circuits,” in *Proceedings of the 1996 Design Automation Conference*, June 1996.

G

Introduction to the PLL library

The models in the phase lock loop (PLL) library support top-down design of PLLs. Figure G-1 shows the three steps of the design flow. This appendix describes the first step in detail; all three steps are described briefly.

Figure G-1 PLL Top-Down Design Flow



1. The first step in Figure G-1 is to develop an executable specification. The executable specification is an arrangement of fast behavioral models that permits fast architectural studies to separate specification and implementation issues. The executable specification contains baseband models [1,2,3,4,5,6,7,8,9]. (Reference [1] uses the terms “baseband” and “bandpass” explicitly.)

   These baseband models suppress clocks and RF/IF carriers. Some literature refers to PLL baseband models as “relative phase” or “phase-domain” models [2]. This appendix uses the latter term. Phase-domain PLL models are exceptionally fast, capture the important non-linear mechanisms, and can be linearized directly for AC analysis.

2. The second step in Figure G-1 is to translate the executable specification into a system testbench. The system testbench, unlike the executable specification, is composed of passband models [1]. This Appendix refers to passband models as voltage-domain models because they simulate voltages you can observe in a laboratory.

   Comparing voltage- and phase-domain voltage-controlled oscillator (VCO) models highlights the difference between the two models. The output of a voltage-domain VCO model is a clock voltage, a periodic signal. The output of a phase-domain VCO model is a voltage numerically equal to phase. If you unwrap the VCO phase, in steady state, it ramps up indefinitely. Unwrapped phase is not periodic. Voltage-domain models describe non-linear effects related to the shapes of the actual RF waveforms. Such waveform effects include spurs and harmonic locking. Harmonic locking occurs when the PLL locks on to a harmonic of the reference.

   Phase-domain models do not simulate waveform effects. The system testbench is more accurate than the executable specification, but it is still behavioral. Equipped only with behavioral voltage-domain models, the testbench does not simulate device-level effects associated with specific implementations. Examples of such implementation effects are interstage loading, improper bias, and device parasitics.

3. The last step in Figure G-1 is to gradually replace the behavioral models in the system testbench with device-level models, one or two blocks at a time. Device-level models check for the previously mentioned implementation problems. The entire PLL is simulated at the device-level only as a final verification step because such simulations are very lengthy.

Models in the PLL library

The PLL library includes the following phase-domain models:

- Analog multiplier phase detector
- XOR phase detector with bipolar output

The XOR phase detector is not explicitly discussed here because it is very similar to the analog multiplier phase detector. The only difference is that the duty cycle-phase error transfer curve is triangular instead of sinusoidal.

- Three-state digital phase frequency detector (PFD)
- Charge pump (current source version)
- VCO tuning curve (analytic and tabular versions)
- Frequency divider
- Lock indicator

Introduction

The primary system-level specifications captured by this first set of phase-domain models are acquisition time, lock and capture ranges [12], and phase margin. The PFD model also simulates backlash[8]. Backlash is sometimes called “deadband” effect. It is a limit cycle caused by the phase-frequency detector’s inability to linearly reduce its output pulse width to zero as phase error goes to zero.

The remainder of this appendix is divided into two main sections.

- The first section introduces phase-domain modeling, describes a feature included to prevent DC convergence problems, and then shows you some examples of using phase-domain models.
- The second section explains how to assemble a more complex PLL and discusses an example. The examples are introductory and are not a comprehensive discussion of all applications of phase-domain models.

Phase-Domain Model of a Simple PLL

Description

This PLL example, which is built around the simplest phase detector in the library, introduces the fundamentals of phase-domain modeling. Figure G-2 on page 1258 shows a voltage-domain model of the example and also some selected waveforms. The phase detector in this case is an ideal analog multiplier.

Figure G-2 Voltage Domain Model

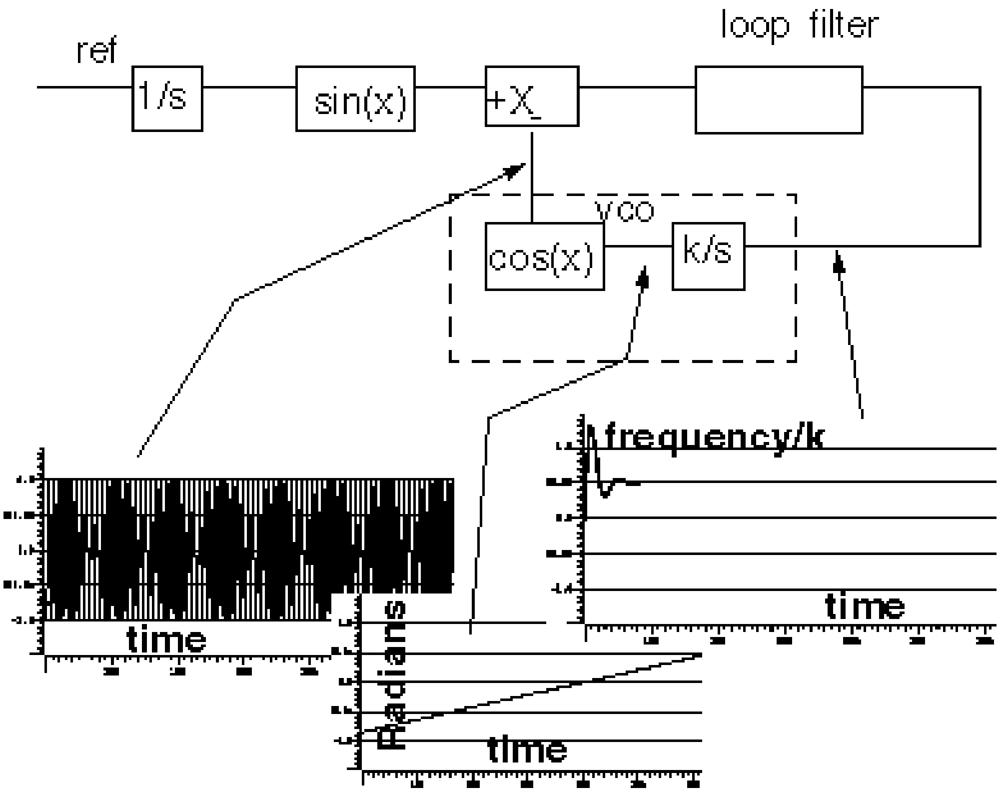


Figure G-3 on page 1259 shows the equivalent phase-domain model. The phase-domain model is based on the following trigonometric identity:

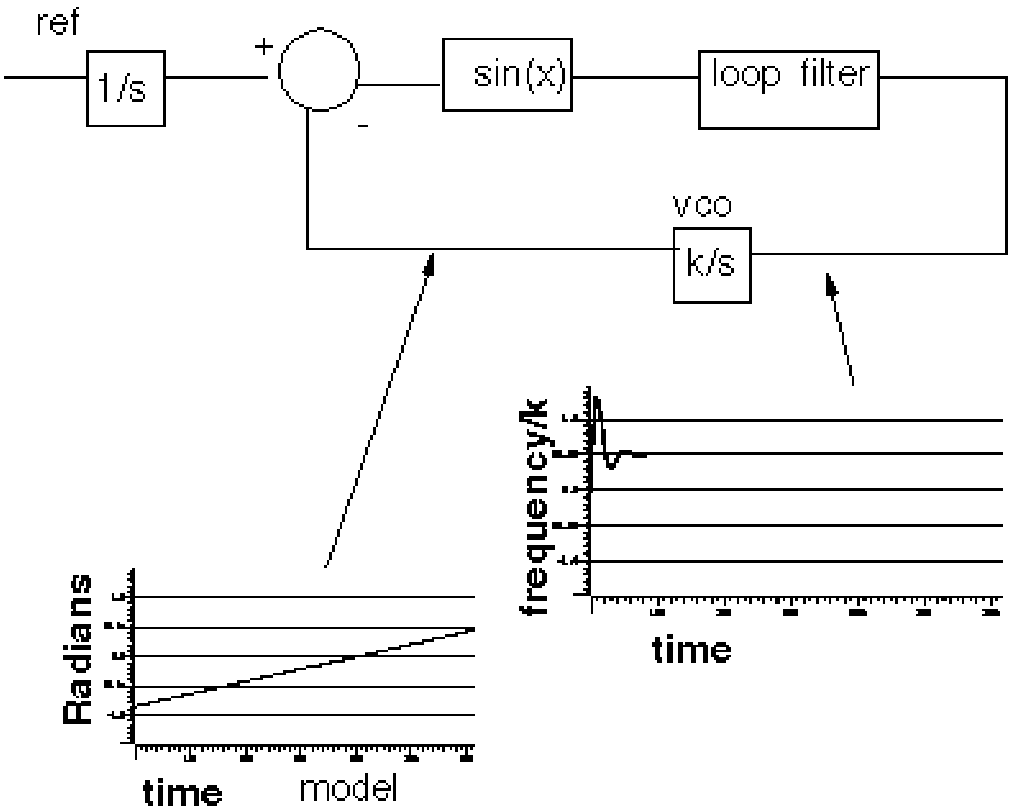
$\sin(\theta_1)*\cos(\theta_2) = (1/2)*[\sin(\theta_1+\theta_2)+\sin(\theta_1-\theta_2)]$

which, after filtering is approximately

$(1/2)*\sin(\theta_1-\theta_2)$

Note: $q_1+q_2=(w_1+w_2)*t$ and w_1+w_2 usually lie far beyond the filter's corner frequency.

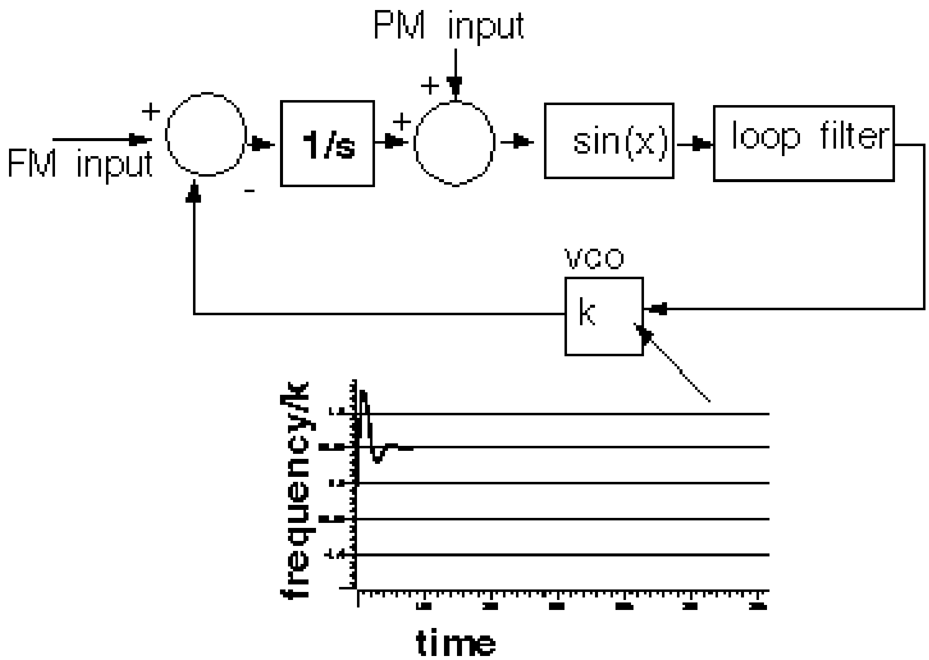
Figure G-3 Phase Domain Model



The phase and frequency waveforms from the phase-domain model match their voltage-domain counterparts, but the simulation runs faster because the oscillatory waveform is not explicitly simulated.

Combining the integrators, as shown in Figure G-4 on page 1260, eliminates the integrator outside the feedback loop which might cause a problem if you forget to specify an initial condition. Combining the integrators is also necessary if you build phase-domain models of phase-frequency detectors because, in this case, the non-linearity has memory (hysteresis).

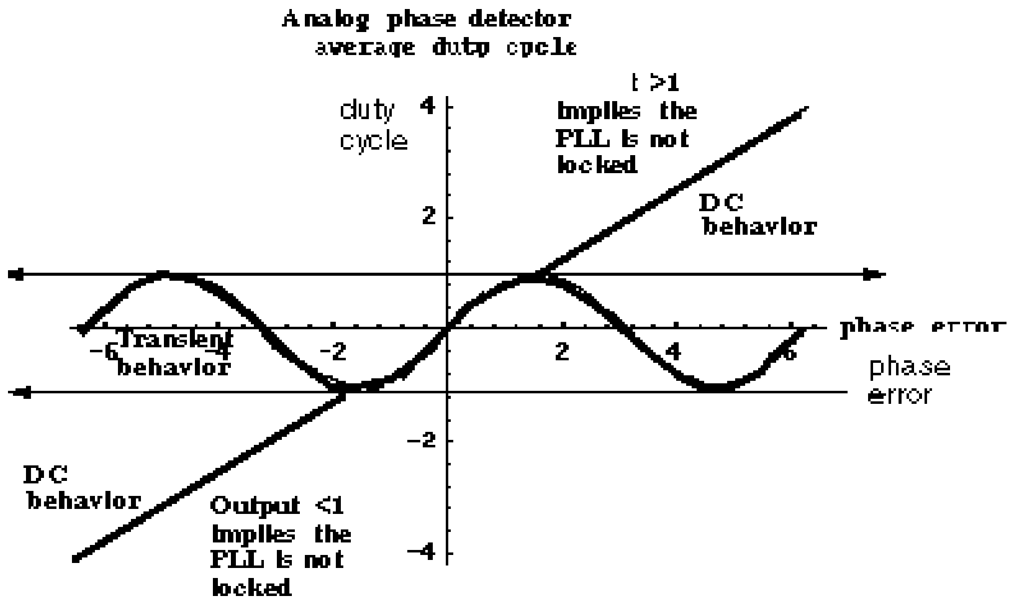
Figure G-4 A More Practical Phase Domain Model



The models in both Figure G-3 and Figure G-4 can fail to achieve DC convergence because the phase detector model either has no DC operating point or because it has an infinite number of operating points.

The sinusoidal function in Figure G-5 on page 1261 is the phase detector transfer curve. The phase detector is the only non-linear element in this PLL model. For reasons associated with phase-frequency detectors, the phase detector output is called the *duty cycle*.

Figure G-5 Phase Detector Transfer Curves



If the required duty cycle lies outside [-1,1], the loop is not locked in steady state. If the required duty cycle lies within [-1,1], there are an infinite number of possible phase errors. In either case, a Spectre simulation might not converge. The ability of Verilog-A© to perform different tasks for different analyses provides an elegant solution to the DC convergence problems and a quick way to map out lock range. Lock range is the range of input frequencies for which the PLL can maintain lock. (Some literature refers to lock range as hold-in range [8].)

The phase detector model uses the monotonic transfer curve for DC analysis and the true periodic transfer curve for transient analysis. The two transfer curves coincide when the phase error lies in the interval [–π/2, π/2]. If the required duty cycle lies within [-1,1], the monotonic transfer curve forces the steady-state phase error to the interval [-π/2, π/2], where the two curves coincide. The equilibrium point is *open-loop-stable*, meaning that at DC the loop gain is a positive real number. This is true because the slope of the transfer curve is positive over [-π/2, π/2]. The Nyquist stability criterion is therefore easier to apply. The DC analysis is general enough because only the phase error modulo 2π is of interest, and you usually care only about the open-loop-stable operating points. When the loop is not locked, the DC analysis computes a duty cycle with a magnitude greater than one. A duty cycle greater than one is clearly incorrect, but it is much easier to interpret than a convergence error. DC duty cycle is a lock indicator which can be used in a parametric DC analysis to sweep out lock range.

Example 1: Dynamic Test for Capture Range and Lock Range

The circuit used to dynamically test for capture range and lock is range is *example_analog_PD* in the *pllLib* library. *Capture range* is the range of input frequencies that the PLL can acquire from an unlocked state. Since acquisition of frequencies near the edge of the capture range involves a pull-in mechanism [1-12], measuring the capture range requires a transient analysis. You can measure capture and lock ranges by slowly sweeping the input frequency and observing the frequency at which the duty cycle begins and ends a long ramp [12]. You must skip the DC analysis to observe the capture limits. Figure G-6 on page 1262 plots the VCO control voltage against the input frequency voltage. The input frequency first ramps up and then down. A buffered auxiliary circuit responds to the duty cycle and adds 2.5 volts when the input frequency changes direction. This technique makes the plot easier to read because the forward and reverse sweeps occupy different parts of the vertical scale. In this example, lock range is from 1.36Khz to 3.4Khz, and the capture range is from 1.8Khz to 3Khz.

Figure G-6 Lock Range and Capture Range

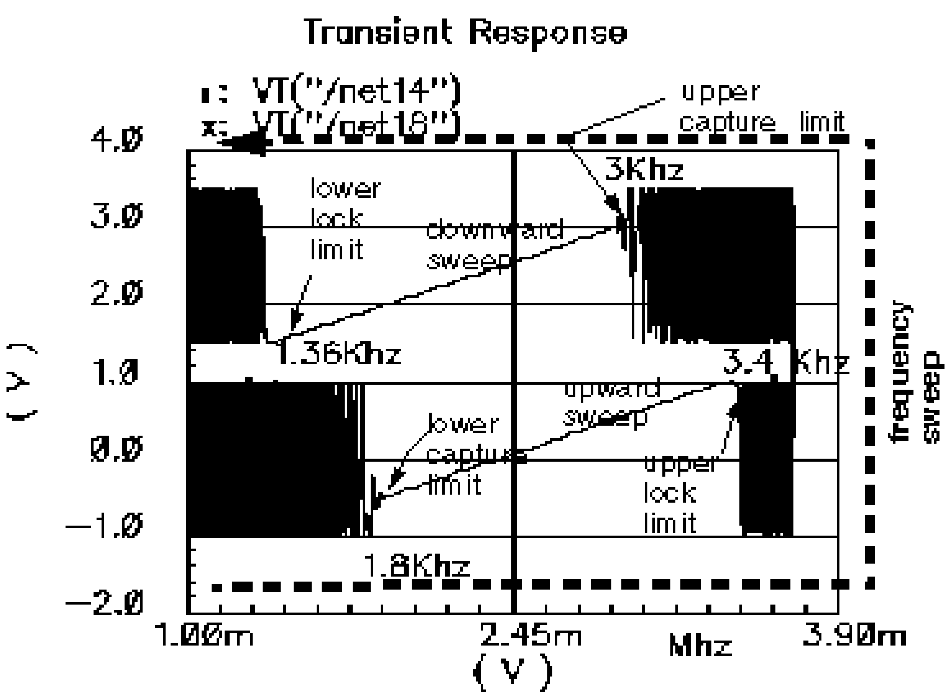
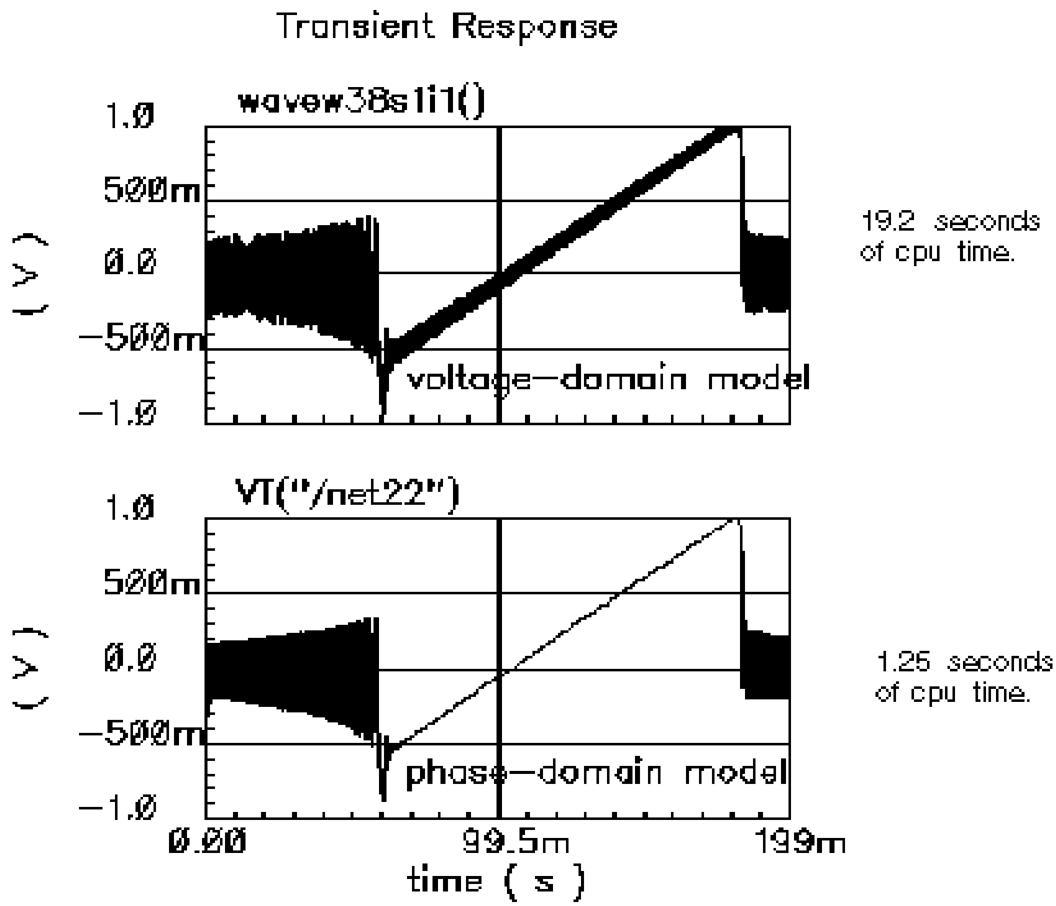


Figure G-7 on page 1263 compares VCO control voltages in the forward sweep when computed with voltage- or phase-domain models. The models produce similar results. In this example (2.5Khz center frequency), the phase-domain model is only about 20 times faster than the voltage-domain model.

Figure G-7 VCO Control Voltages Computed with Voltage- and Phase-Domain Models

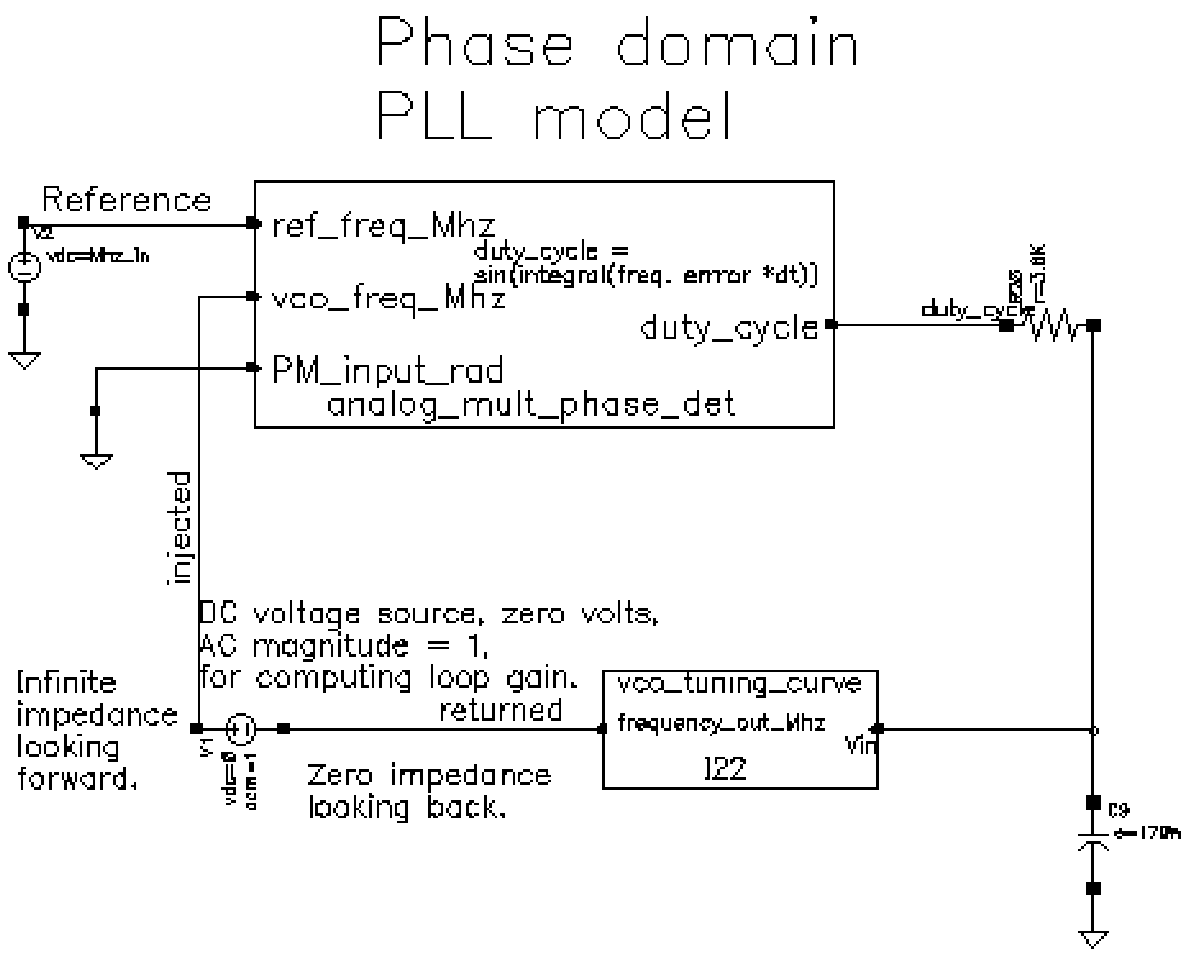


Example 2: Loop Gain Measurement

Spectre cannot perform a useful AC analysis on a voltage-domain model because by design, a voltage-domain PLL model has no DC operating point. However, because Spectre linearizes phase-domain models about phase error, and phase error is a meaningful DC quantity, subsequent AC analyses are valid.

This example describes how to compute loop gain with a phase-domain model. Figure G-8 on page 1264 shows an analog design environment version of the model shown in Figure G-4 on page 1260. The circuit used to measure loop gain is *example_loop_gain* in the *pllLib* library.

Figure G-8 Set Up for Loop Gain Measurement



The phase-domain model in Figure G-8, *example_loop_gain*, includes a voltage source inserted after the VCO. The DC voltage is zero volts, and the AC magnitude is 1 volt. The new voltage source inserts a test signal without changing the DC operating point. You must insert this source at a point where the impedance looking back is much smaller than the impedance looking forward. The accuracy of the resulting loop gain computation depends on how well this condition is met.

Use the following procedure to compute the loop gain.

Open the example_loop_gain Schematic

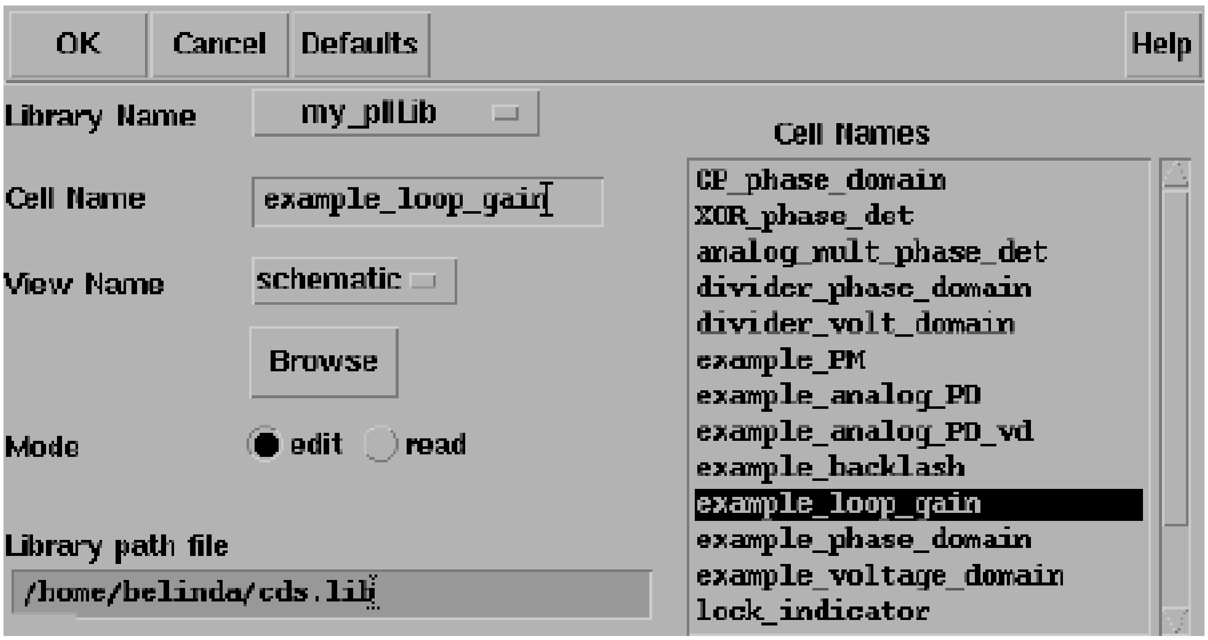
1. In the CIW, choose *File – Open*.

   The Open File form appears.

2. In the Open File form, choose *my_pllLib* in the *Library Name* cyclic field. Choose the editable copy of the pllLib library you created. (You can create an editable copy of the *pllLib* in the same way as is described for the *rfExamples* library in Chapter 3, “Setting Up for the Examples.”)

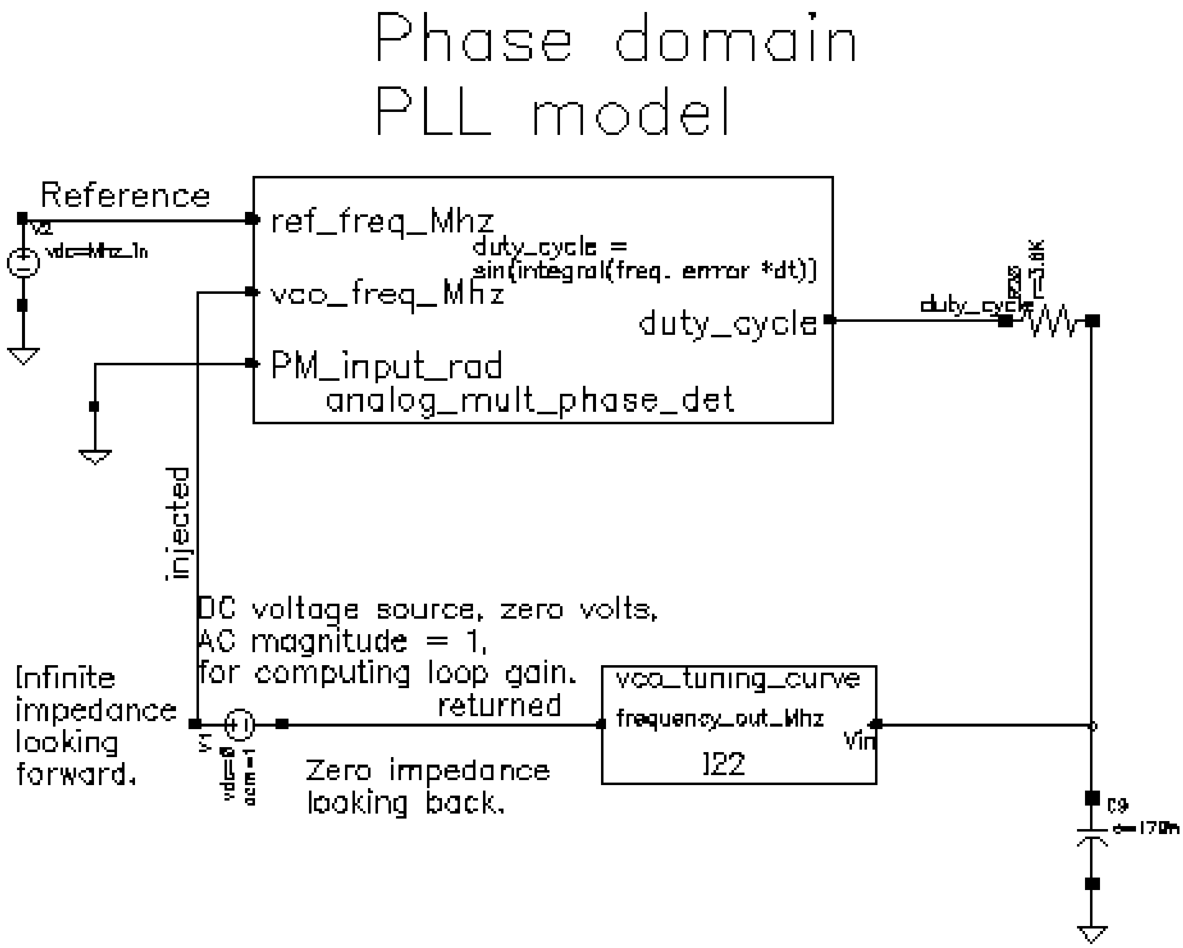
3. Choose *example_loop_gain* in the *Cell Names* list box.

   The completed Open File form appears like the one below.



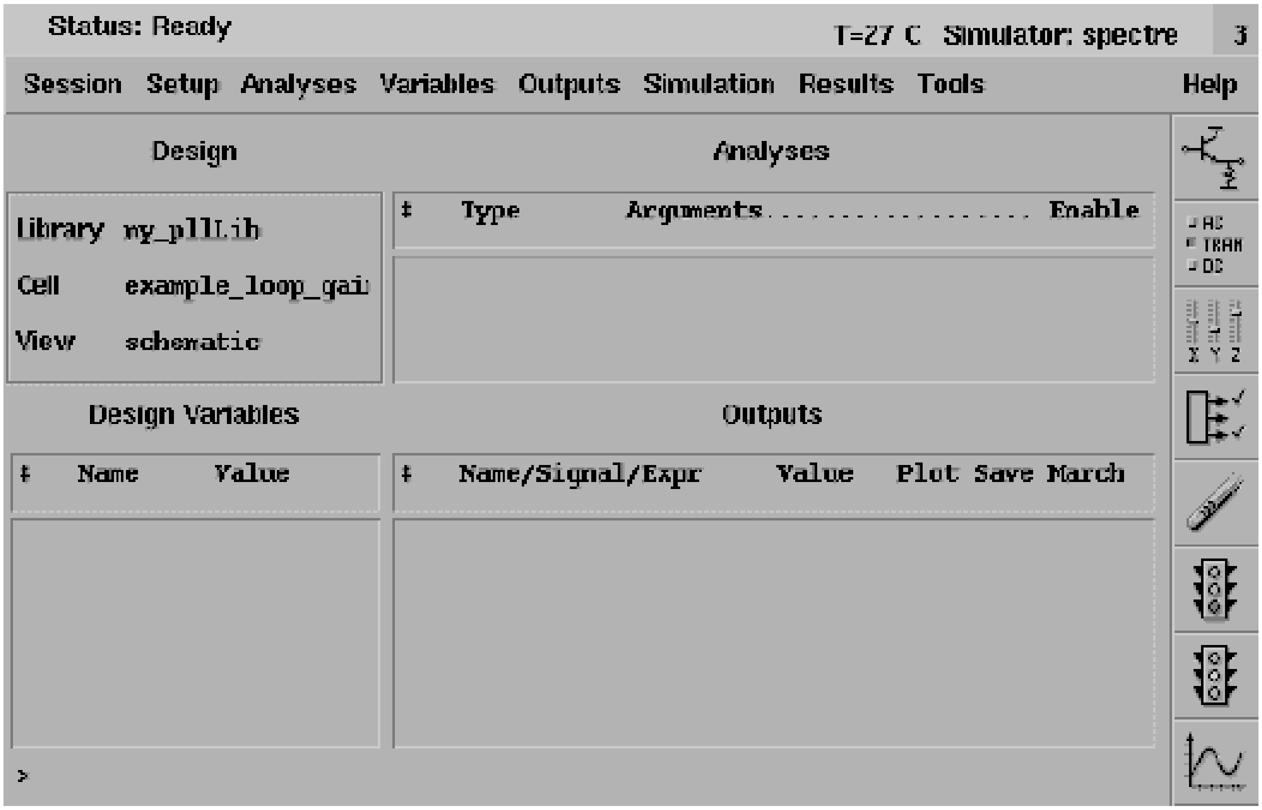
4. Click *OK*.

The Schematic window for the *example_loop_gain* circuit appears.



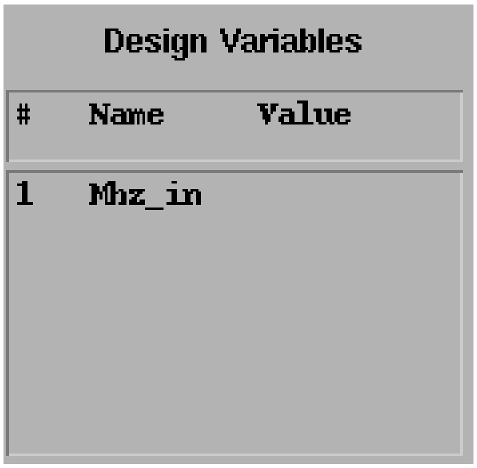
5. In the Schematic window, choose *Tools– Analog Environment*.

The Simulation window opens. This window is also called the Cadence® Analog Circuit Design Environment.



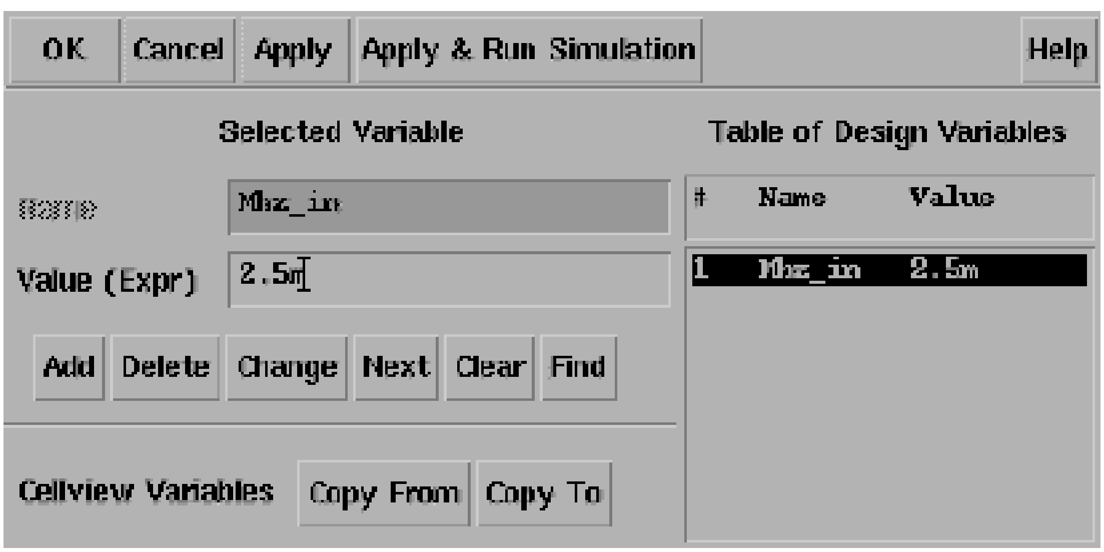
Setting Up the Design Variables

1. In the Simulation window, choose *Variables—Copy From Cellview* to copy variables from the schematic to the Simulation window. *Mhz_in* displays in the *Design Variables* area of the Simulation window.



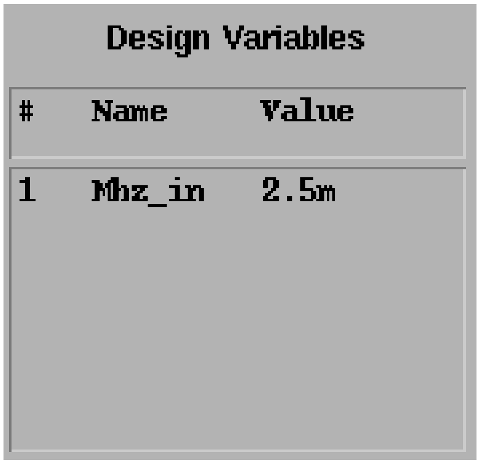
2. In the Simulation window, choose *Variables—Edit*, to provide a value for the *Mhz_in* variable.

   The Editing Design Variables form appears.



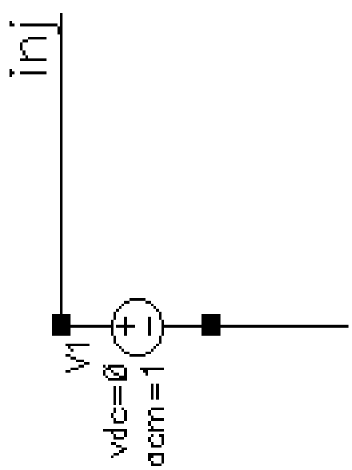
3. In the *Value (Expr)* field, type `2.5m` for the value of `Mhz_in` and click *Change*.
4. In the Editing Design Variables form, click *OK*.

5. The new value for *Mhz_in* displays in the *Design Variables* area of the Simulation window.



Setting Up the AC and DC Simulations

Set up both AC and DC analyses. The zero-voltage *vdc* source must be the only source with a non-zero AC magnitude.



When you set up the DC analysis, save the DC data so you can annotate the schematic with DC node voltages.

1. In the Simulation window, choose *Analysis - Disable* to disable any analyses you ran previously. (Check the Simulation window to verify whether or not any analysis is enabled.)

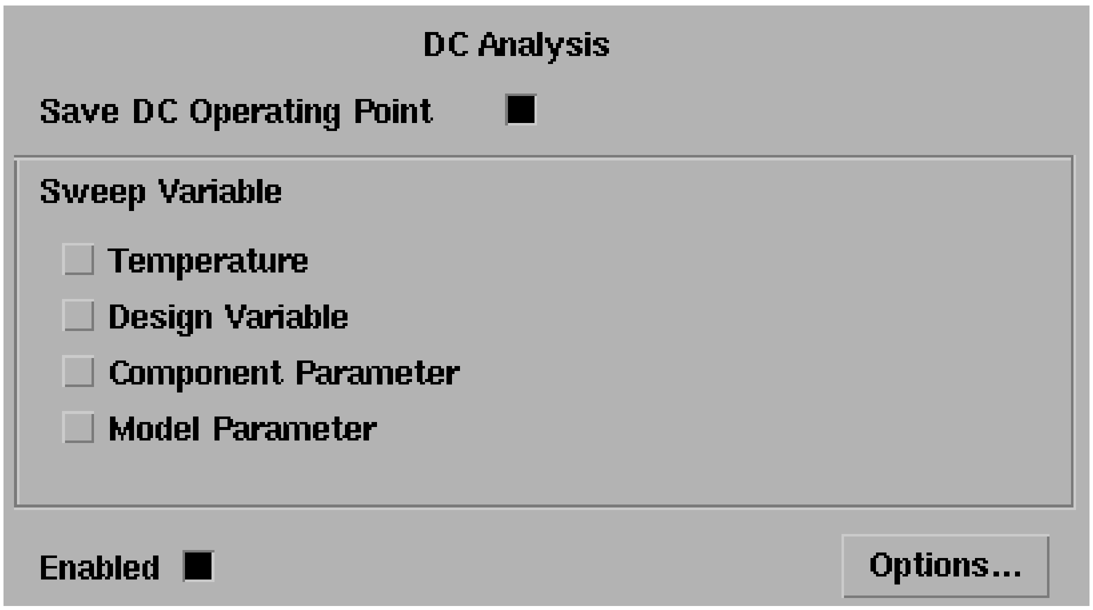
2. In the Simulation window, choose *Analysis - Choose* to display the Choosing Analyses form.

3. Click on *dc* to set up the DC analysis.

   In the DC Analysis area

   a. Highlight *Save DC Operating Point*.

   b. Highlight *Enabled*.



4. Click on *ac* to set up the AC analysis.

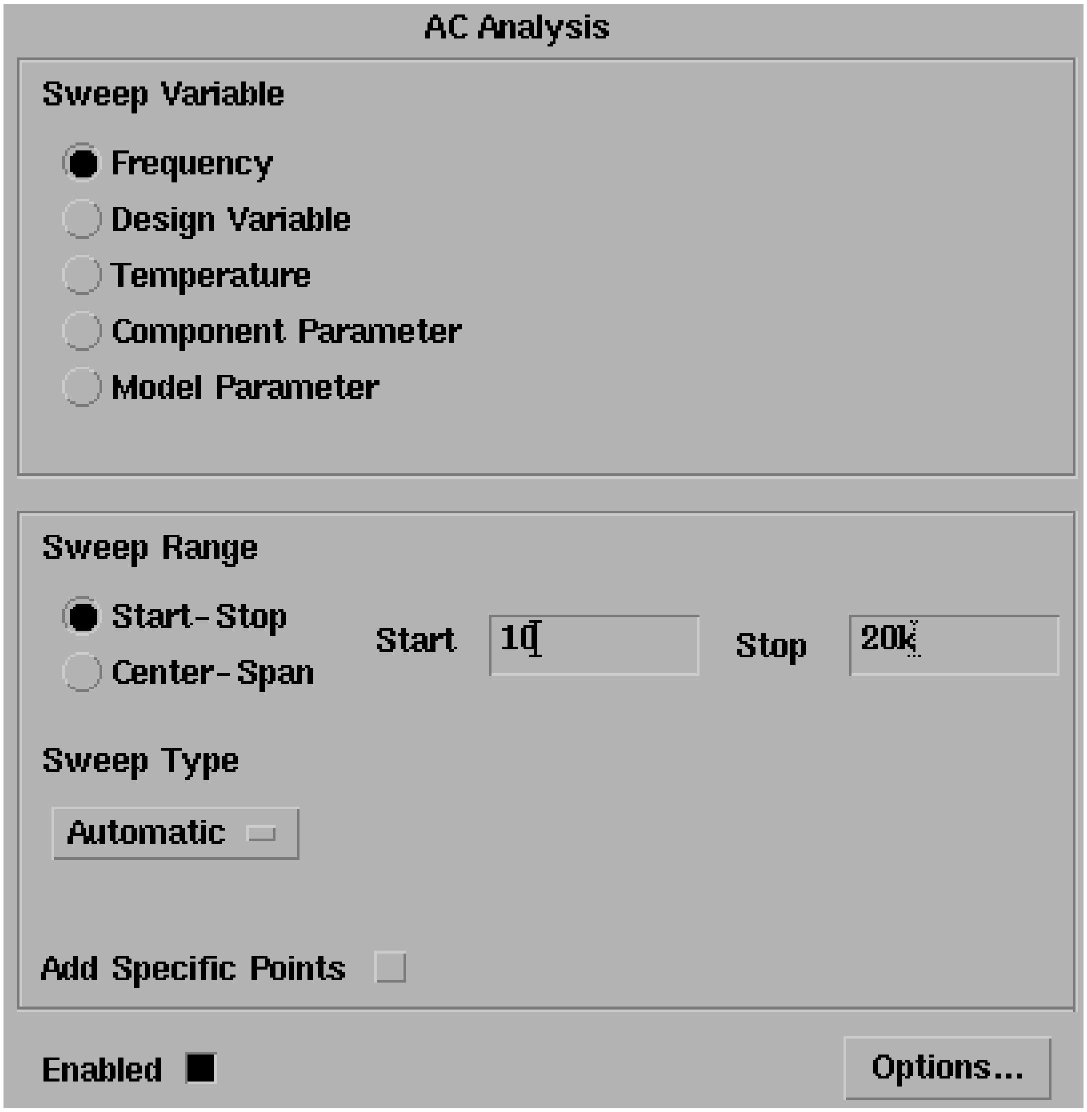
   In the AC Analysis area

   a. Highlight *Frequency* for *Sweep Variable.*

   b. Highlight *Start - Stop* for *Sweep Range*. Type `10` in the *Start* field and `20k` in the *Stop* field.

   c. Select *Automatic* in the *Sweep Type* cyclic field.

d. Highlight *Enabled*.



5. Click *OK* in the Choosing Analyses form.

Both analyses are displayed in the Simulation window.

Analyses

#	Type	Arguments..................				Enable
1	dc	t				yes
2	ac	10	20K	Auto..	Star..	yes

Run the Analyses

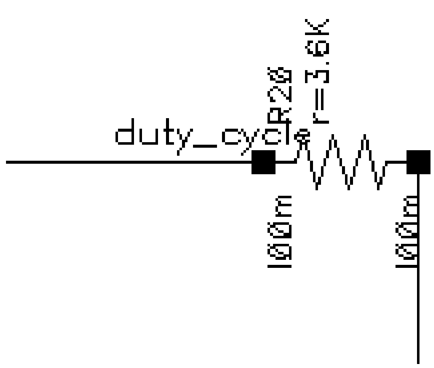
1. To run the analyses, choose *Simulation – Netlist and Run* in the Simulation window.

   The output log file appears and displays information about the simulation as it runs.

2. Look in the CIW for a message that says the simulation completed successfully.

Displaying the DC Voltages on the Schematic Nodes

➤ In the Simulation window, choose *Results – Annotate – DC Node Voltages* to display the node voltages on the schematic. The DC operating point for the net called *duty_cycle* must remain between -1 volt and 1 volt.



If the operating point falls outside the interval [-1, 1] volt, the loop is not locked and the AC analysis is invalid.

AC Response as Gain and Phase

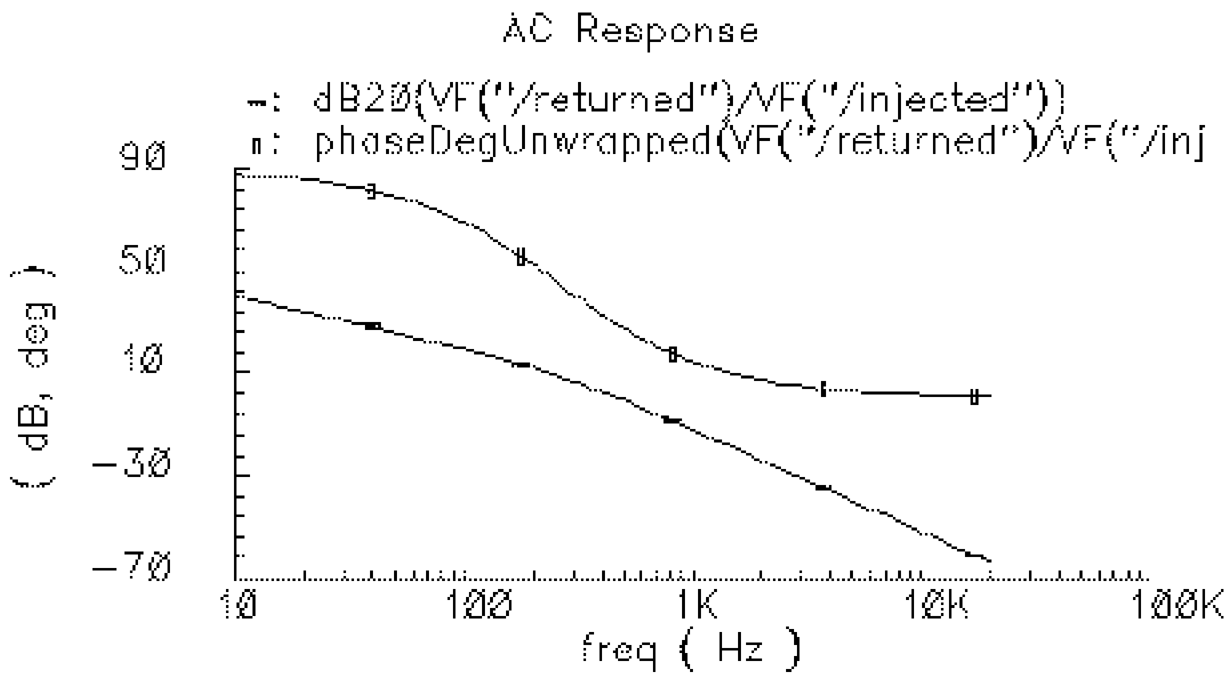
1. In the Simulation window, choose *Results – Direct Plot – AC Gain & Phase* and follow the prompts at the bottom of the Schematic window.

   a. *Select first point*—Select the net labeled *returned* in the schematic.

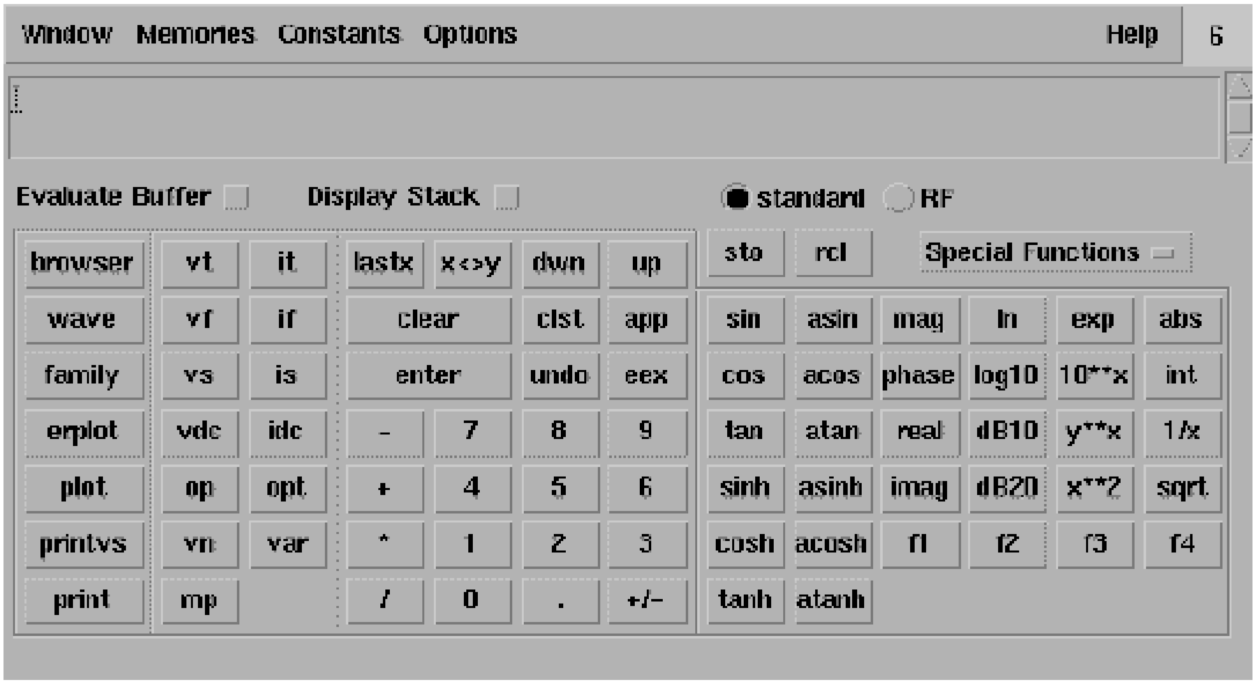
   b. *Select second point*—Select the net labeled *injected* in the schematic.

   The Waveform window displays two curves.

   - The top curve plots phase.
   - The bottom curve plots gain.



2. In the Waveform window, choose *Tools—Calculator* to open the Waveform Calculator.

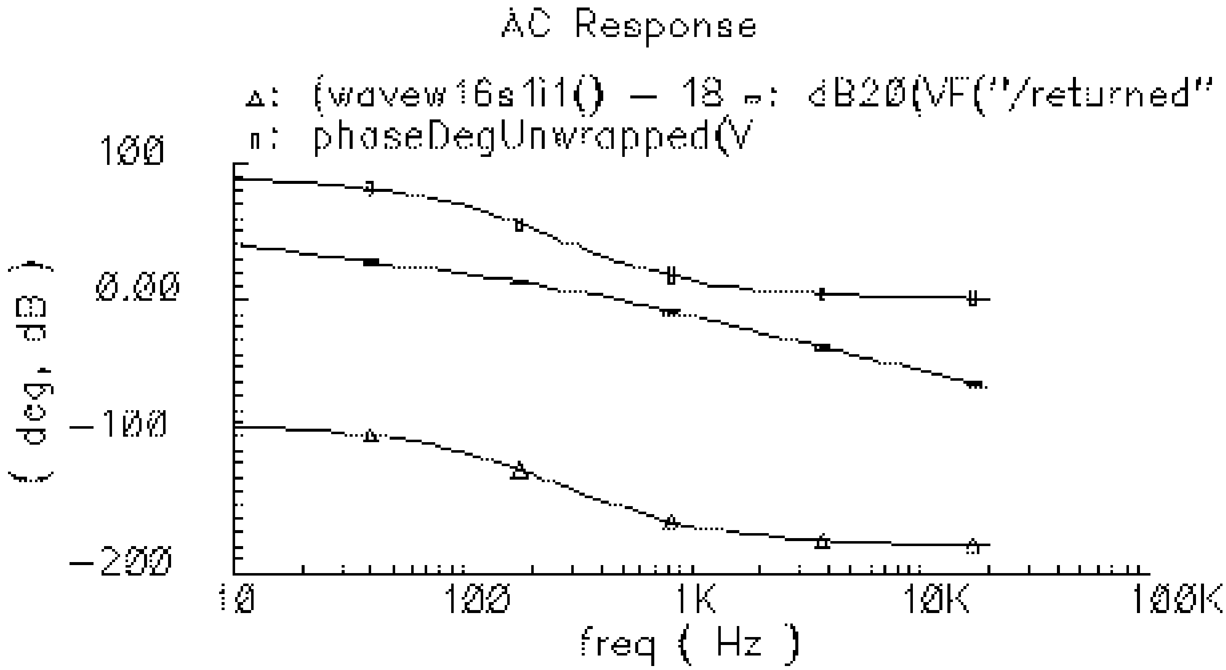


3. In the Calculator, click the *wave* button (on the left).
4. Then, in the Waveform window, select the phase curve (on the top).

5. In the Calculator, to perform the calculations algebraically, choose *Options—Set Algebraic*.
6. Subtract 180 from the phase waveform—In the Calculator, click the subtraction symbol (-) followed by the numbers *180*.

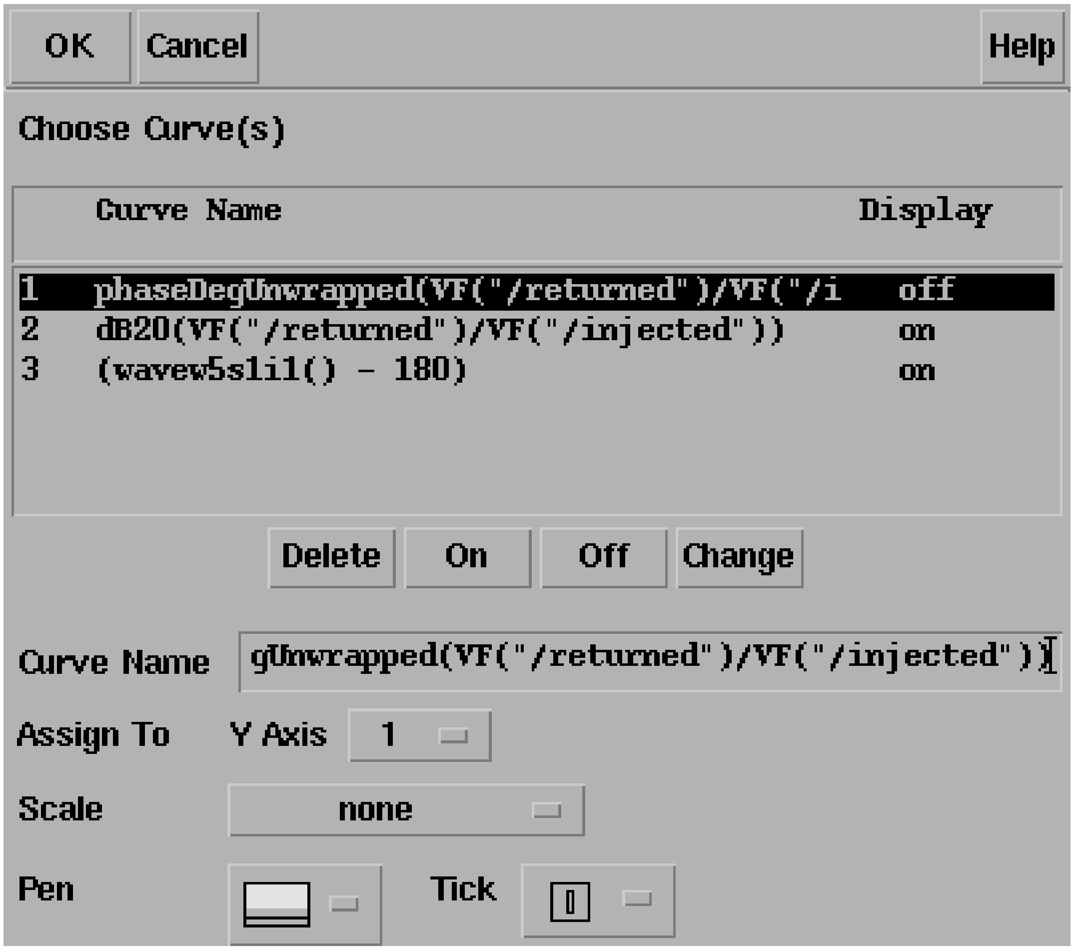
   The Calculator buffer should look similar to the following

7. Click the *plot* button to plot the calculated waveform.

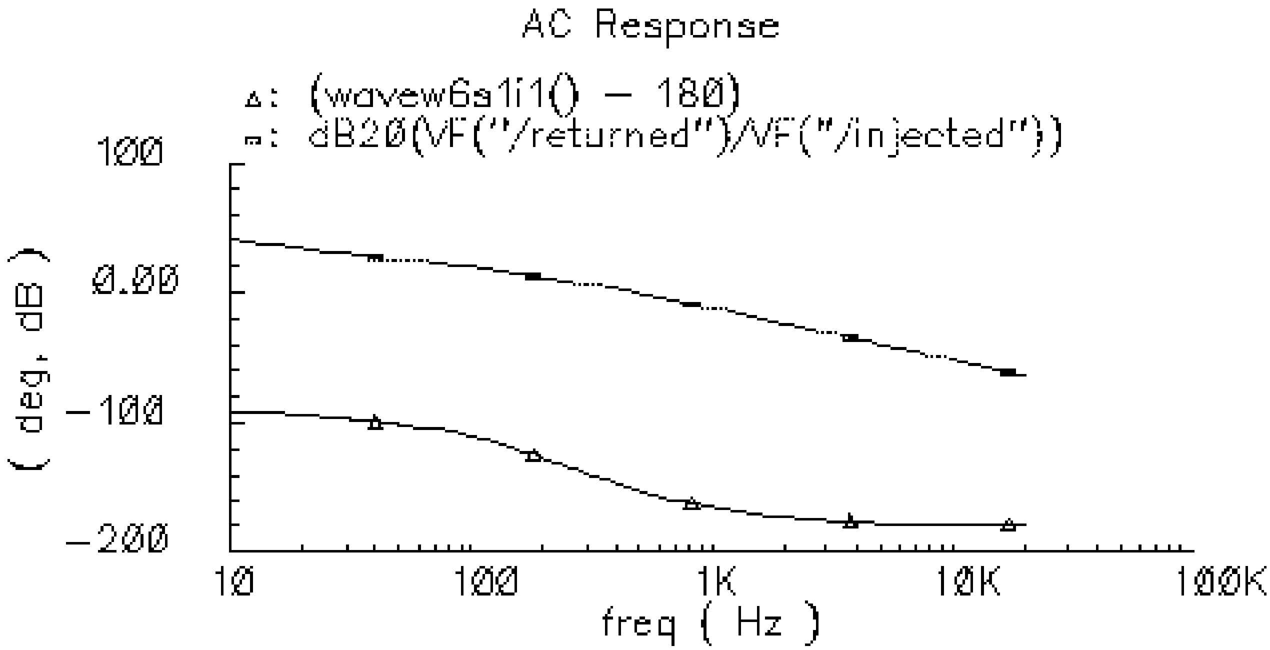


8. To remove the original phase curve from the Waveform window, in the Waveform window
   - **a.** Choose *Curves—Edit* to display the Curves form.
   - **b.** In the Choose Curves list box, highlight the original phase curve.
   - **c.** Click *Off*.

The Curves form looks similar to the following.



The original phase waveform is no longer displayed in the Waveform window.



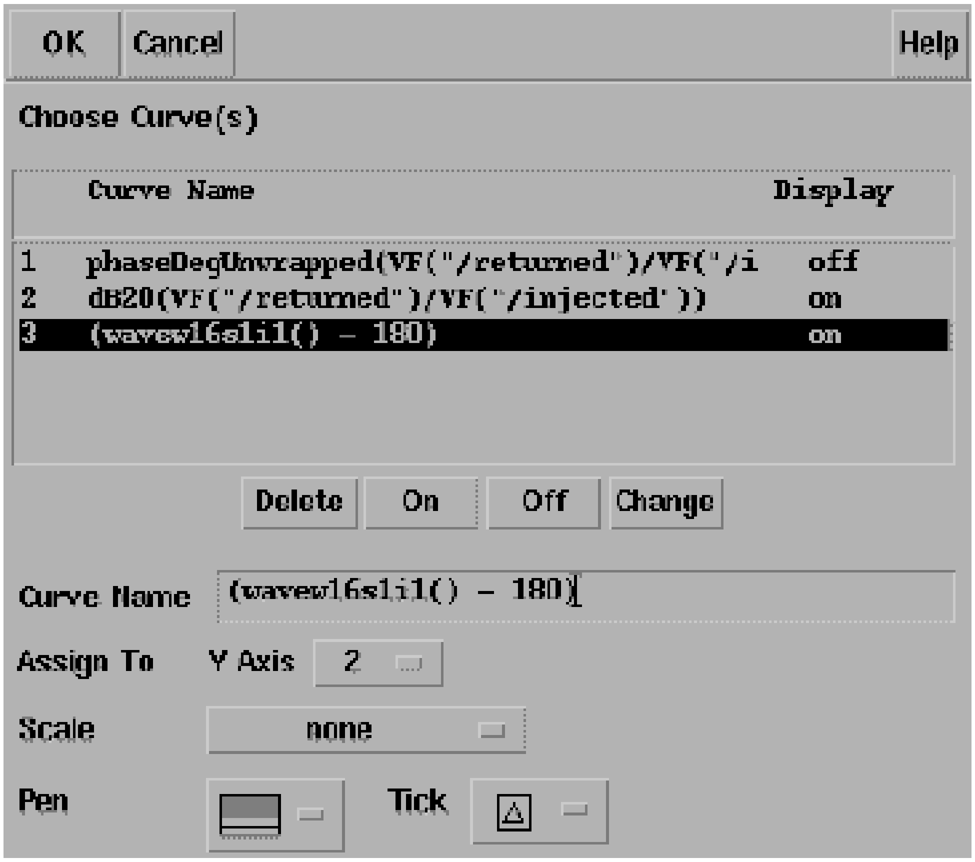
9. To create the Bode plot

   a. If necessary, in the Waveform window, choose *Curves – Edit* to display the Curves form.

   b. In the Curves form, select the shifted (by 180 deg) phase curve

   c. In the Curves form *Assign to Y Axis* cyclic field, select *2*.

The Curves form looks similar to the following.

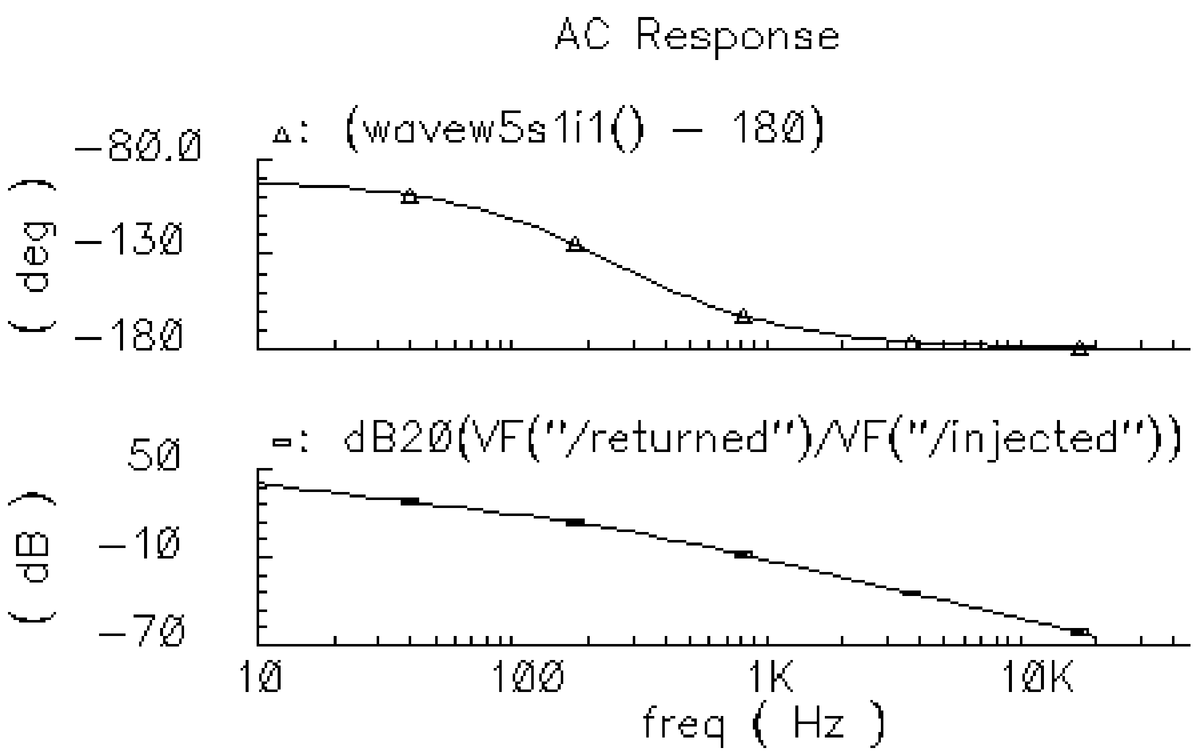


10. Click *OK* in the Edit Curves form.

11. In the Waveform window, choose *Axes—To Strip* to change the display as follows.

    This produces a Bode plot (magnitude and phase) of the loop gain in the Waveform window Shown in Figure G-9.

Figure G-9 Bode Plots, Magnitude and Phase of Loop Gain

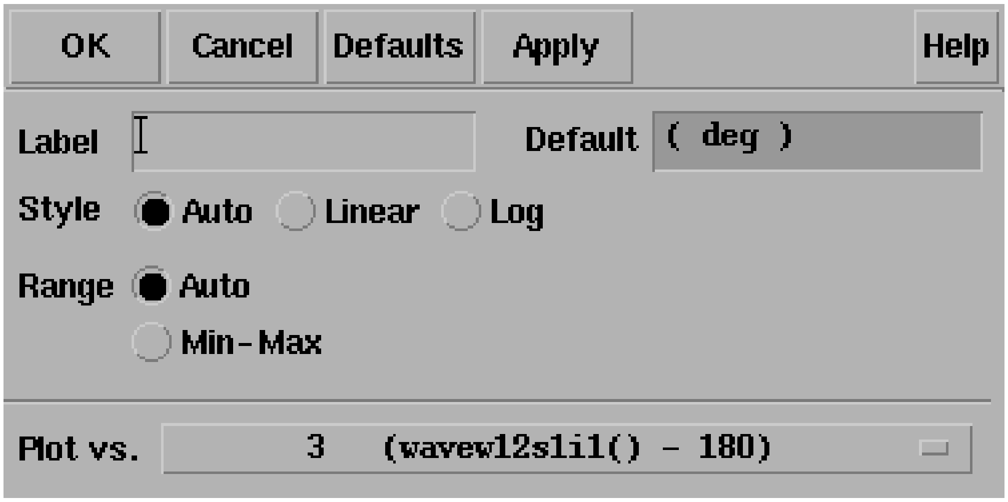


12. To generate a Nichols chart (dB versus degrees) from which you can pick off phase and gain margins,

    a. In the Waveform window, choose *Axes – X Axes*

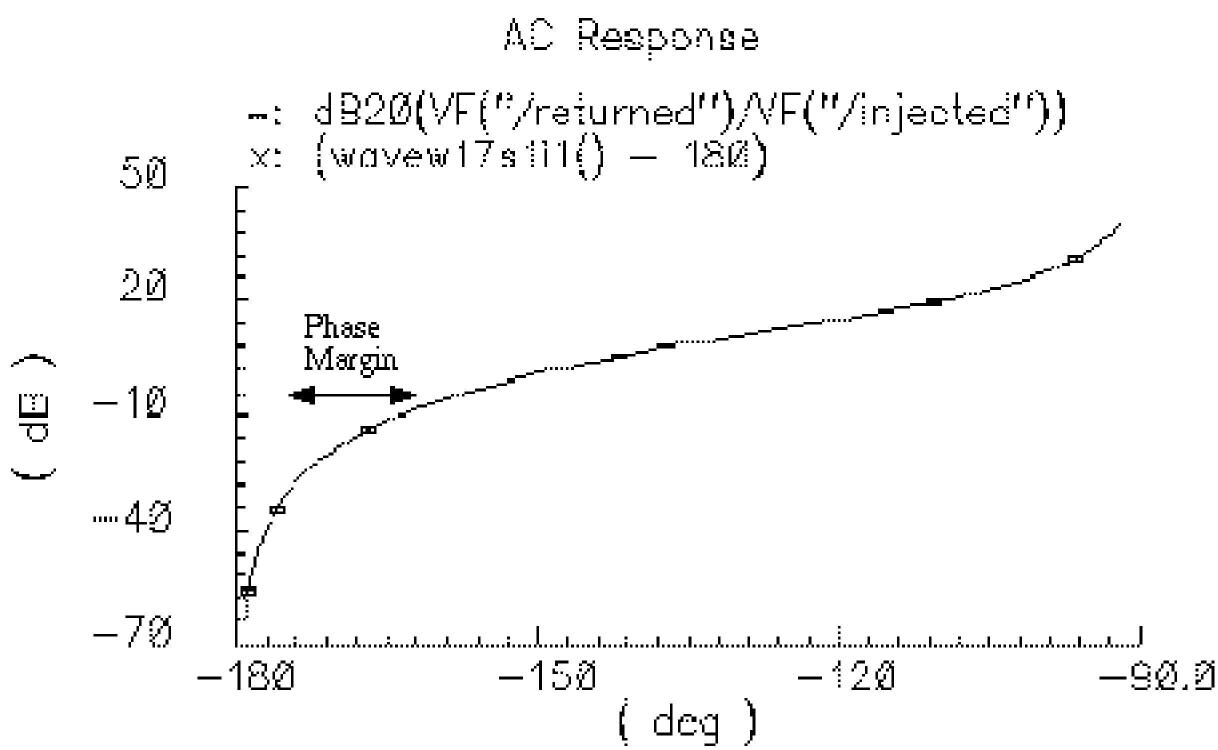
    b. In the *Plot vs. cyclic* field select the phase curve you created that is 180 degrees out of phase.

c. Click *OK*.



You now have a Nichols chart like the one in Figure G-10. The phase margin is 30 degrees.

Figure G-10 Nichols Chart of Loop Gain



To compute phase margin directly do the following:

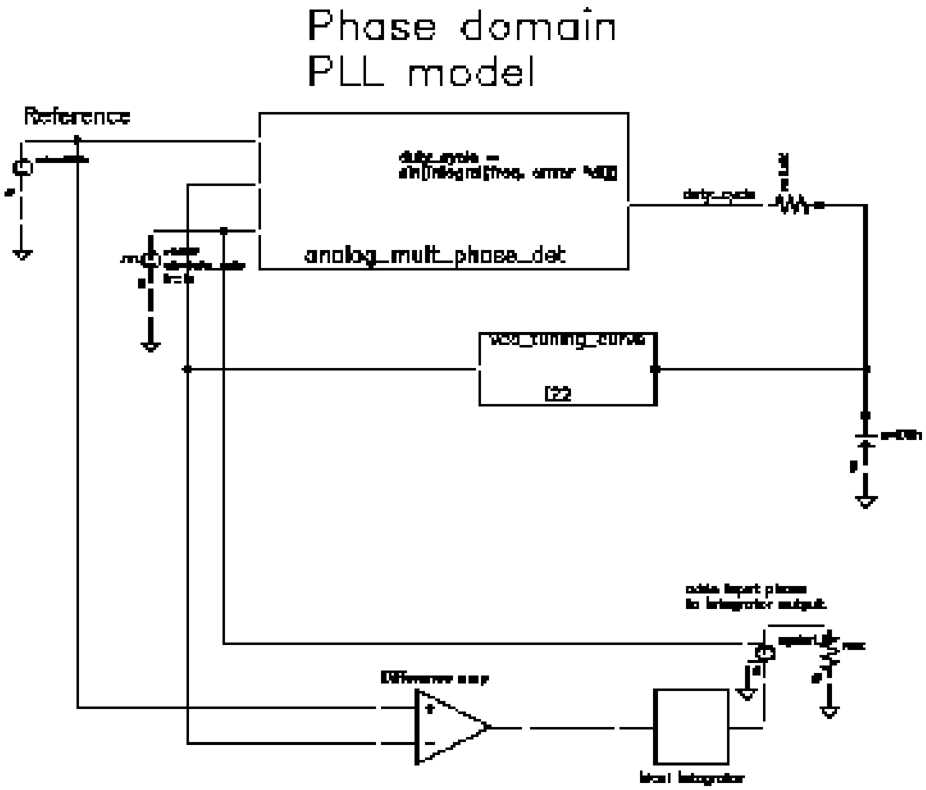
1. In the Waveform Calculator, click on *vf*.
2. In the Schematic window, click first on the return node and then on the injected node.
3. In the Waveform Calculator, click on the *divide* button.
4. In the Waveform Calculator Special Functions menu, choose *phase margin* followed by *print*.
5. Add 180 degrees to the expression in the waveform calculator then choose *print* from the Special Functions menu.

    The Results Display Window displays the phase margin.

Example 3: PM Input

The circuit used to test for PM input is *example_PM* in the *pllLib* library. The PM (phase modulation) input pin is useful if the PLL is used as a modulator or demodulator, but it also provides a convenient place to perturb the PLL to assess large signal stability. Figure G-11 on page 1282 shows a test circuit for such a stability check.

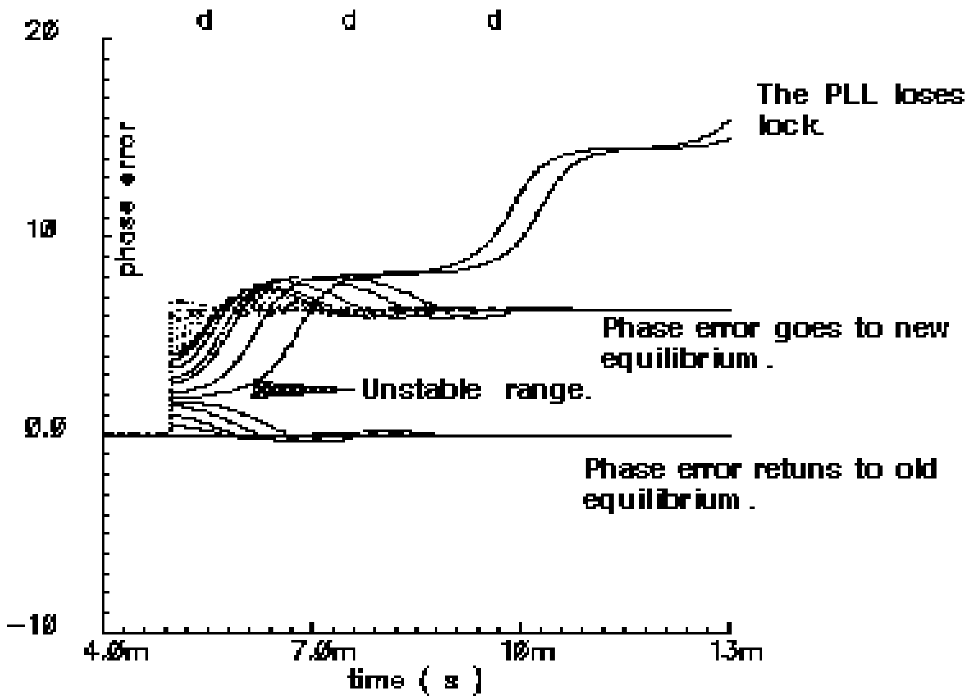
Figure G-11 Stability Check Using the PM Input



This PLL is very simple but yet different from the previous example. It was modified to produce more interesting results. The lower circuitry requires explanation. The difference amplifier computes frequency error and converts it from Mhz to rad/sec. The integrator computes VCO and reference contributions to phase error. The voltage-controlled-voltage-source at the end adds the input phase stimulus to compute total phase error. The difference amplifier and integrator are from the *ahdLib*.

The input phase is a delayed step. The delay makes the initial phase error easy to read. A parametric analysis on the phase error's step response with respect to the size of the input phase step reveals some interesting behavior. Figure G-12 on page 1283 shows the family of phase error step responses produced by the parametric analysis. The external integrator is intentionally not a circular integrator like the one inside the phase detector model. For large and small steps in input phase, the PLL settles into equilibrium, possibly a new one. However, a narrow intermediate range of steps puts the PLL into an unstable mode. The references examine this behavior in mathematical detail [1,4,6,10]. This example shows one way to assess large-signal stability and to demonstrate that the phase-domain models capture the major non-linear mechanisms.

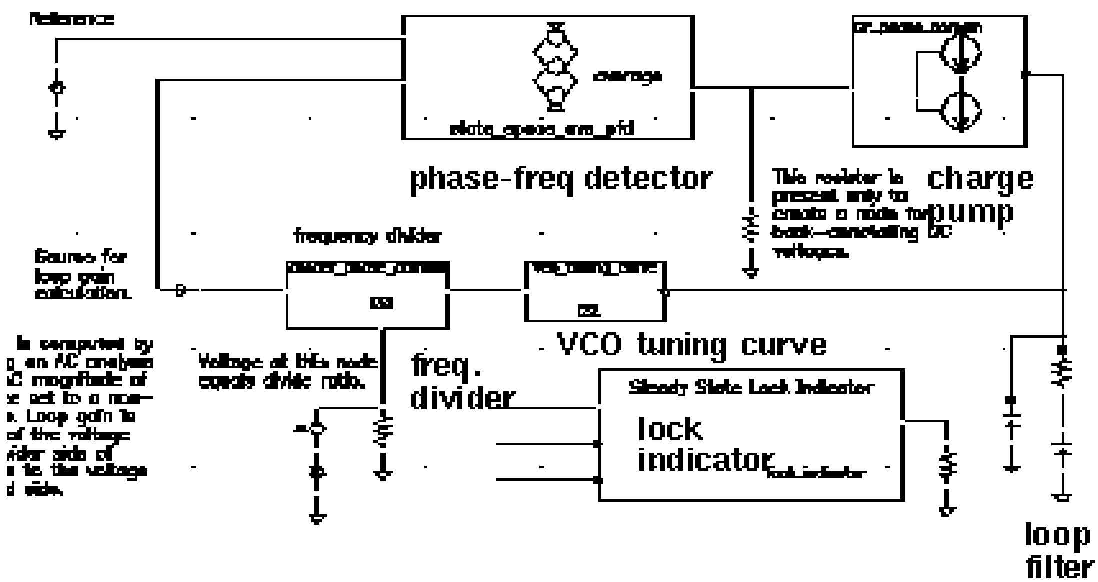
Figure G-12 Phase Error Response to Step PM Input



Modeling a PFD-Based PLL

Figure G-13 on page 1284 shows a block diagram of a typical PLL with a phase-frequency detector. This section describes how to specify each component in Figure G-13 and briefly explains what each model does.

Figure G-13 PLL Block Diagram



VCO

The VCO is modeled by its tuning curve. The tuning curve characterizes the relationship between the input voltage and the output frequency. The input to the VCO model is the loop filter output voltage, also called the VCO control voltage. The VCO output is a voltage representing the VCO's instantaneous frequency in Mhz. Therefore, when the VCO operates at 2 Mhz, the model output is 2 Volts.

The VCO tuning curve is generally nonlinear and can be specified in one of two ways:

- With the coefficients of a fourth order polynomial
- With a look-up table

Polynomial tuning curve: The input voltage is internally clamped to the nearest end point if it moves outside the interval [*min-vco-input-voltage*, *max-vco-input-voltage*]. Although the input voltage may fall outside the interval, the output behaves as though the input voltage value is at the end points. Within the interval, the output is a fourth order polynomial in the quantity, V_{input} minus the free running voltage. When the input voltage equals the free running voltage, the output frequency equals the free running frequency. The scale factor scales the entire polynomial and has a default value of 1. The scale factor is useful in converting data in Khz/volt, for example, to the required Mhz/volt. The parameters are the coefficients of the polynomial.

Table look-up tuning curve: The two parameters are the *scale factor* and the *path to the look-up data*. The look-up model linearly interpolates between data points and linearly extrapolates outside the data interval. The data format is two columns of data delimited by spaces. There is no header, and there are no extra lines at the end. The first column is input voltage. The second column is output frequency. The path to the data can be absolute or relative to the netlist. The netlist is usually stored at

```
<home>/simulation/ckt_name/spectre/schematic/netlist/input.scs
```

but you can choose a different location.

If the data is at

```
<home>/data/table
```

and the netlist is as shown above, the relative path is

```
../../../../../data/table.
```

Frequency Divider

The frequency divider is essentially a simple gain element. It takes an input voltage that represents frequency in Mhz, and then scales it by the divide ratio to generate an output voltage that represents the divided frequency. The divide ratio is numerically equal to the voltage on the control pin. If the divide ratio drops below 0.001, the model assigns it to 0.001 and issues a warning. This assignment prevents division by zero during simulation.

Charge Pump

The charge pump transforms the duty cycle into the expected average current sourced or sunk by the charge pump. You define the maximum source and sink currents, and they can be different from each other. If the charge pump output voltage exceeds the rails you define, the output voltage is clamped to the rail through a 0.001 Ohm resistance. The other parameters are the leakage resistance and open circuit voltage. These last two parameters specify the Thevenin equivalent circuit of a leakage path. The leakage path can source or sink current depending on the open circuit voltage.

Loop Filter

The loop filter is entered component-for-component.

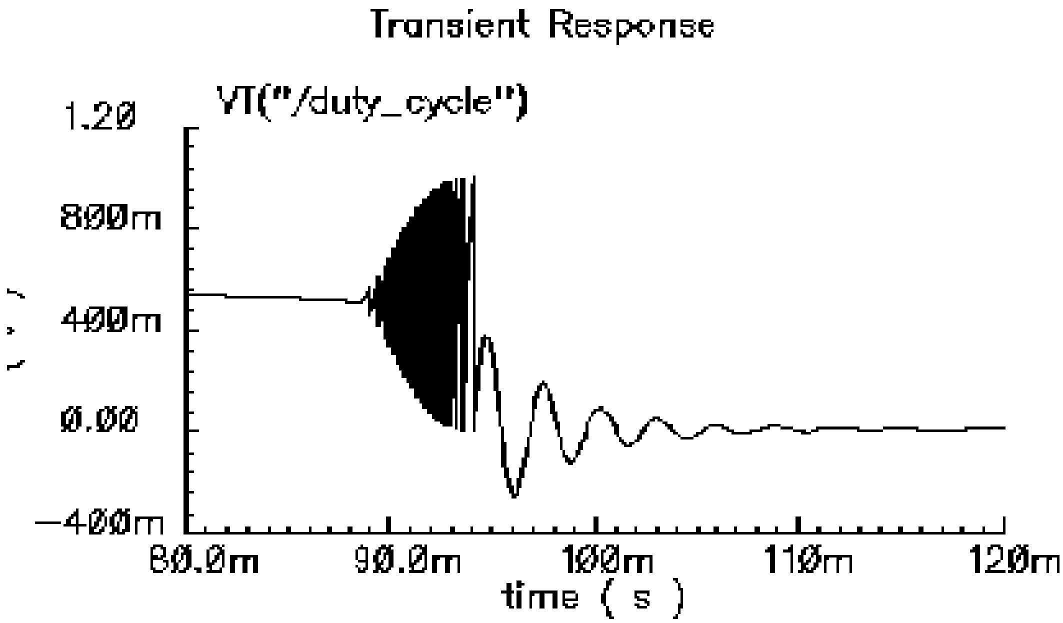
State-Space Averaged PFD (Phase-Domain Phase-Frequency Detector Model)

The phase-frequency detector (PFD) model approximates average behavior of a digital, three-state, phase frequency detector. This is the most complicated model in the PLL library. The term *state-space averaged* is borrowed from the power electronics field [14]. The charge pump currents for the three PFD states are averaged together with a duty cycle much like voltages are averaged together with a duty cycle in a switch-mode power supply model. The PFD inputs are voltages representing the reference and divider output frequencies in Mhz. The output is a voltage that when multiplied by the maximum charge pump current, numerically equals the average charge-pump output current. The PFD output is a duty cycle. When the frequency error is large, the duty cycle is a smooth waveform directly related to the normalized frequency error [1,4]. When the frequency error is small, an integrator inside the PFD model converts frequency error to phase error, and the duty cycle is proportional to the phase error. The duty cycle starts jumping to zero (or resets) as it changes from frequency-mode to phase-mode.

As phase error enters a deadband determined by the minimum-on-time parameter and reference frequency, the model computes a duty cycle pulse with magnitude one and duration equal to the minimum-pulse-width. After the pulse expires, the duty cycle drops to zero until the phase error exits the deadband. As the phase error exits the deadband, the duty cycle increases to a non-zero value. The deadband and fixed-width unity pulse simulate what some texts call *backlash* [8].

The PFD model has two parameters. The first is a numerical option that controls the trade-off between execution speed and accuracy. The *speed_vs_accuracy* parameter controls the number of times the internal integrator is reset during the transition from frequency-mode to phase-mode. Too few resets can cause error. Too many resets can needlessly slow run time. The default value of this parameter is 50k. To reduce the number of resets in a slow PLL, and thereby reduce run time, increase the *speed_vs_accuracy* parameter to 70k or 100k. To increase the number of resets in fast PLLs, and thereby increase the accuracy, reduce the *speed_vs_accuracy* parameter to 10k or 20k. A reasonable setting for the *speed_vs_accuracy* parameter produces a duty cycle step response that resets approximately to zero at least 3-10 times before entering the final transient. Figure G-14 on page 1287 shows reasonable duty cycle step response.

Figure G-14 Reasonable Duty Cycle Waveform



The other parameter is the *minimum_on_time* which controls the backlash. This is the minimum pulse width the PFD can generate. As the phase error decreases, the pulse width drops discontinuously from the minimum pulse width to zero. This effect creates a deadband in the duty cycle versus phase error curve.

Figure G-12 on page 1283 was generated with the default *minimum_on_time* parameter value of zero µs. The default value of the *minimum_on_time* parameter produces no deadband and no unity pulses. The default deactivates the backlash mechanism.

Figure G-15 on page 1288 was generated with a *minimum_on_time* parameter value of 0.2 µs. Figure 15a illustrates that the pulses only occur as the phase error enters the deadband. Figure 15b shows the limit cycle created by the backlash. The limit cycle is primarily determined by leakage on the loop filter and the minimum pulse width. Some references suggest biasing the duty cycle away from the deadband or loading the filter down to force the limit cycle frequency to a value in which the loop filter attenuates it. The phase-frequency detector model can help quantify the problem and check the solution.

A pulse is not kicked out upon exiting the deadband because that behavior causes convergence problems for Spectre. If phase error is entering the deadband, a pulse at that moment pushes phase error in the same direction it was going, into the deadband. If a pulse occurs as phase error exits the deadband, the pulse drives phase error back into the deadband and Spectre has trouble figuring out whether phase error should leave the deadband at all. Fortunately, no significant error is introduced by implementing the pulse only when phase error enters the deadband. In a backlash limit cycle, the feedback loop quickly drives phase error back into the deadband and a pulse occurs on the way in. The error is in the time the feedback loop takes to return phase error to the edge of the deadband and that is usually small when PLL is in a backlash limit cycle.

(The PLL model that generated Figure 15 had a center frequency of about 1Mhz and the simulation ran out to 130ms. A voltage-domain model might easily simulate 10 points per carrier cycle. The voltage-domain model would require 1.3 million points to simulate the same amount of action. I did not attempt it. The phase-domain simulation that generated Figure G-15 ran in a matter of seconds!)

Figure G-15 Duty Cycle Waveforms With Pulses and Backlash

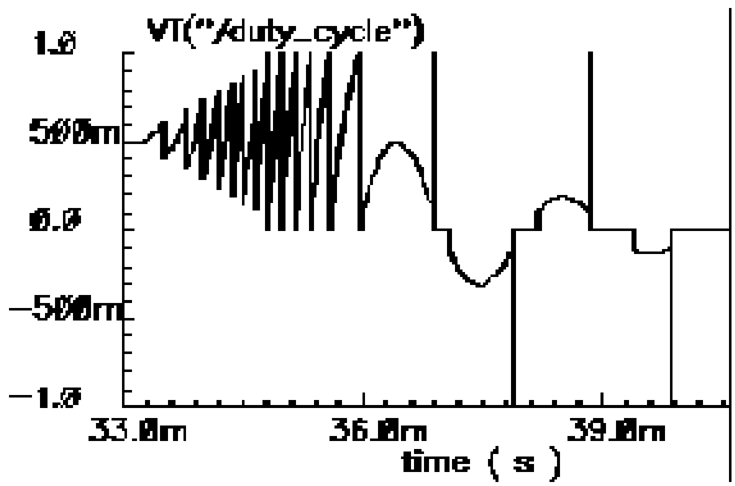


Figure 15a
Duty cycle between
33 ms and 40 ms

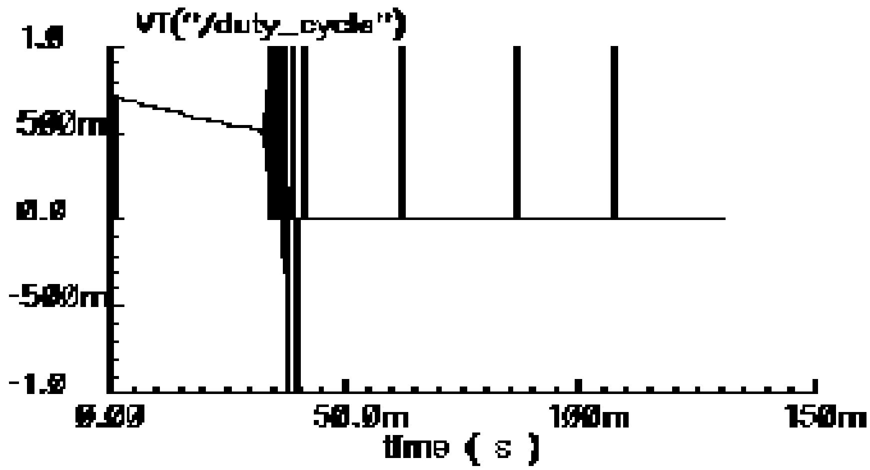


Figure 15b
Duty cycle to
130 ms

Lock Indicator

All real components have limited outputs and the limits of any one component can keep the PLL from locking. Just like the simple phase detector model, all of the phase-domain models operate one way for DC analysis and another for transient analysis to prevent DC convergence errors. The lock indicator monitors three signals. In the example, the lock indicator monitors the phase detector output, the VCO control, and the charge pump output. If any of those signals exceeds its limit, the lock indicator output is zero, signifying that the loop is not locked in steady state. If all signals are within their limits, the output is 1 volt, specifying that the loop is locked. The lock indicator is only valid for DC analysis. Use node names to tie the lock indicator inputs to the right nodes and use variables for the component limits. You must specify the units manually twice, once for each component and once for the lock indicator. With variables, the lock indicator parameters are linked to the proper component parameters, and you specify changes in only one place.

Example 4: Modeling Acquisition Transients

The circuit used to model acquisition transients is the *example_phase_domain* in the *pllLib* library. Figure G-16 on page 1290 shows the duty cycle and VCO frequency response to a momentary change in the divider ratio. When the divider ratio changes, the PFD enters the frequency-mode and slews the VCO frequency toward the new value. As the VCO frequency approaches the final value, the PFD model gradually changes from frequency-mode to phase-mode. When the frequency error is small, but still large enough to slew phase error, the duty cycle waveform looks like a sawtooth waveform. The model gradually increases the amplitude of the sawtooth component of the duty cycle, and always maintains the correct average, as frequency error reduces to zero. The final duty cycle transient is the sawtooth that depends mostly on phase error. Figure G-12 on page 1283, modifies the x-axis of the graph to show the duty cycle in the first transition.Figure G-17 on page 1290 modifies it further to show the sawtooth waveform.

Figure G-16 Response to Momentary Change in Divider Ratio

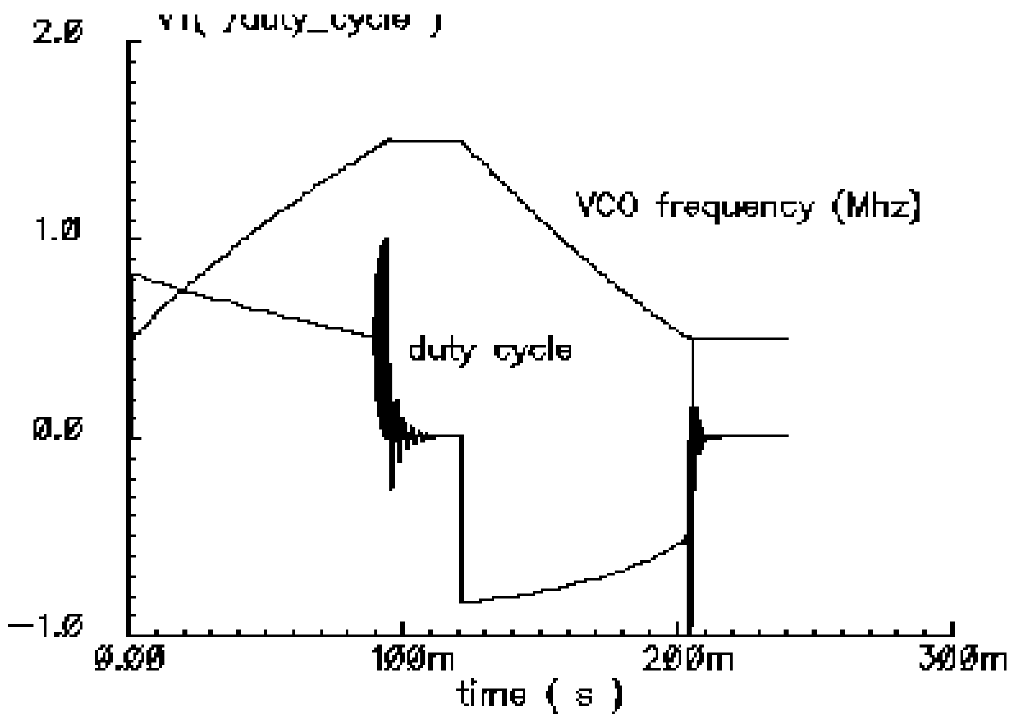
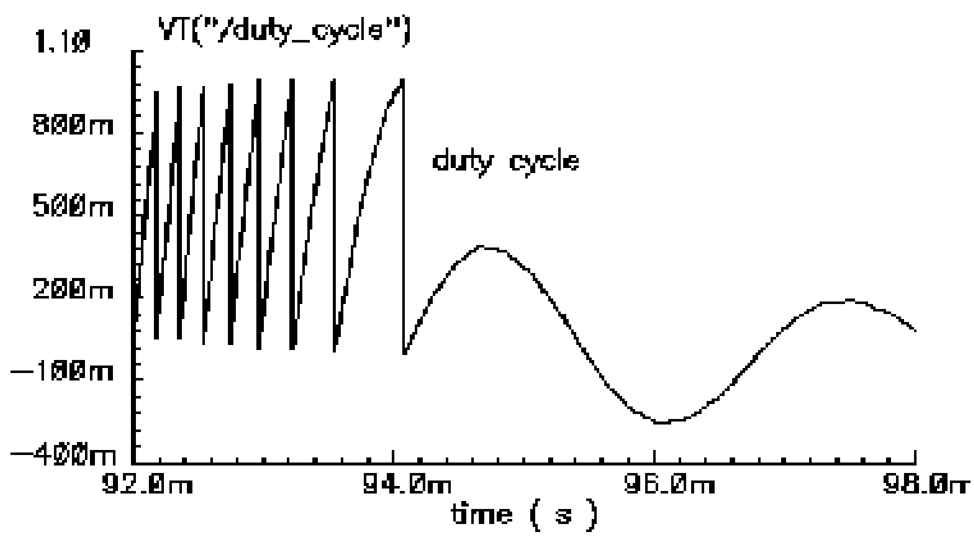


Figure G-17 Duty Cycle During Transition From Frequency to Phase Mode

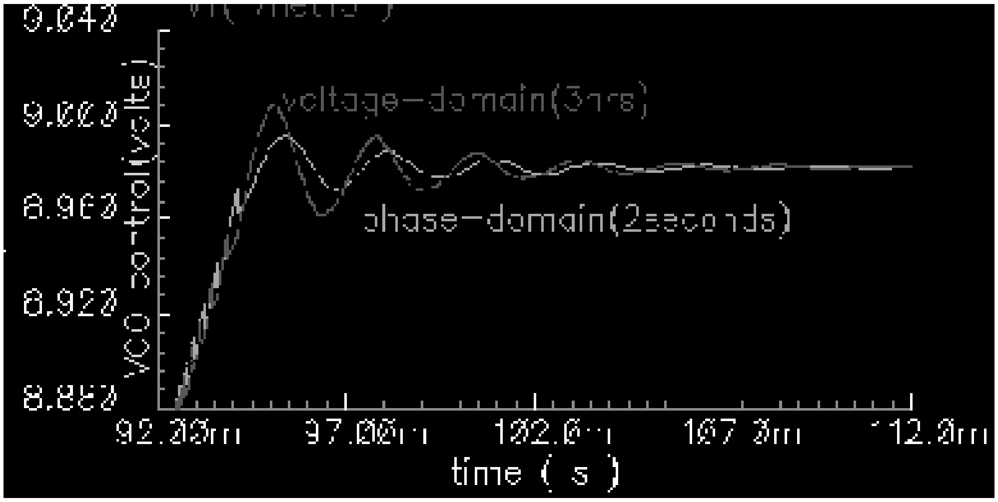


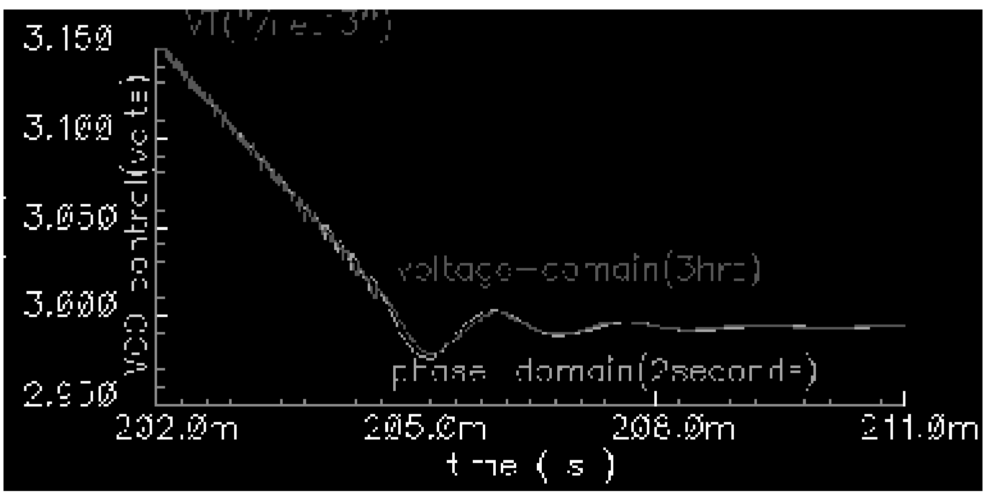
Example 5: Comparison With a Voltage-Domain Model

Use the *example_voltage_domain* model in the *pllLib* library with the only node that can be directly compared between phase- and voltage-domain models, the VCO control node. At full scale, the difference between the two models is not visible. The differences occur at the transitions. Figure G-18 on page 1292 compares the two models at the transitions. In this example and on the same machine, the voltage-domain model simulates in three hours while the phase-domain model simulates in two seconds.

The error between the two models does not appear to be consistent. It is larger in the first transition. Furthermore, decreasing the *speed_vs_accuracy* parameter does not always increase the similarity of the waveforms. This is because the final transient, the one driven primarily by phase error, depends on the residual frequency error at the time the phase error last crossed 2π. The frequency error at that moment, especially after a long frequency slewing period, is sensitive to, among other effects, initial conditions.

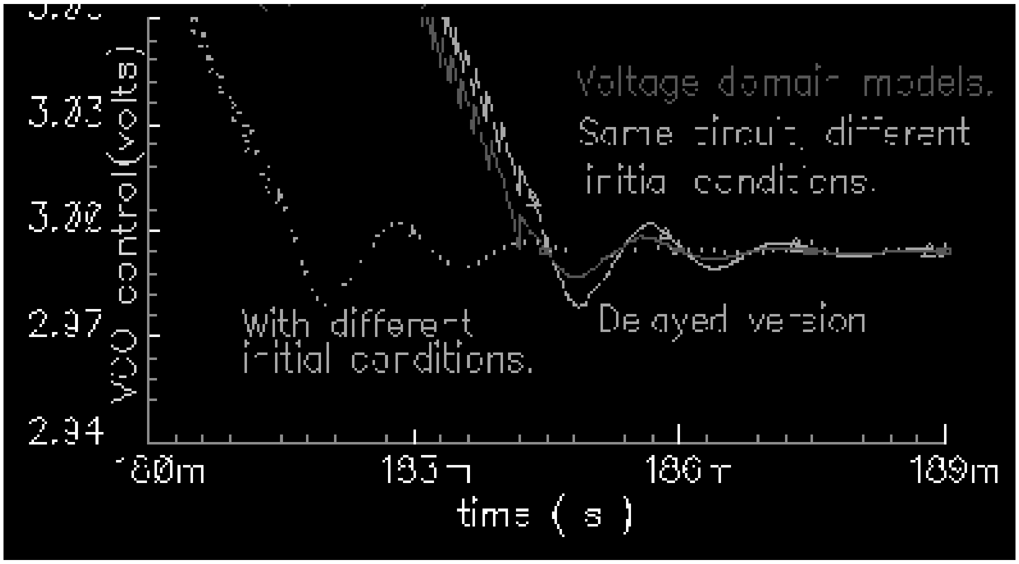
Figure G-18 Comparison with Voltage Domain Model





The voltage-domain model therefore shows the same level of variation for small differences in the initial conditions preceding the transition to the new equilibrium. Figure G-19 on page 1293 compares two voltage-domain simulations of the VCO control signal during the second transition. One of the voltage-domain simulations (the dotted waveform) used different initial conditions. The solid waveform is a delayed version of the dotted waveform. The delay overlays the two simulations for direct comparison. The error is comparable to the error between voltage-domain and phase-domain simulations.

Figure G-19 Effect of Initial Conditions



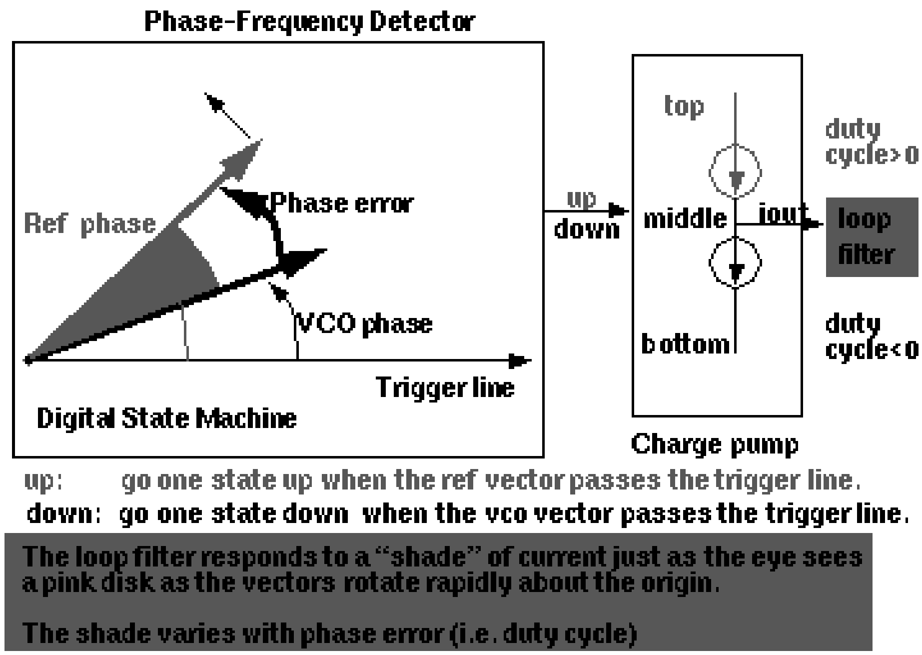
How the PFD Model Works

The heart of the PFD is a digital state machine [8,9].The model is for PFDs with three digital states. The output stage is usually a charge pump or pair of switches. It is convenient to model the PFD in two pieces. The first piece models the state machine and computes a duty cycle that is independent of the output stage. The second piece models the output stage. The charge pump (CP) is used here as an example.

How the PDF/CP Pump Works

Let the three PFD states be stacked. In the top state, the PFD commands the CP to source current. In the middle state, the pump is off. In the bottom state, the CP sinks current. The PFD is edge triggered. Figure G-20 on page 1294 shows vectoral representations of the reference and VCO clocks [1,4]. Both vectors rotate counter-clockwise around the origin. The angle between the hands equals phase error. Phase error lies between +-2π. Whenever the reference passes a trigger line, like 3 o'clock, the state jumps to the next state up. If the PFD is already in the top state, the state does not change. Whenever the VCO passes the trigger line, the state jumps to the next state down. If it is already in the bottom state, it again does not change. For a fixed phase error, the state toggles as the hands rotate. State toggles between the middle and top, or between the middle and bottom states. The percentage of time spent in the top, or bottom, state is the duty cycle. The duty cycle is positive for top to middle toggling and negative for middle to bottom toggling.

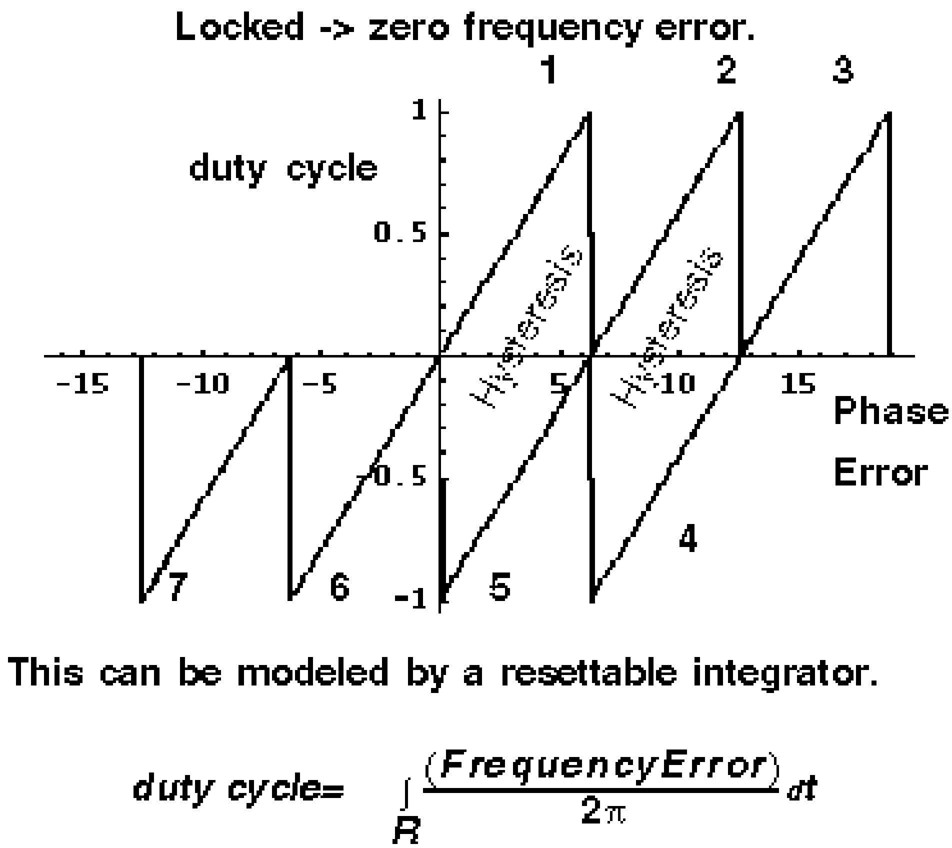
Figure G-20 PFD Operation



If the reference and VCO frequencies are identical, the vectors in Figure G-20 rotate together. Let the sector defined by phase error be red if the reference leads and blue if it lags. If the hands rotate once per minute and the reference leads, you see two colors, white and red. At two million revolutions per second, you see pink. The shade of pink depends linearly on the phase error. Although PFD output current toggles between two values, the loop filter and VCO respond mainly to the "shade" of current. The shade is proportional to the duty cycle. With zero frequency error, duty cycle equals phase error divided by 2π. Existing literature uses one function to describe the *phase-error-to-duty-cycle* relationship and a different function to describe the *frequency-error-to-duty-cycle* relationship. These two functions are the *locked duty cycle function* and *averaged unlocked duty cycle function*, respectively. The new model combines these two functions into one practical model.

The locked duty cycle function is a multivalued sawtooth. For monotonic movements away from the origin of steady-state phase error, the duty cycle is a sawtooth in the upper-half plane. The duty cycle lies in the lower-half plane for negative movements. If a movement starts off positive, then changes direction, the duty cycle crosses zero and becomes a sawtooth in the lower half plane. The duty-cycle-phase-error trajectory encloses a nonzero area as shown by the {1-2-3-4-5-6-7} sequence of peaks in Figure G-21 on page 1295. This is a good reason for putting the integrator next to the non-linearity—hysteresis involves memory and the integral supplies it.

Figure G-21 Duty Cycle Versus Phase Error With Zero Frequency Error

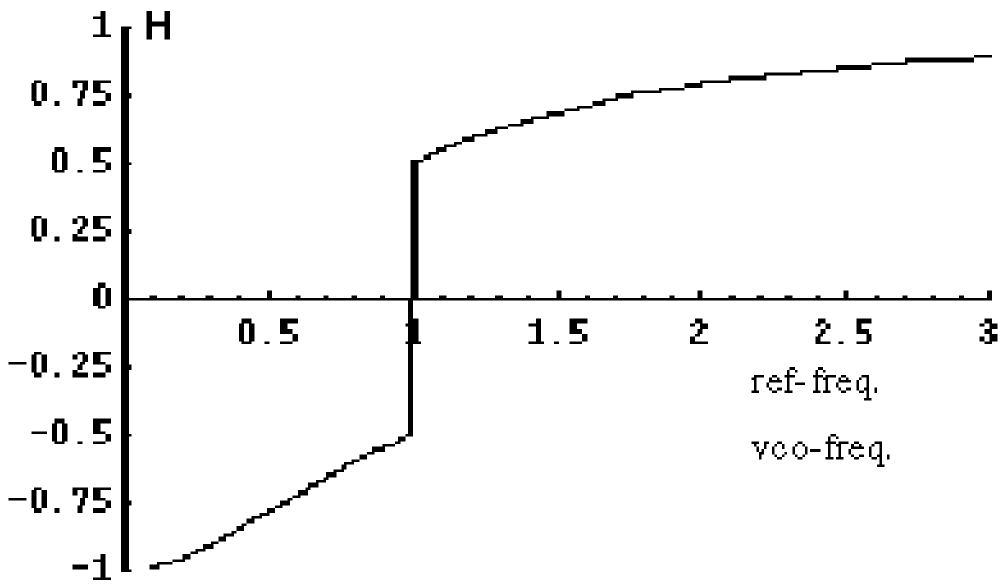


The new model operates only on frequency error. For small-frequency errors, the duty cycle is indeed proportional to the phase error. The phase error is the integral, with respect to time, of the frequency error. The duty cycle therefore equals the integral of the frequency error divided by 2π. Resetting the integral whenever it hits $+-2\pi$ produces the multivalued sawtooth described above. If the frequency error changes sign, the resetting integrator ramps to zero, passes through zero, and generates a sawtooth in the lower-half plane. The phase-error-duty-cycle trajectory is precisely the multivalued sawtooth described above. The resettable integrator (RI) merges the integrator of a phase-domain model with the locked duty-cycle function.

For a sustained frequency error, the RI model predicts an average duty cycle of +-1/2 regardless of error size. This is correct only for small-frequency errors. The true duty cycle goes to +-1 for large-frequency errors. Let the reference frequency far exceed VCO frequency. Whenever the VCO passes the trigger line, the reference frequency passes shortly thereafter. The reference frequency might pass the trigger line several more times before the VCO passes again. In this case, the phase error is still a sawtooth, but the average duty cycle is nearly 1. This behavior lets the PLL acquire input signals faster. The function H, in

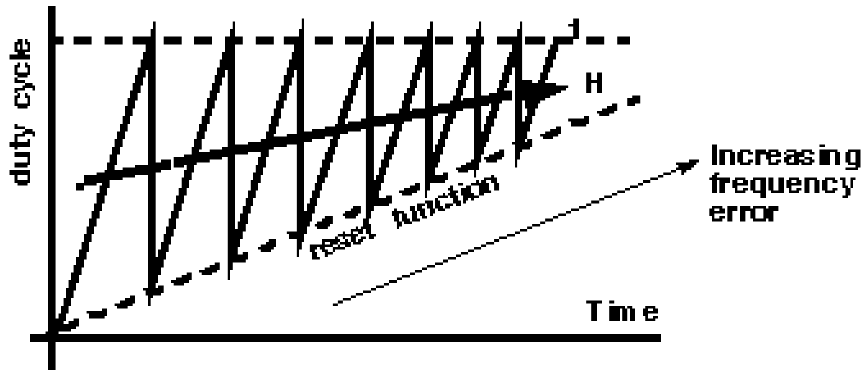
Figure G-22 on page 1296, shows the averaged unlocked duty cycle. This cycle depends on the ratio of the reference to VCO frequencies and is discontinuous where frequency error is zero.

Figure G-22 Averaged Unlocked Duty Cycle



The RI is modified to include the frequency effect. For small-frequency errors, the predicted average duty cycle equals 1/2. This is true because the RI runs from the reset point (=0) to the reset threshold (=2π). It is not necessary to reset the integrator to zero. Resetting the integrator to a “reset” function gives the correct average duty cycle (Figure G-23). As the frequency ratio goes to +-infinity, the reset point changes to +-2π. Because the reset threshold is still +-2π, the predicted average duty cycle changes to +-1. As the frequency ratio approaches unity, the reset point returns to zero, and the predicted average duty cycle returns to 1/2.

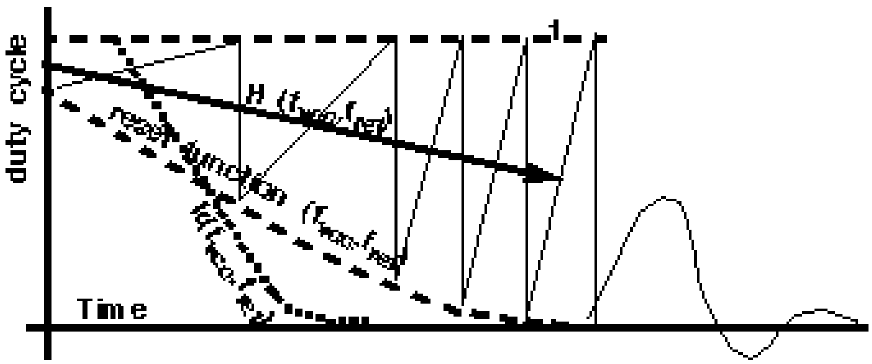
Figure G-23 Combining Averaged Locked and Unlocked Duty Cycles



$$\textit{duty cycle} = \int_{R(\textit{frequncyerror})} \frac{(\textit{FrequencyError})}{2\pi} dt$$

The state space averaged PFD model requires one more addition to be practical. As the reset point nears +-2π, the integrator resets very frequently and execution stalls. The integrator must be deactivated for large-frequency errors. The new PFD model uses the weighted sum of a RI and the H. The weighting factors are *k* and (1-*k*). *k* is a function of the ratio of the two input frequencies. *k* approaches 1 for large-frequency errors and approaches 0 as the frequency ratio approaches unity. A factor of (1-*k*) under the integral deactivates the integral for large-frequency errors. The resulting PFD model looks like H for large-frequency errors, and it looks like the reactivated RI for small-frequency errors. *k* determines how fast the RI reactivates and how gradually the model changes from H to the RI. A *speed_vs_accuracy* parameter controls *k*. If the model does not reset a few times before reaching frequency lock, you can improve the results by decreasing the *speed_vs_accuracy* parameter. If the model resets so often that the simulation is too slow, you can speed execution by increasing the parameter. The default setting of 50000 covers a wide range of loop speeds. Figure G-24 on page 1298 shows the transition from frequency-mode to phase-mode.

Figure G-24 Complete Model



$$duty\ cycle = k^*H + (1\text{-}k)\int_R (1-k)\frac{(FrequencyError)}{2\pi}dt$$

References

[1“Loops, Theory and Application,“ J. L. Stensby.

[2]“Relative -phase modeling speeds Spice simulation of modulated systems,” John Kesterson, November 11, 1993 issue of EDN

[3]“Simulation of Communication Systems,” M. Jeruchim, P. Balaban and K. Shanmugan. Plenum, 1992.

[4]“Synchronization in Digital Communications,” Heinrich Meyr and Gerd Ascheid, Published by John Wiley and Sons.

[5]“Phase-Locked and Frequency-Feedback Systems,” Klapper and Frankle, Published by Academic Press

[6]“Phaselock Techniques,” Floyd Gardner, Published by John Wiley and Sons

[7]“Non-Linear Relative Phase Models of Phaselock Loops,” Jess Chen, Cadence Technical Conference, 1996.

[8]“Phase-Locked Loops: Theory, Design, and Applications,” Roland Best, Published by McGraw-Hill

[9]“Phase-Locked Loop Circuit Design,” Dan Wolaver, Published by Prentice Hall

[10]“Phase-Lock Basics,” Willeam F. Egan. Published by John Wiley and Sons.

[11]“Non-Linear State Space Averaged Modeling of a 3-State Digital Phase-Frequency Detector,” Jess Chen, Cadence Technical Conference, 1997

[12] “Analog and Mixed-Signal Hardware Description Languages,” Edited by A. Vachoux, J. Berge, O. Levia, and J. Rouillard. Kluwer Academic Publishers.

[13] “Macromodeling with SPICE,” J.A. Connelly and P. Choi. Prentice Hall. Pages 168-169.

[14] S. Cuk, Cal. Inst. of Tech. PhD Thesis, “Modelling, Analysis, and Design of Switching Converters.” 1977

H

Using Port in SpectreRF Simulations

The *port* component, located in the *analogLib* library, can be used in RF circuits for SpectreRF and Spectre S-parameter simulations. The *port* component is similar to the existing *psin*, also located in the *analogLib* library. Unlike *psin*, *port* supports all the *Source types* of the *port* primitive in Spectre, specifically: *pwl*, *pulse*, *sine*, *dc*, and *exp*.

The *port* is a resistive source that is tied between positive and negative terminals. It is equivalent to a voltage source in series with a resistor, where the reference resistance of the *port* is the value of the resistor.

In IC4.4.3_QSR1, *port* is supported by Spectre Direct netlisting *only* in the Analog Circuit Design Environment (spectreS, or Spectre through the cdsSpice socket, is not supported).

Capabilities of the port Component

While the *port* component is generally useful as a stimulus in high-frequency circuits, it also has the following unique capabilities:

- Defines the ports of the circuit to the S-parameter analysis
- Has an intrinsic noise source that lets the noise analysis directly compute the noise figure of the circuit
- Is the only source for which the amplitude can be specified in terms of power

Terminating the port

When you specify the voltage on a *port*, Spectre assumes that *the port is properly terminated in it's reference resistance.* The specified voltage value is not the voltage on the internal voltage source, which is actually set to twice the value specified on the `port`. If you use a `port` source to drive an open circuit, the voltage (for DC, transient, AC, and PAC signals) is double its specified value. However, you can alternatively specify the amplitude of the sine wave in the transient and PAC analyses as the power in dBm delivered by the *port when terminated with the reference resistance.*

Port Parameter Types

In this application note, the parameters are grouped by the following types, rather than in the order they appear on the *port* Edit Object Properties form.

Port parameters

Resistance

Port number

Multiplier

General waveform parameters

Source type

Delay time

DC Waveform parameters

DC voltage

Pulse waveform parameters

Zero value

One value

Period of waveform

Rise time

Fall time

Pulse width

PWL waveform parameters

Waveform Entry Method

File name

Number of PWL/Time pairs

DC offset

Amplitude scale factor

Time scale factor

Breakpoints

Period

Transition Width

Sinusoidal waveform parameters

Sine DC level

Frequency name 1

Frequency 1

Amplitude 1 (Vpk)

Amplitude 1 (dBm)

Phase for Sinusoid 1

Frequency name 2

Frequency 2

Amplitude 2 (Vpk)

Amplitude 2 (dBm)

Phase for Sinusoid 2

Amplitude and Frequency modulation parameters

AM modulation index 1

AM modulation frequency 1

AM modulation phase 1

FM modulation index 1

FM modulation frequency 1

Damping factor 1

Exponential waveform parameters

Delay time

Zero value

One value

Rise time start

Rise time constant

Fall time start

Fall time constant

Small-signal parameters

PAC magnitude

PAC magnitude (dBm)

PAC phase

AC magnitude

AC phase

XF magnitude

Temperature effect parameters

Linear temperature coefficient

Quadratic temperature coefficient

Nominal temperature

Noise parameters

Noise Entry Method

Noise file name

Number of noise/freq pairs

Port Parameters

Port parameters include *Resistance, Port Number, and Multiplier*.

Resistance

The reference resistance of the system. The value must be a real number, but not 0. Default: 50 Units: Ω

Port number

The number of the *port*. The value must be a nonzero integer. Each *port* in a schematic must have a different *Port number*. The *Port number* is not automatically indexed when you place each *port* on your schematic.

Multiplier

The multiplicity factor. The value must be a nonzero real number. This number lets you specify a number of *ports* in parallel. For example, if you set *Resistance* to 50 and *Multiplier* to 2, you specify two *ports* in parallel, with an effective reference resistance of 25 Ω. Default: 1

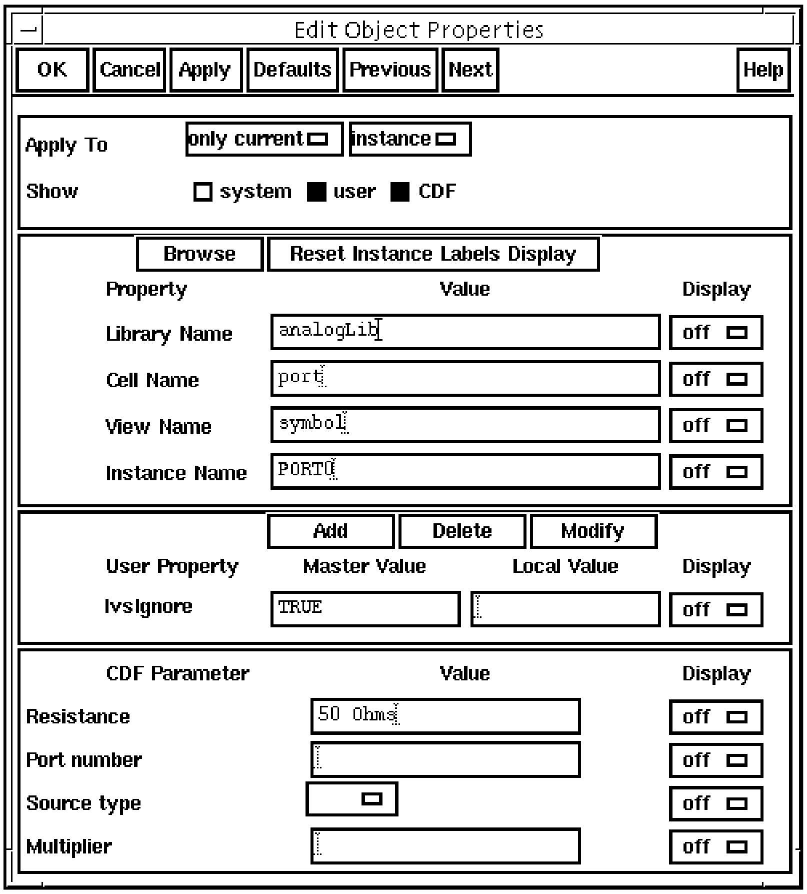
General Waveform Parameters

Source type

The *Source type* parameter lets you specify the parameters for several different wave shapes and then quickly switch between them without losing the settings. Possible values for *Source type* are: *dc*, *pulse*, *exp*, *pwl*, *sine*, or blank. The typical *Source types* used in SpectreRF analyses are *dc*, *pulse*, and *sine*. For example, you can quickly switch from a sinusoid level (for PSS analysis) to a DC level (for PAC analysis) by changing the *Source type* from *sine* to *dc* in the Edit Object Properties form.

If you set *Source type* to a blank value, the *port* acts as a resistive load.

Figure H-1 Source type=blank in Edit Object Properties form



If you set *Source type= dc*, *pulse*, *pwl*, *sine*, or *exp*, you can create a dc, pulse, piecewise linear, sine, or exponential waveform whose parameters you can alter. These parameters are described in more detail in the *DC, Pulse, Piecewise Linear, Sinusoidal,* and *Exponential Waveform Parameters* sections that follow.

Delay time

The waveform *Delay time,* or the time that the source stays at the DC level before it starts generating waveforms (assuming the *Source type* is set to *sine*, *pulse*, *pwl*, or *exp*). The value must be a real number. Default: 0 Units: seconds

DC Waveform Parameters

To generate a dc waveform from the *port* component, set the CDF parameter *Source type=dc* as shown in the figure below. When *Source type=dc*, the *dc* and *temperature effect* parameters are active and set the DC level for all analyses. In DC analysis, this setting also determines the DC level generated by the source, regardless of what *Source type* you specify.

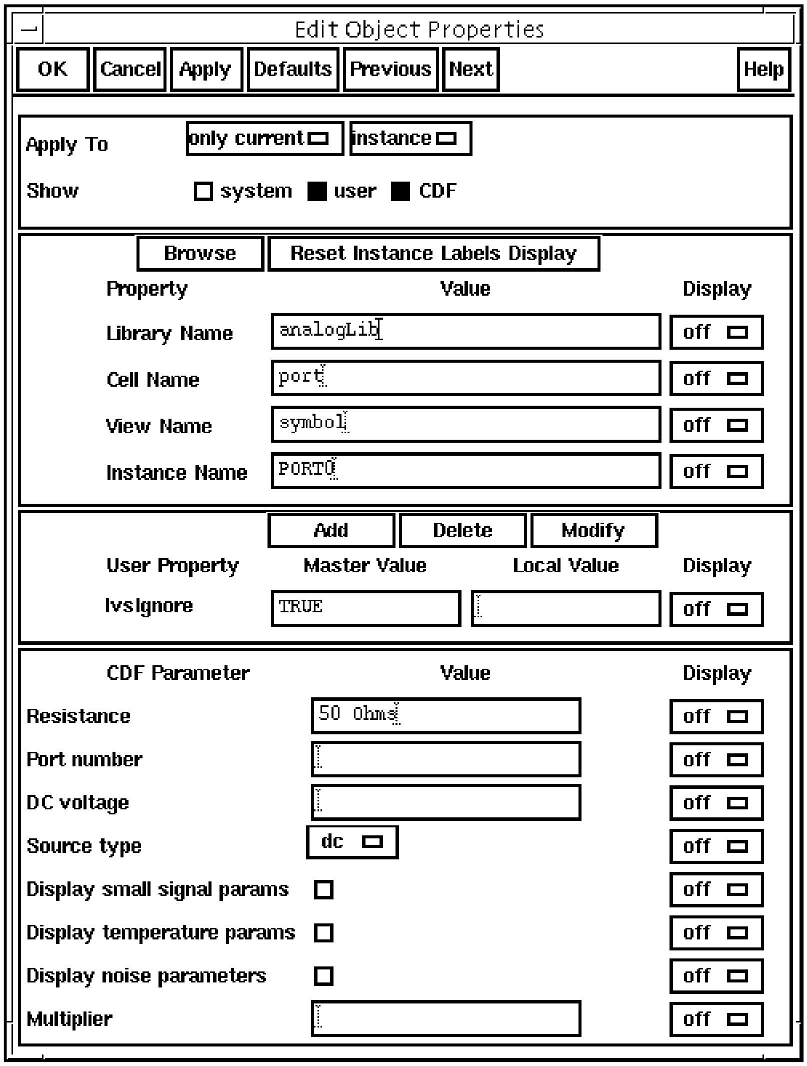
DC voltage

Sets the DC level of the source for DC analysis. The value must be a real number. If you do not specify the DC value, it is assumed to be the *time* =0 value of the waveform. Default: 0 Units: V

The *DC voltage* parameter specifies the DC voltage across the *port* when it is terminated in its reference resistance. In other words, the *DC voltage* of the internal voltage source is double the user specified DC value, *dc*. The same is true for the values for the *transient*, *AC*, and *PAC* signals of the `port`. However, you can alternatively specify the amplitude of the sine wave in the transient and PAC analyses as the power in dBm delivered by the *port* when it is terminated with the reference resistance.

Because all small signal analyses (AC, XF, and Noise) use DC analysis results, the *DC voltage* level also affects small-signal analyses. Transient analysis is not affected unless you specify *Source type=dc* or use *dc* as a default for the other waveform types.

Figure H-2 Source type=dc in the Edit Object Properties form



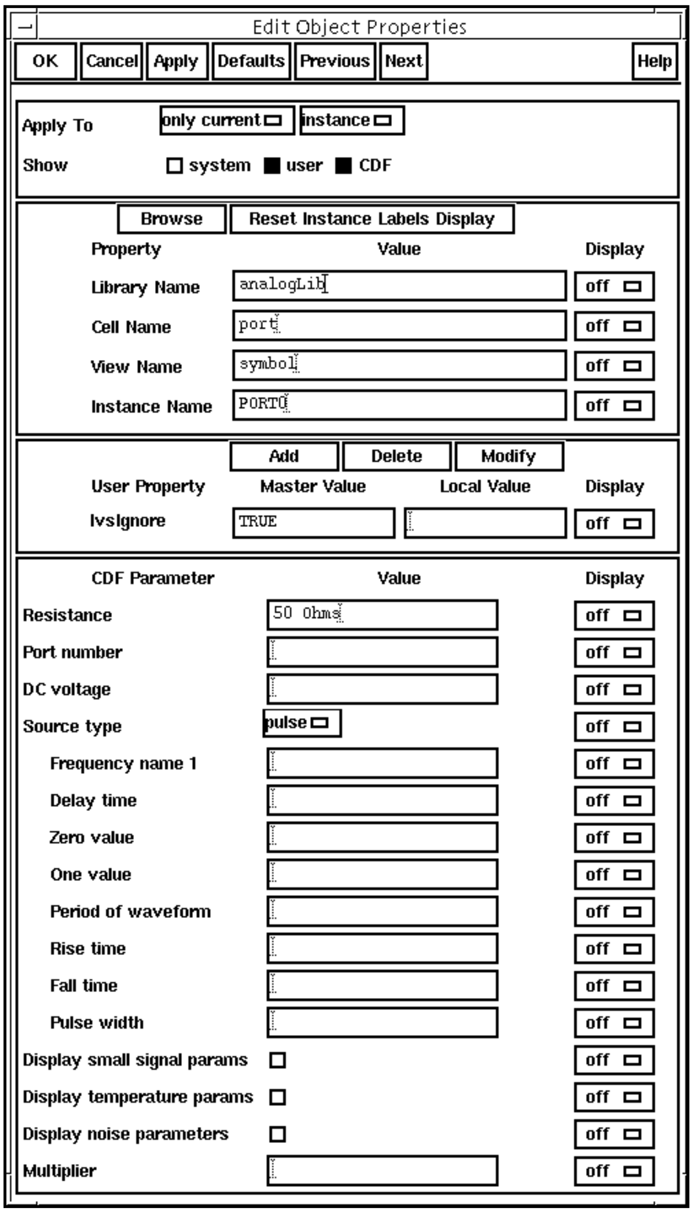
The *small signal, temperature,* and *noise* parameters are discussed in a later section of this application note.

Pulse Waveform Parameters

To generate a pulse waveform from the *port* component, set the CDF parameter *Source type=pulse* as shown below.

Remember, when you specify the voltage on a *port*, you are specifying the voltage *when the port is properly terminated*, and not the voltage on the internal voltage source. Therefore, the voltage on the internal source is set to twice the value specified on the *port*.

Figure H-3 Source type=pulse in the Edit Object Properties form



Zero value

The *Zero value* (*val0*) used in *pulse* and *exp* waveforms. Default: 0 Units: V

One value

The *One value* (*val1*) used in *pulse* and *exp* waveforms. Default: 1 Units: V

Period of waveform

The *period* of the *pulse* waveform. Default: *infinity*. Units: seconds

Rise time

The *Rise time* for the *pulse* waveform (time for transition from the *Zero value* to the *One value*). Units: seconds

Fall time

The *Fall time* for the *pulse* waveform (time for transition from the *One value* to the *Zero value*). Units: seconds

Pulse width

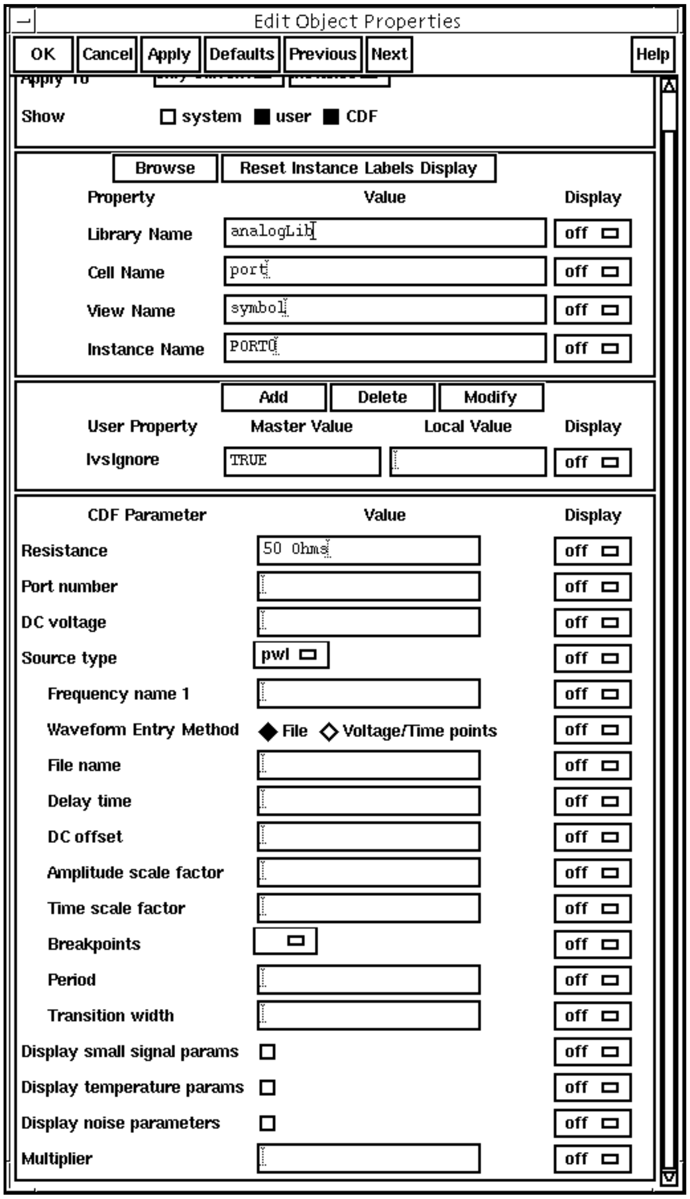
Pulse width (*width*, duration of *One value*). Default: *infinity*. Units: seconds

PWL Waveform Parameters

To generate a piecewise linear waveform from the *port* component, set the CDF parameter *Source type=pwl* as shown in the next figure.

Remember, when you specify the voltage on a *port*, you are specifying the voltage *when the port is properly terminated*, and not the voltage on the internal voltage source. Therefore, the voltage on the internal source is set to twice the value specified on the *port*.

Figure H-4 Source type=pwl in the Edit Object Properties form



Waveform Entry Method

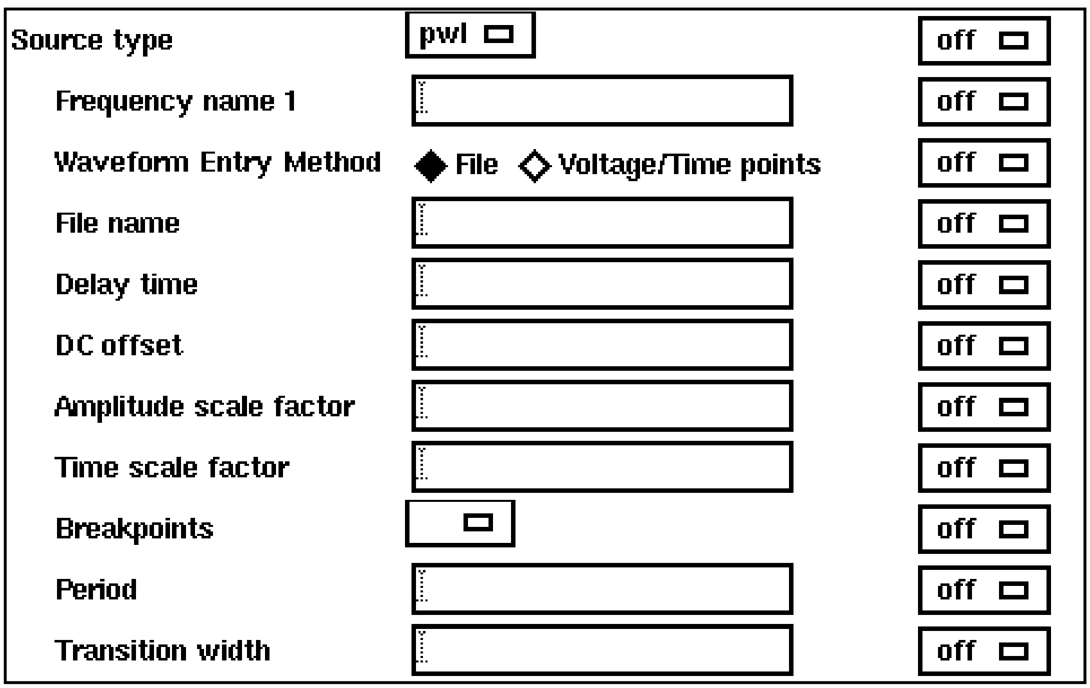
You can specify piecewise-linear data in two ways, either by specifying a *File name* or by entering a series of *Voltage/Time points*.

File name

If you select *Waveform Entry Method=File*, you enter the name of the file containing the piecewise-linear data in the form of time-value pairs. The value must be a string. Default: no value

In your file, list the time-value pairs as one pair per line with a space or tab between the time and voltage values. The numbers in the file must be given as simple numbers - SI scale factors (p, n, u, m, k, M, G, etc.) cannot be used.

Figure H-5 Waveform Entry Method = File

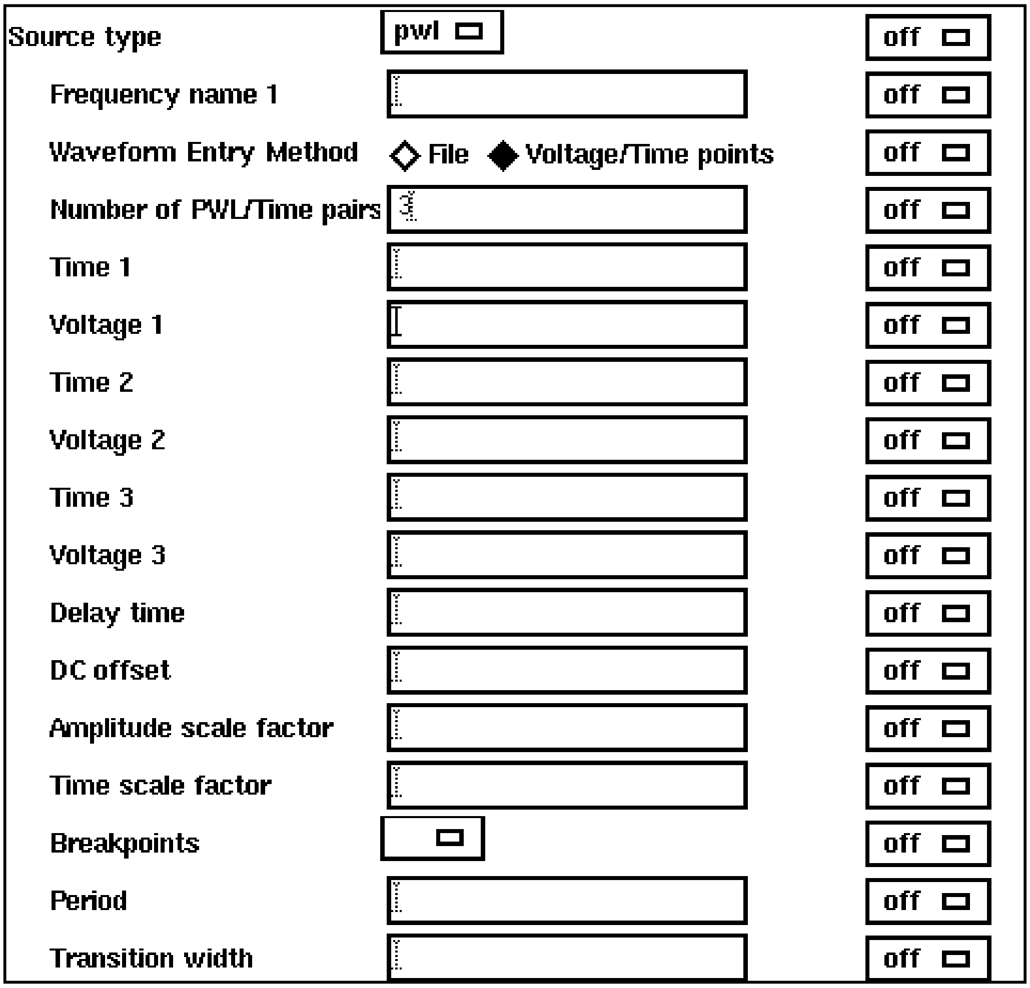


Number of PWL/Time pairs

If the *Waveform Entry Method=Voltage/Time points*, you specify the *Number of voltage-time pairs (tvpairs)*. The form expands to let you to type in the designated *Voltage* and *Time* values. Units: V and seconds. Default: 0 Maximum value: 50

In the following figure, the number of voltage-time pairs is set to 3.

Figure H-6 Waveform Entry Method = Voltage/Time points



The following *pwl* parameters: *DC offset, Amplitude scale factor,* and *Time scale factor* let you quickly adjust the amplitude, frequency and offset without editing each individual point in the *pwl* waveform.

DC offset

DC offset (*offset*) for the *pwl* waveform. Default: 0 Units: V

Amplitude scale factor

Amplitude scale factor (*scale*) for the *pwl* waveform. Default: 1

Time scale factor

Time scale factor (*stretch*) for the time given for the *pwl* waveform. Default: 1

Breakpoints

Possible values are *no*, *yes*, or blank. If you set *Breakpoints* (*allbrkpts*) to *yes*, you force Spectre to place time points at each point specified in a *pwl* waveform during a transient analysis. This can be very expensive for waveforms with many points. If you set *Breakpoints* to *no*, Spectre inspects the waveform, looking for abrupt changes, and forces time points only at those changes. If you set *Source type = pwl* and set *Breakpoints* to blank, the default is *yes* if the number of points you specify is less than 20.

Period

The *pwl* waveform is periodic if you specify *Period* (*pwlperiod*). Units: seconds

If the value of the waveform you specify is not exactly the same at both its beginning and its end, then you must provide a nonzero value for *Transition Width*.

Transition Width

Transition width (*twidth*) is used when making *pwl* waveforms periodic. Default: PWL period/1000. Units: seconds

Before repeating, the waveform changes linearly in an interval of *Transition Width* from its value at *(Period – Transition Width)* to its value at the beginning of the waveform. Thus the *Transition Width* must always be less than the *Period*.

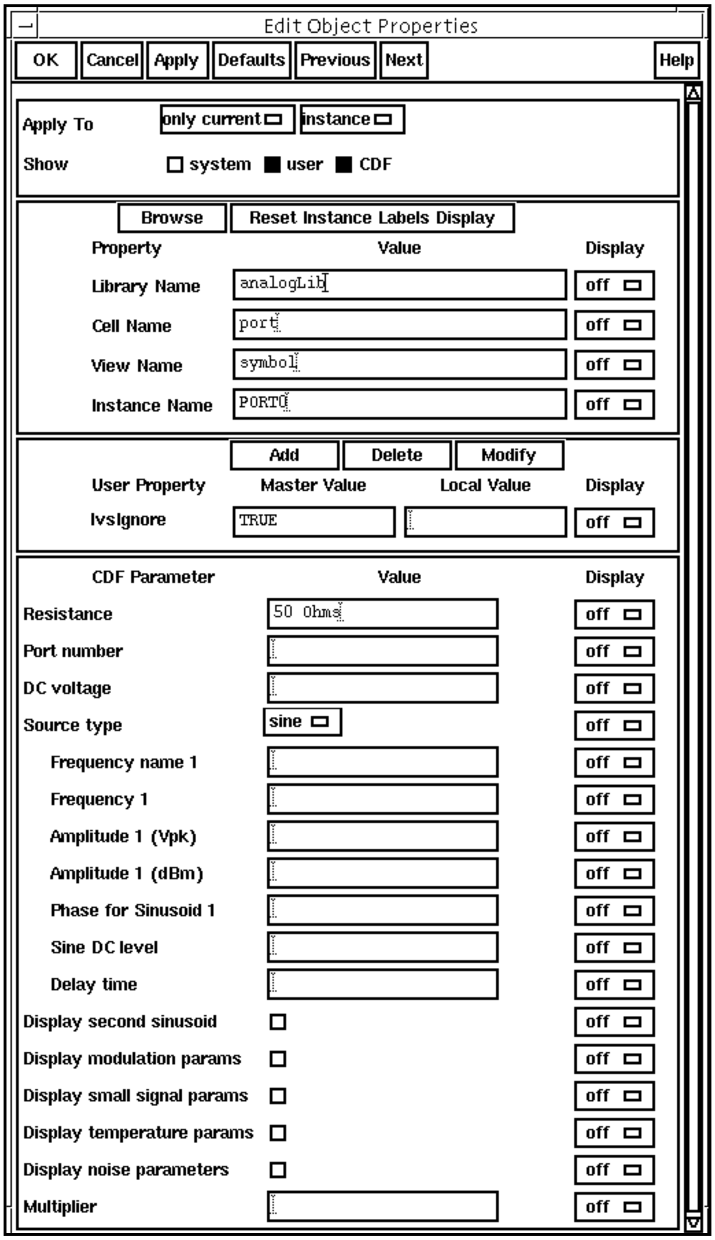
Sinusoidal Waveform Parameters

The *port* component can generate up to two sinusoids simultaneously. They are denoted as 1 and 2. You can set the amplitude, frequency and phase for both individually. The amplitude can be set to either a voltage or a power level. You can also specify sinusoidal AM or FM modulation of sinusoid 1.

The Edit Object Properties form for *Source type=sine* is shown below.

The first sinusoid is described by the parameters *Frequency name 1, Frequency 1, Amplitude 1 (Vpk), Amplitude 1 (dBm), Phase for sinusoid 1, Sine DC level, AM or FM modulation terms,* and *Damping factor 1.*

Figure H-7 Source type=sine in the Edit Object Properties form



Frequency name 1

Names the fundamental tones of sinusoid 1. After you save the schematic, the names you assign appear in the *Fundamental Tones* list box on the Choosing Analyses form.

Frequency 1

The frequency of the first sinusoidal waveform (carrier frequency). You typically use unmodulated signals in SpectreRF analyses. The value must be a real number. Default: 0 Units: Hz

Amplitude 1 (Vpk)

Peak amplitude of the first sinusoidal waveform that you generate. The value specified is the voltage delivered into a matched load. You can select either *Amplitude 1 (Vpk)* or *Amplitude 1 (dBm)*, but not both. If *Amplitude 1 (Vpk)* has a value, the *Amplitude 1 (dBm)* field is grayed out. The value must be a real number. Default: 1 Units: V

Remember, when you specify the voltage on a *port*, you are specifying the voltage *when the port is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Amplitude 1 (dBm)

Amplitude of the first sinusoidal waveform, in dBm. The value specified is the power delivered into a matched load. You can select either *Amplitude 1 (Vpk)* or *Amplitude 1 (dBm)*, but not both. If *Amplitude 1 (dBm)* has a value, the *Amplitude 1 (Vpk)* field is grayed out. The value must be a real number. Units: dBm

Phase for Sinusoid 1

The phase at the specified *Delay time*. The sinusoidal waveform might start before the given *Delay time* in order to achieve a specified phase and still remain continuous. For example, if you want to generate a cosine wave, set this parameter to 90^o. The value must be a real number. Default: 0 Units: degrees

Sine DC level

Sets the DC level for sinusoidal waveforms in transient analyses. This parameter is used when the sinusoid has a different average level than the one specified for the DC analyses. If not specified, the average value of the sinusoid is the same as that of the DC level of the source. The value must be a real number. Default: dc Units: V

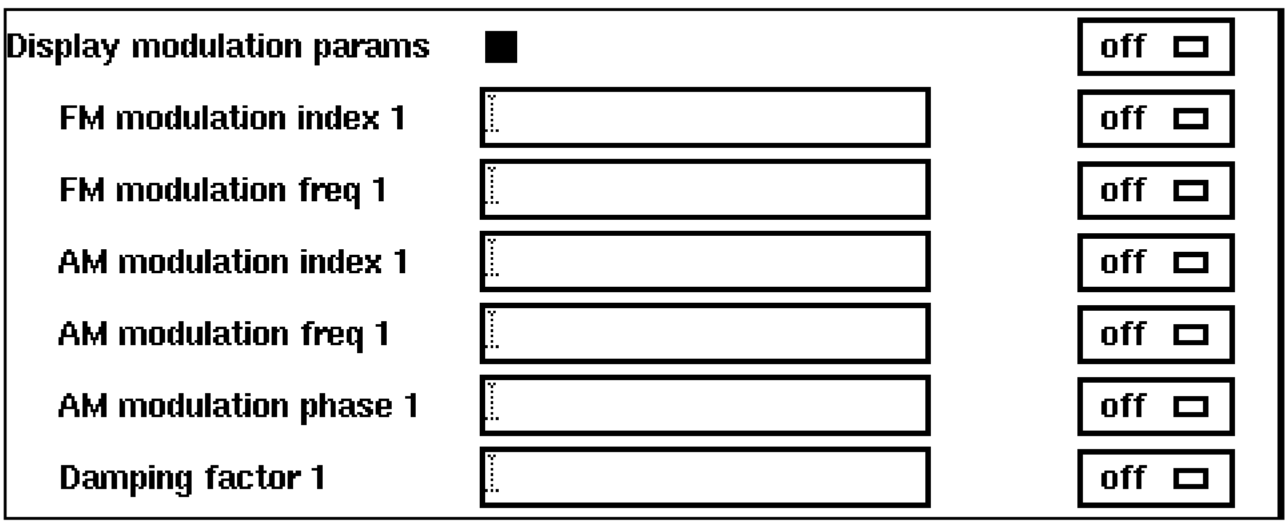
Modulation Parameters

Display Modulation Parameters

When selected, the form expands and following modulation parameters are displayed: *FM modulation index 1, FM modulation freq 1, AM modulation index 1, AM modulation freq 1, AM modulation phase 1,* and *Damping factor 1.*

Note: *Only* the first sinusoid can be modulated.

Figure H-8 Display modulation parameters in Edit Object Properties form



FM Modulation (Background Information)

The frequency modulation for the sinusoidal case is defined as

$$v_{FM}(t) = A\ \sin(2\pi f_c t + \beta \sin(2\pi f_m t) + \phi)$$

where

- A is the amplitude of sinusoid 1
- β is the FM modulation index
- $\sin(2\pi f_m t)$ is the modulation signal
- f_c is the carrier frequency

- ϕ is the *phase for sinusoid 1*

The frequency modulation parameters affect *only* the first sinusoid generated by *port*. They have no effect on the second sinusoid.

FM modulation frequency 1

FM modulation frequency for the sinusoidal waveform (f_m in the previous equation). The value must be a real number. Default: 0 Units: Hz

FM modulation index 1

FM index of modulation for the sinusoidal waveform, the ratio of peak frequency deviation divided by the center frequency (β in the above equations).

$$\beta = \Delta f / f_m$$

The value must be a real number. Default: 0

Effect of Amplitude Modulation (Background Information)

The amplitude modulation (double sideband suppressed carrier, or DSB-SC) is defined as

$$v_{AM}(t) = A\,(1 + m\,\sin(2\pi f_m t + \phi))\,\sin(2\pi f_c t)$$

where

- A is the carrier amplitude (amplitude of sinusoid 1)
- m is the AM modulation index
- f_m is the AM modulation frequency
- ϕ is the AM modulation phase
- $\sin(2\pi f_c t)$ is the carrier signal

The amplitude modulation parameters affect *only* the first sinusoid generated by *port*. They have no effect on the second sinusoid.

AM modulation frequency 1

AM modulation frequency for the first sinusoidal waveform (f_m in the previous equation). The value must be a real number. Default: 0 Units: Hz

AM modulation phase 1

AM phase of modulation for the first sinusoidal waveform (ϕ in the previous equation). The value must be a real number. Default: 0 Units: degrees

AM modulation index 1

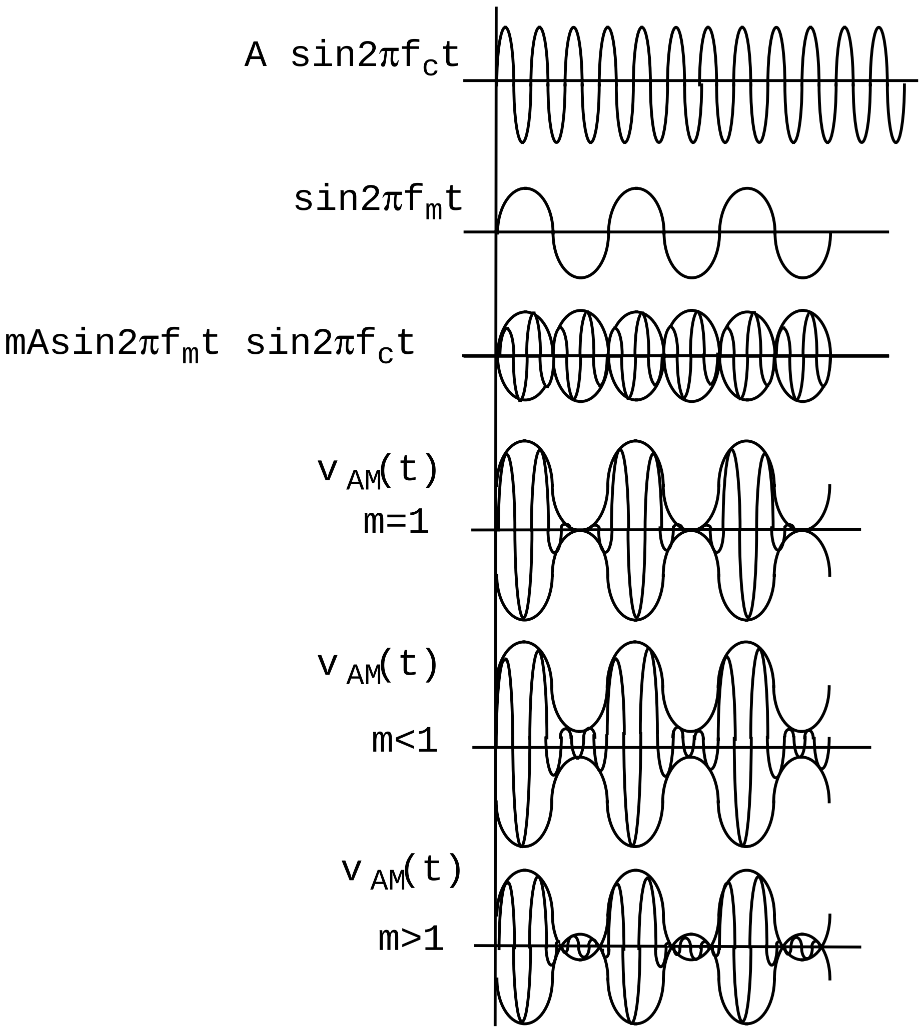
AM index of modulation for the first sinusoidal waveform (`m` in the previous AM equation). The *AM modulation index 1* is a dimensionless scale factor used to control the ratio of the sidebands to the carrier.

m = (peak DSB-SC amplitude)/(peak carrier amplitude)

The value must be a real number. Default: 0

The following figure shows the effect of varying modulation indexes for the following three cases: $m < 1$, $m = 1$, and $m > 1$. f_c is the carrier frequency, and f_m is the modulation frequency.

Figure H-9 Amplitude Modulation: Effects of Varying Modulation Indexes



Damping factor 1

Damping factor for the sinusoidal waveform. *Damping factor 1*specifies the time it takes to go from the envelope (full amplitude) at *time*=0 to 63 percent of the full amplitude. For example, consider the following damped sinusoid:

$$v(t) = A\ e^{-\sigma t}\ \sin(2\pi ft + \phi)$$

where

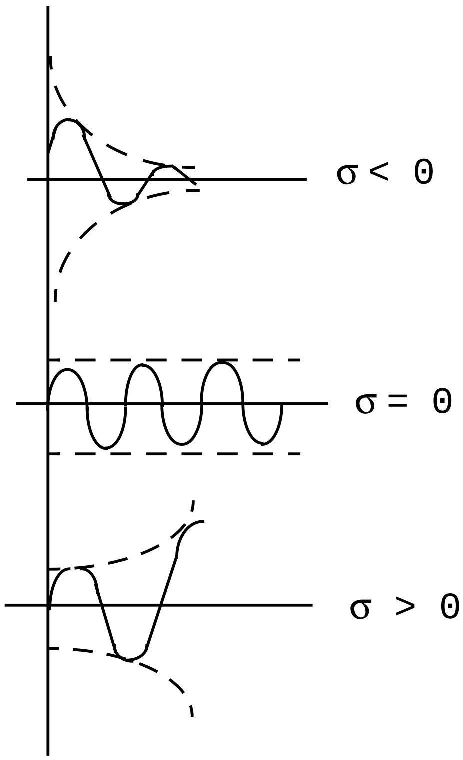
σ = *Damping factor 1*, A is the amplitude of sinusoid 1, and ϕ is the *Phase for sinusoid 1*

- If $\sigma = 0$, the waveform is a pure sinusoid (steady state).
- If $\sigma < 0$, the waveform exhibits decaying oscillations.
- If $\sigma > 0$, the waveform exhibits growing oscillations.

It takes 5σ to diminish to 1 percent of the peak amplitude. The value must be a real number. Default: 0. Units: 1/seconds

The following figure shows the effect of *Damping factor 1* on the first sinusoid for three values of σ.

Figure H-10 Effect of Damping Factor 1 on the First Sinusoid

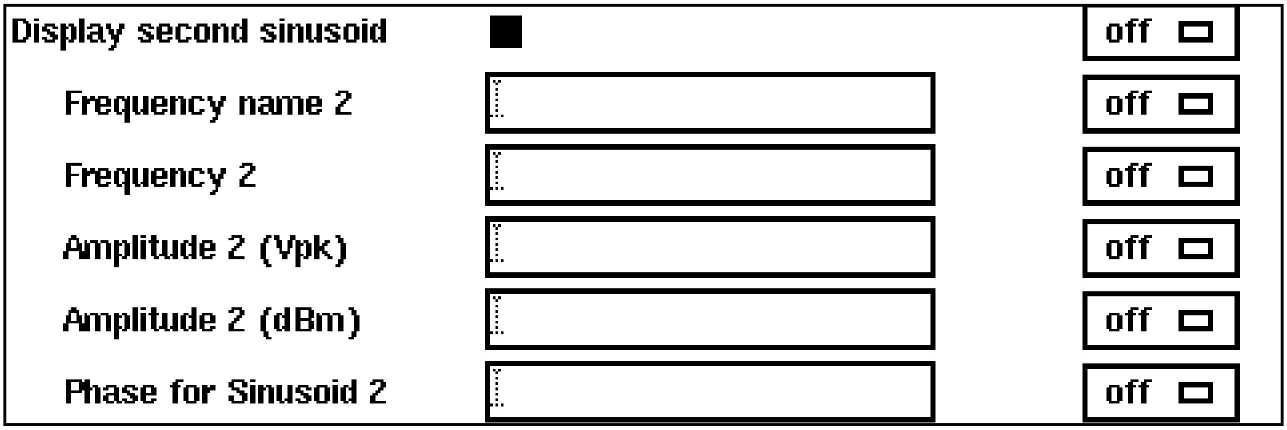


Display second sinusoid

Displays the CDF parameters for the second sinusoid in the Edit Object Properties and Add Instance forms. When selected, the form expands to show the following CDF parameters: *Frequency name 2, Frequency 2, Amplitude 2 (Vpk), Amplitude 2 (dBm),* and *Phase for Sinusoid 2.*

Note: The second sinusoid cannot be modulated.

Figure H-11 Display second sinusoid in the Edit Object Properties form



Frequency name 2

Name for the second sinusoid. After you save the schematic, the name you assign appears in the Fundamental Tones list box on the Choosing Analyses form.

Frequency 2

Frequency of the second sinusoidal waveform. The value must be a real number. Default: 0 Units: Hz

Amplitude 2 (Vpk)

Peak amplitude of the second sinusoidal waveform. The value specified is the voltage delivered into a matched load. You can select either *Amplitude 2 (Vpk)* or *Amplitude 2 (dBm)*, but not both. If *Amplitude 2 (Vpk)* has a value, the *Amplitude 2 (dBm)* field is grayed out. The value must be a real number. Default: 1 Units: V

Remember, when you specify the voltage on a *port*, you are specifying the voltage *when the port is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Amplitude 2 (dBm)

Amplitude of the second sinusoidal waveform, in dBm. The value specified is the power delivered into a matched load. You can select either *Amplitude 2 (Vpk)* or *Amplitude 2* *(dBm),* but not both. If *Amplitude 2 (dBm)* has a value, the *Amplitude 2 (Vpk)* field is grayed out. The value must be a real number. Units: dBm

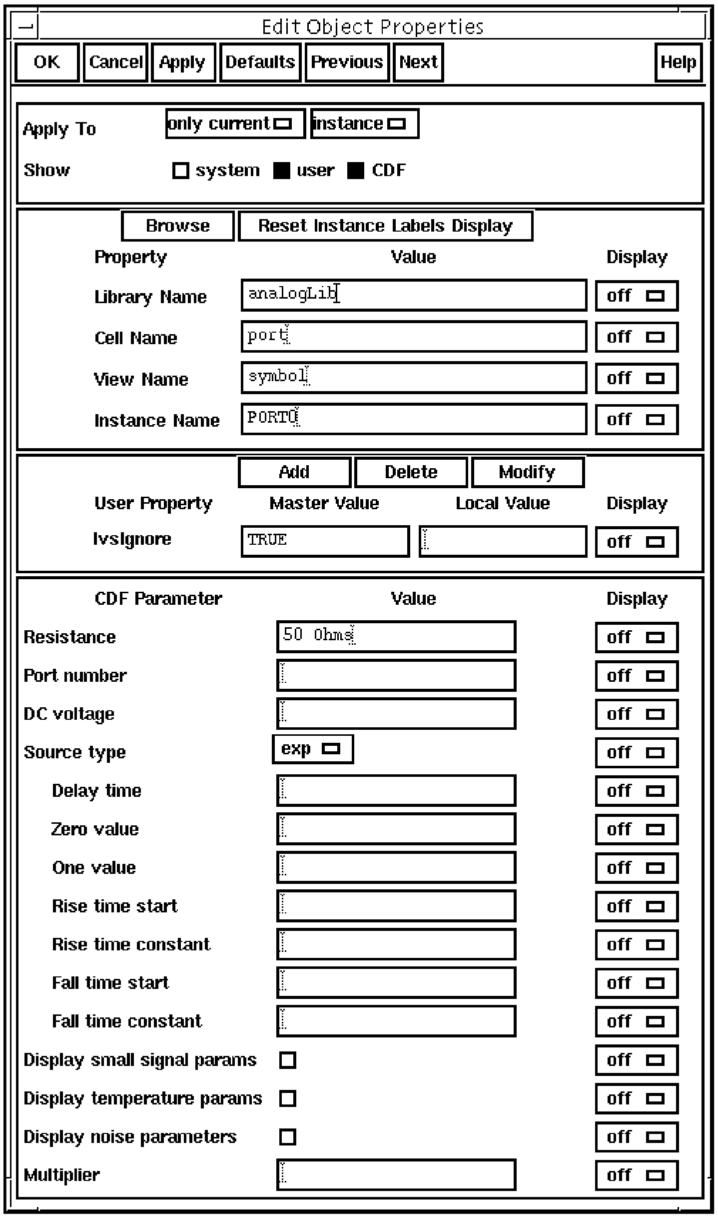
Phase for Sinusoid 2

The phase at the specified *Delay time* for the second sinusoid. The sinusoidal waveform might start before the given *Delay time* in order to achieve specified phase while still remaining continuous. The value must be a real number. Default: 0 Units: degrees

Exponential Waveform Parameters

To generate an exponential waveforms from the `port` component, set the CDF parameter *Source type=exp,* as shown in the next figure.

Figure H-12 *Source type*=*exp* **in Edit Object Properties form**



Remember, when you specify the voltage on a *port*, you are specifying the voltage *when the port is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*.

Zero value

The *Zero value* (*val0*) used in *pulse* and *exp* waveforms. Default: 0 Units: V

One value

The *One value* (*val1*) used in *pulse* and *exp* waveforms. Default: 1 Units: V

Rise time start

Rise start time (*td1*) for the *exp* waveform. Default: 0 Units: seconds

Rise time constant

Rise time constant (*tau1*) for the *exp* waveform. Units: seconds

Fall time start

Fall start time (*td2*) for the *exp* waveform. Units: seconds

Fall time constant

Fall time constant (*tau2*) for the *exp* waveform. Units: seconds

Noise Parameters

Noise parameters include *Noise temperature, Noise Entry Method, Noise file name, and Number of Noise Frequency Pairs.*

Noise temperature

The *Noise temperature* of the *port*. If not specified, the *Noise temperature* is assumed to be the actual temperature of the *port*. When you compute the noise figure of a circuit driven at its input by a *port*, set the *Noise temperature* of the *port* (Spectre parameter *noisetemp*) to 16.85C (290K). This setting matches the standard IEEE definition of noise figure. In addition, disable all other sources of noise in the *port*, such as the Spectre parameters *noisefile* and *noisevec.* If you want a noiseless *port*, set the *Noise temperature* to absolute zero or below, and do not specify a noise file or noise vector. Default: Actual temperature of the port. Units: ^{o}C

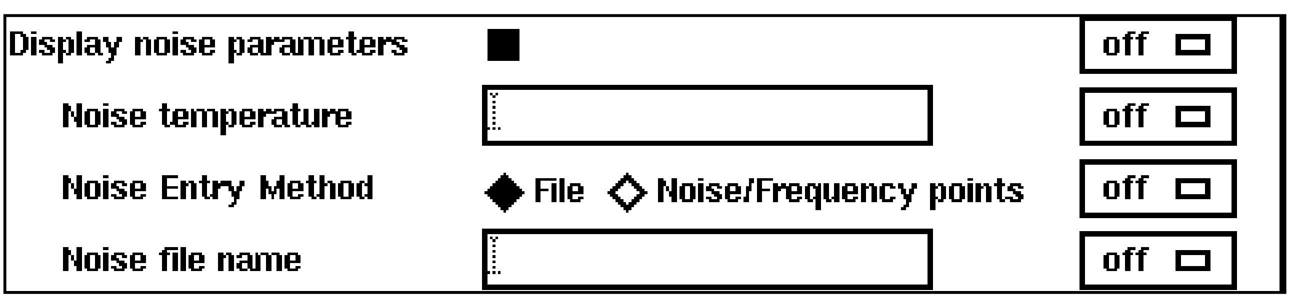
Noise Entry Method

You can select one of two ways to enter noise data, either by specifying a *Noise file name* or entering a series of *Noise/Frequency points*.

Noise file name

If *Noise Entry Method = File* is selected, you enter the name of the file containing the excess spot noise data in the form of frequency-noise pairs. In your file, list the frequency-noise pairs as one pair per line with a space or tab between the frequency and noise values. The value must be a string. Default: none

Figure H-13 Display noise parameters: Noise Entry Method=File

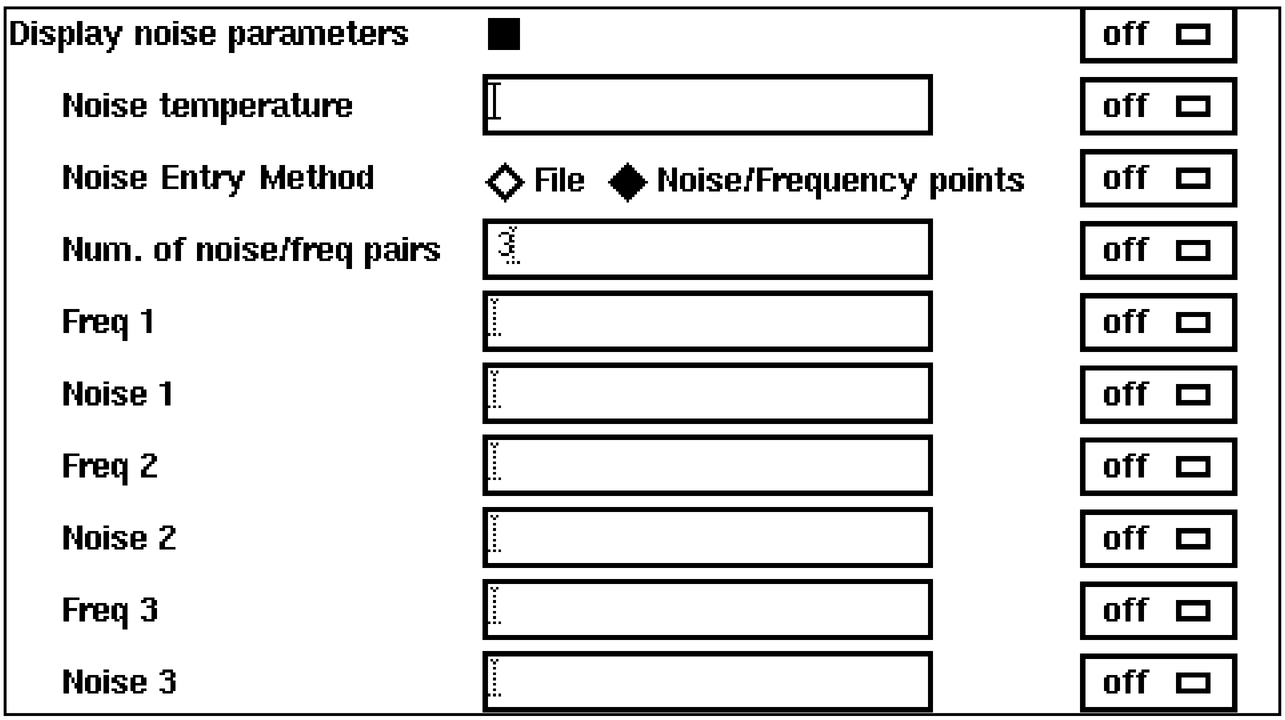


Num. of noise/freq pairs

If the *Noise Entry Method=Noise/Frequency points*, you specify the *Number of noise/ freq pairs*. The form expands to let you type in the designated *Freq* and *Noise* values. The noise values must be in V^2/Hz, and frequency in Hz. Default: 0 Maximum value: 10

The example in the next figure has the *Number of noise/freq pairs* set to 3.

Figure H-14 Noise parameters: Noise Entry Method=Noise/Freq Points

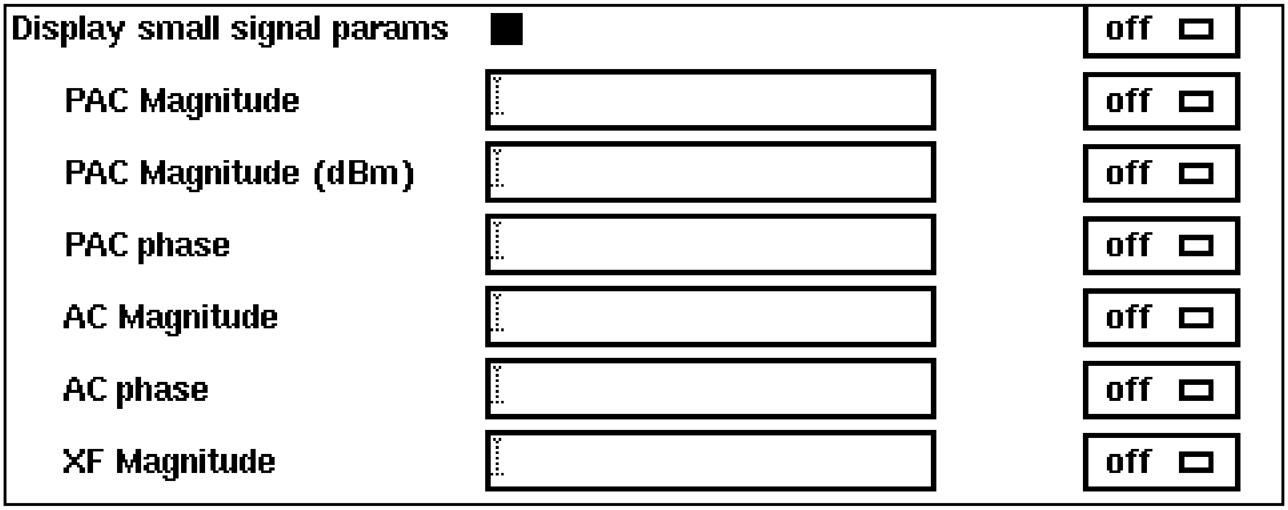


Small-Signal Parameters

Display small signal params

If selected, the Edit Object Properties/Add Instance form expands to show the small-signal parameters *PAC magnitude, PAC magnitude (dBm), PAC phase, AC magnitude, AC phase,* and *XF magnitude.*

Figure H-15 Display small signal parameters selected



Remember, when you specify the voltage on a *port*, you are specifying the voltage *when the port is properly terminated*, and not the voltage on the internal voltage source. Thus, the voltage on the internal source is set to twice the value specified on the *port*. The same is true for the values for the transient, AC, and PAC signals. However, the amplitude of the sine wave in the PAC and transient analysis can alternatively be specified as the power in dBm delivered by the `port` *when terminated with the reference resistance.*

PAC Magnitude

The peak periodic AC analysis magnitude. Setting this value to unity is a convenient way of computing the transfer function from this source to the output.

You can select either *PAC magnitude* or *PAC magnitude (dBm)*, but not both. If *PAC magnitude* has a value, the *PAC magnitude (dBm)* field is grayed out. The value must be a real number. Default: 0 Units V

PAC Magnitude (dBm)

The periodic AC analysis magnitude in dBm (alternative to *PAC magnitude*). You can select either *PAC magnitude* or *PAC magnitude (dBm)*, but not both. If *PAC magnitude (dBm)* has a value, the *PAC magnitude* field is grayed out. The value must be a real number. Units: dBm

PAC phase

The periodic AC analysis phase. The value must be a real number. Default: 0 Units: degrees

Typically, only one source in the circuit has a *PAC magnitude* set to a value other than zero, and usually it has a *PAC magnitude*=1 and *PAC phase*=0. However, there are situations where more than one source has a nonzero *PAC magnitude*. For example, applying a differential small-signal input could be done with two sources with the *PAC magnitude*s set to 0.5 and the *PAC phase*s set to 0 and 180.

You do not specify the PAC frequency in the *port* Edit Object Properties form. Instead, you set the PAC frequency in the PAC *Choosing Analysis* form. For example, when making an IP3 measurement, you set the PAC frequency to a variable value in the Choosing Analysis Form. Then, you enter the same variable in the *PAC Amplitude* (or *PAC Amplitude dBm*) field of the *port* Edit Object Properties form.

AC Magnitude

The peak small-signal voltage. The value must be a real number. Default: 0 Units: V

AC phase

The small-signal phase. The value must be a real number. Default: 0 Units: degrees

Typically, only one source in the circuit has *AC Magnitude* set to a value other than zero, and usually it has an *AC magnitude*=1 and *AC phase*=0. However, there are situations where more than one source has a nonzero *AC magnitude*. For example, you can apply a differential small-signal input with two sources with the *AC magnitude*s set to 0.5 and the *AC phase*s set to 0 and 180.

XF Magnitude

The transfer function analysis magnitude. Use *XF magnitude* to compensate for gain or loss in the test fixture. The value must be a real number. Default: 1 Units: V/V

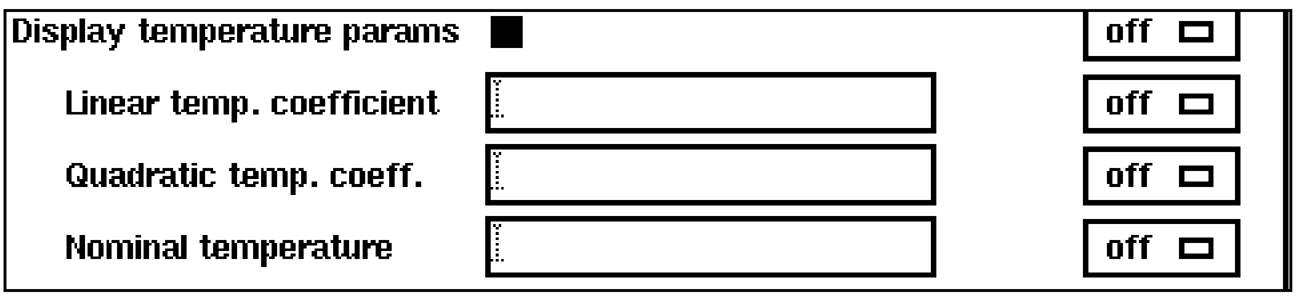
Temperature Effect Parameters

Temperature effect parameters include the *Linear temperature coefficient*, *Quadratic temperature coefficient*, and *Nominal temperature*.

Display temperature params

If selected, the Edit Object Properties/Add Instance form expands and the following three parameters appear in the form: *Linear temp. coefficient, Quadratic temp. coeff.,* and *Nominal temperature.*

Figure H-16 Display temperature parameters selected



Linear temp. coefficient

First order (linear) temperature coefficient of the *DC voltage* (*tc1*). The value must be a real number. Default: 0 Units: $^{o}C^{-1}$

Quadratic temp. coeff.

Second order (quadratic) temperature coefficient of the *DC voltage* (*tc2*). The value must be a real number. Default: 0 Units: $^{o}C^{-2}$

Nominal temperature

The *Nominal temperature* for *DC voltage* (*tnom*). The value must be a real number. Default: Set by options specifications. Units: ^{o}C

How Temperature Parameters Affect the Voltage Level (Background Information)

The value of the *DC voltage* can vary as a function of the temperature if you specify *tc1* and *tc2*. The variation is given by

$V_{DC}(T) = dc * [1 + tc1 * (T - tnom) + tc2 * (T - tnom)^2]$

where *T* is the analysis temperature specified in the analysis options, *tnom* is the *Nominal temperature* specified in the *Choosing Analyses* form, and dc is the *DC voltage*.

If the analysis temperature equals the nominal temperature, the result is the voltage amplitude that you specified, V(*T*)=dc.

If the nominal and analysis temperatures differ, the voltage amplitude is given by

$V_{DC}(T) = dc * [1 + tc1 * (T - tnom) + tc2 * (T - tnom)^2]$

where *T* is the analysis temperature you specify in the analysis options and tnom is the nominal temperature, *tc1* and *tc2* are the *Linear* and *Quadratic temperature coefficients*, and *dc* is the *DC voltage.*

For example, if the *Nominal temperature* is 27^{o}C and the analysis temperature is 25^{o}C, there is a 2^o difference between the nominal and analysis temperature. The voltage amplitude is

$V_{DC}(T) = dc * [1 + tc1 * (-2) + tc2 * (-2)^2]$

Additional Notes

Active Parameters in Analyses

In DC analyses, the only active parameters are *dc, m*, and the temperature coefficient parameters.

In AC analyses, the only active parameters are *m, mag,* and *phase*.

In transient analyses, all parameters are active except the small-signal parameters and the noise parameters.

In PAC, the only active parameters are *m*, *PAC magnitude* (amplitude or dBm), and *PAC phase*.

XF magnitude is active in XF and PXF analyses only.

I

Analyzing Time-Varying Noise

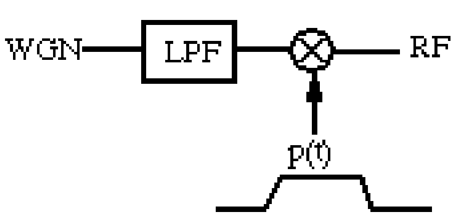
RF circuits are usually driven by periodic inputs. The noise in RF circuits is generated by sources that can therefore typically be modelled as periodically time-varying. Noise that has periodically time-varying properties is said to be cyclostationary. In order to allow more detailed characterization of cyclostationary noise in RF circuits, a *NoiseType* parameter with three possible values exists in the *pnoise* analysis.

- Selecting *sources* from the *NoiseType* cyclic field computes the total time-average noise at an output over a given frequency range. Each noise source's contribution to the noise is computed at each frequency. The *sources* option is the default and represents the functionality present in the pnoise analysis in previous releases of SpectreRF.
- Selecting *timedomain* from the *NoiseType* cyclic field computes the time-varying instantaneous noise power in a circuit with periodically driven components.
- Selecting *correlations* from the *NoiseType* cyclic field correlates the noise at different ports of a multi-port circuit. For example, it computes the correlation of noise at different outputs, and the correlation of noise at the input and output of a circuit that exhibits frequency conversion. You can extract equivalent noise sources from these calculations.

Characterizing Time-Domain Noise

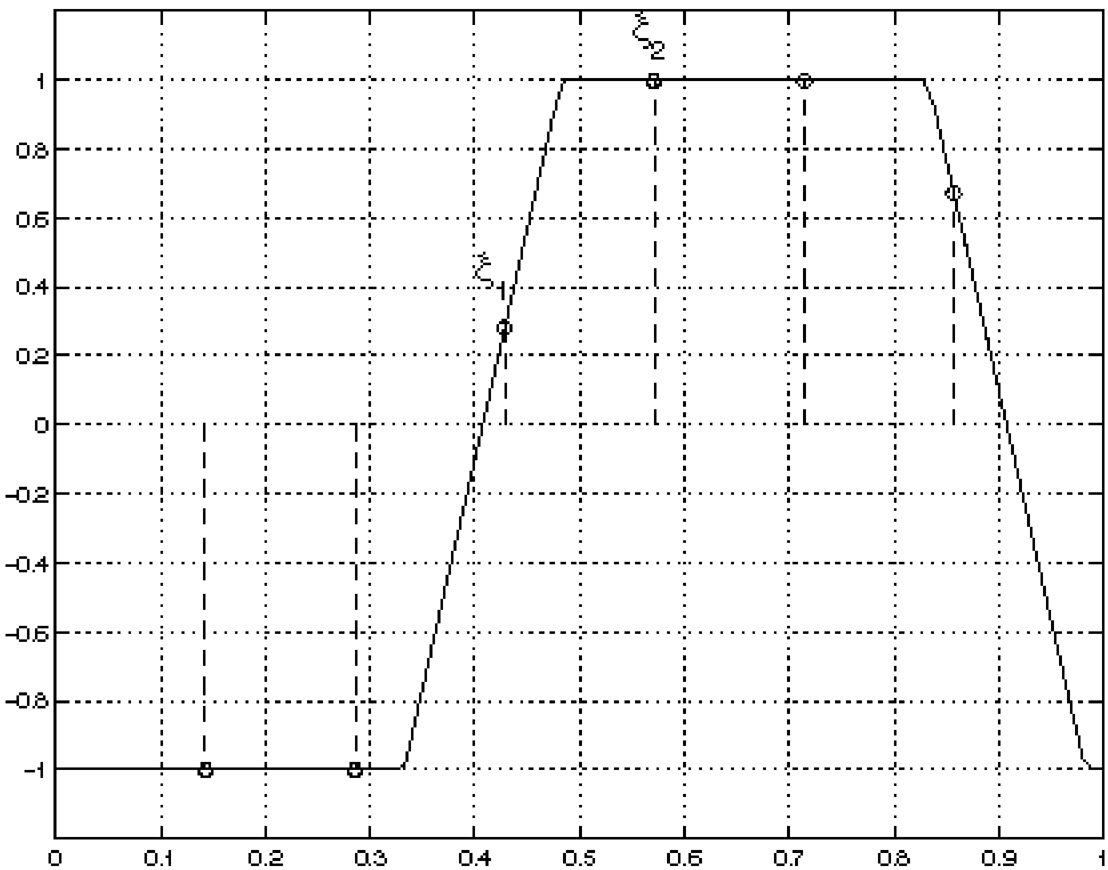
Noise in a circuit that is periodically driven, say with period T, exhibits statistical properties that also vary periodically.To understand time-domain characterization of noise, consider the simple circuit shown in Figure I-1.

Figure I-1 Very Simple Mixer Schematic



The amplitude of the noise measured at the RF output shown in Figure I-1 periodically varies depending on the magnitude of the modulating signal *p(t)*, as shown by the sample points in Figure I-2.

Figure I-2 Time-Varying Noise Process Analyzed at ξ_1 and ξ_2



In Figure I-2

- The solid line shows the envelope *p(t)* that modulates the noise process.
- The circles show possible phase points on the envelope where you might calculate the time-varying noise power.
- The circles marked ξ_1 and ξ_2 indicate the two phase points on the envelope where time-varying noise power is calculated.

Noise in circuits that are periodically driven, say with period *T*, exhibits statistical properties that also vary periodically. To understand time-domain characterization of noise, consider the simple circuit shown in Figure I-1 on page 1333. The amplitude of the noise measured at the

RF output periodically varies depending on the magnitude of the modulating signal $p(t)$, as shown by the sample points (or circles on the signal envelope) in Figure I-2 on page 1334.

Figure I-2 is a representation of periodically-modulated noise. It shows noise processes for two different phases in the periodic interval. Each process is stationary.

SpectreRF can calculate the time-varying noise power at any point in the fundamental period. In fact, SpectreRF can calculate the full auto correlation function

$$R^{\xi}(p,q) = < x^{\xi}(p)x^{\xi}(p+q) > = R^{\xi}(q)$$

and its spectrum for the discrete-time processes x^{ξ} obtained by periodically sampling the time-domain noise process at the same point in phase.

Figures I-3 and I-4show two such noise processes for two different phases in the periodic interval. Each process is stationary. Figure I-3 shows the noise process for the phase marked ξ_1 in Figure I-2 on page 1334.

Figure I-3 Noise Process for Phase ξ_1

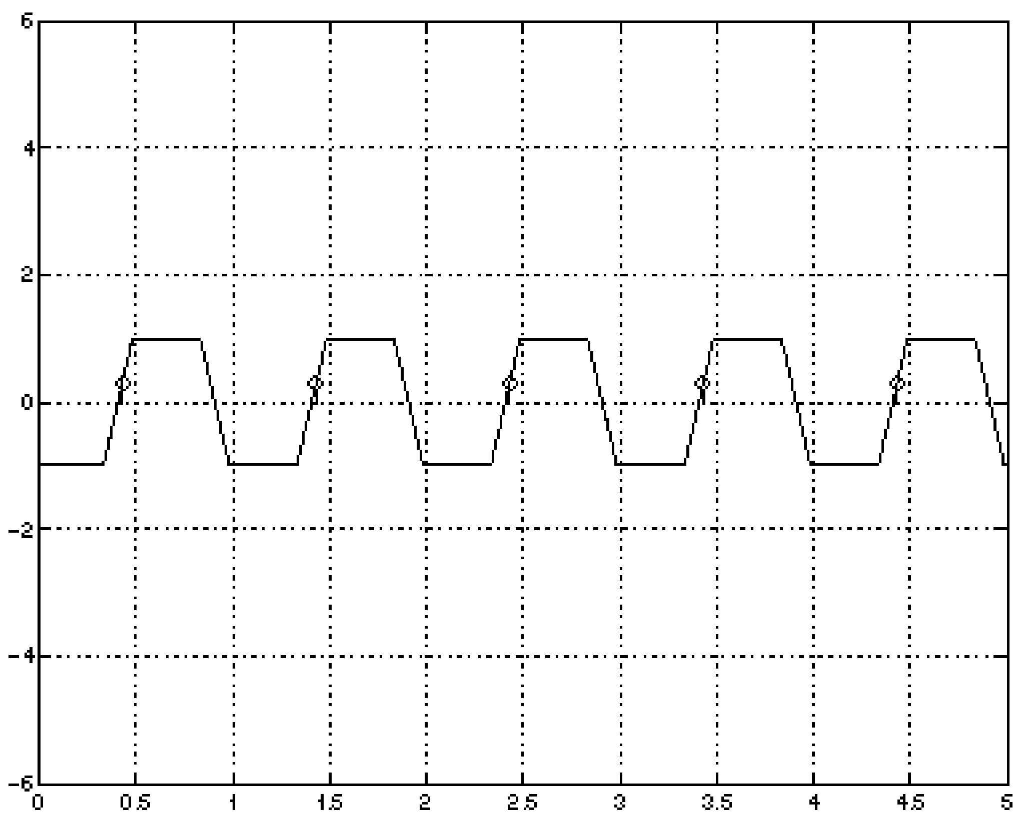
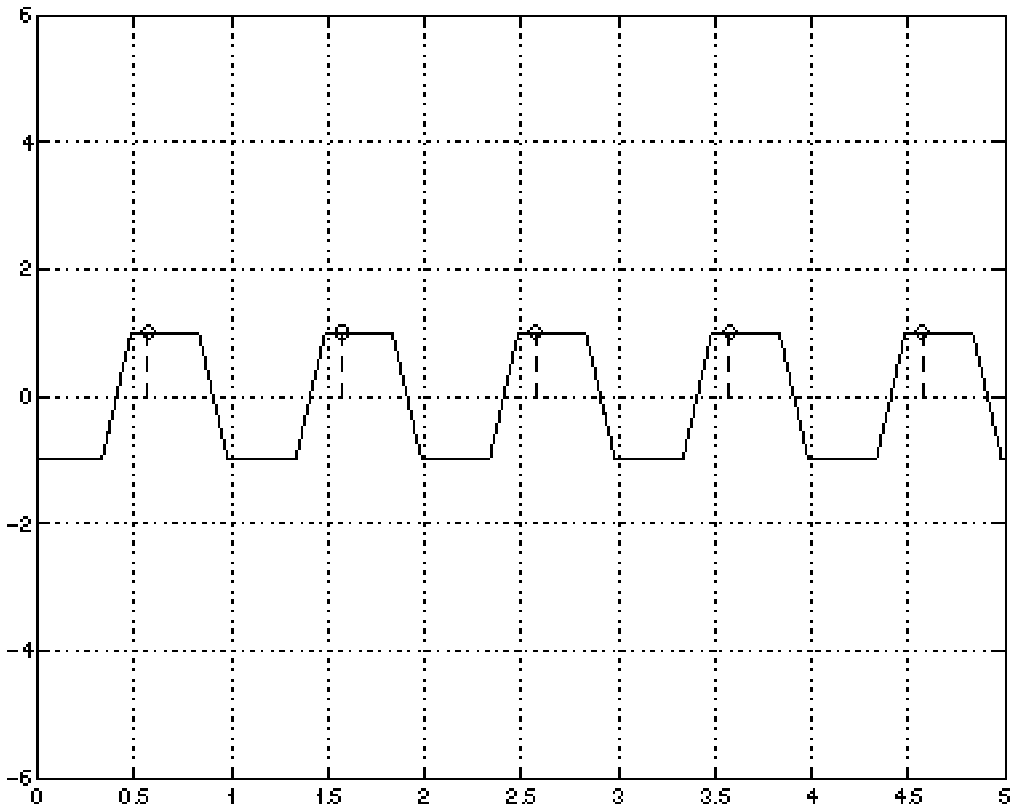


Figure I-4 shows the noise process for the phase marked ξ_2 in Figure I-2 on page 1334.

Figure I-4 Noise Process for Phase ξ_2



See the "Reference Information on Time-Varying Noise" on page 1350 for a more detailed introduction to noise in periodically time-varying systems.

Calculating Time Domain Noise

The following steps tell you how to calculate time-domain noise using SpectreRF.

1. In a terminal window, type `icms` to start the environment.
2. In the Simulation window, select *Analyses – Choose.*

   The Choosing Analyses form appears.
3. In the Choosing Analyses form, highlight pss and perform the PSS analysis setup.
4. In the Choosing Analyses form, highlight *pnoise.*

The Choosing Analyses form changes to let you specify information for a Pnoise analysis.

5. In the Choosing Analyses form, perform the following:

a. Choose Noise Type *timedomain*.

b. Specify an appropriate frequency range and sweep for the analysis.

You might, for example, perform a linear sweep up to the fundamental frequency. Because each time point in the calculation is a separate frequency sweep, use the minimum number of frequency points possible to resolve the spectrum. This step minimizes computation time.

c. Specify a *noiseskipcount* value or specify additional explicit time points with *noisetimepoints.*

d. Specify an appropriate set of time points for the time-domain noise analysis.

Use *noiseskipcount* to calculate time-domain noise for one of every *noiseskipcount* time points.

If you set *noiseskipcount* to a value greater than or equal to zero, the simulator uses the *noiseskipcount* parameter value and ignores any *numberofpoints* parameter value. When *noiseskipcount* is less than zero, the simulator ignores the *noiseskipcount* parameter. The default is *noiseskipcount*=-1*.*

You can add specific points by specifying a time relative to the start of the PSS simulation interval. *noiseskipcount=5* performs noise calculations for about 30 time points in the PSS interval.

If you only need a few time points, add them explicitly with the *noisetimepoints* parameter and set *noiseskipcount* to a large value like 1000.

6. In the Simulation window, choose *Simulation – Netlist and Run.*

The simulation runs.

7. In the Simulation window, choose *Results – Direct Plot – PSS*.

The PSS Results form appears.

8. To calculate time-varying noise power, perform the following steps in the PSS Results form:

a. Click on *tdnoise* and then select *Integrated noise power*.

b. Type `0` as the start frequency and the PSS fundamental frequency as the stop period.

For example, type `1G` if the PSS period is 1ns.

A periodic waveform appears that represents the expected noise power at each point in the fundamental period.

9. To display the spectrum of the sampled processes, perform the following steps in the PSS Results form:

   a. Highlight *Output Noise*.

   b. Highlight *Spectrum* for the type of sweep.

   c. Clicking on Plot.

   A set of curves appears, one for each sample phase in the fundamental period.

10. To calculate the autocorrelation function for one of the sampled processes, perform the following steps:

    a. Display the spectrum using instructions from the previous step.

    b. In the Simulation window, choose *Tools – Calculator*.

       The calculator appears.

    c. Click on *wave* in the calculator and the select the appropriate frequency-domain spectrum.

       One of the sample waveforms is brought into the calculator

    d. Choose DFT from the list of special functions in the calculator. Then set 0 as the *From* and the PSS fundamental as the *To* value.

    e. Choose an appropriate window (e.g., Cosine2) and number of samples (around the number of frequency points in the interval [0,1/T]),

    f. Apply the DFT and plot the results.

    Harmonic q of the DFT results gives the value of the discrete autocorrelation for this sample phase, $R(q)$.

    Note: Be sure the noise is in the correct units of power (e.g., V^2/Hz), not V/square root of Hz) before performing the DFT to obtain the autocorrelation.

Calculating Noise Correlation Coefficients

To characterize the noise in multi-input/multi-output systems, it is necessary to calculate both the noise power at each port and the correlation between the noise at various ports. The situation is complicated in RF systems because the ports may be at different frequencies. For example, in a mixer, the input port may be at the RF frequency and the output port at the IF frequency.

Denote the power spectrum of a signal x by $S_{XX}(\omega)$, that is

$$S_{XX}(\omega) = X^*(\omega)\, X(\omega)$$

where $X(\omega)$ is the Fourier transform of the signal $x(t)$. For random signals like noise, calculate the expected value of the power spectrum $S_{XX}(\omega)$. To characterize the relationship between two separate signals $x(t)$ and $y(t)$, you also need the cross-power spectrum

$$S_{XY}(\omega) = X^*(\omega)\, Y(\omega)$$

For random signals, the degree to which x and y are related is given by the cross-power spectrum. You can define a correlation coefficient $\rho xy(\omega)$ by

$$\rho XY(\omega) = \frac{S_{XY}(\omega)}{\sqrt{S_{XX}(\omega) S_{YY}(\omega)}}$$

- A correlation coefficient ($\rho xy(\omega)$) of `0` indicates the signals are completely uncorrelated.
- A correlation coefficient of `1` indicates the signals are perfectly correlated. For example, a signal is always perfectly correlated with itself.

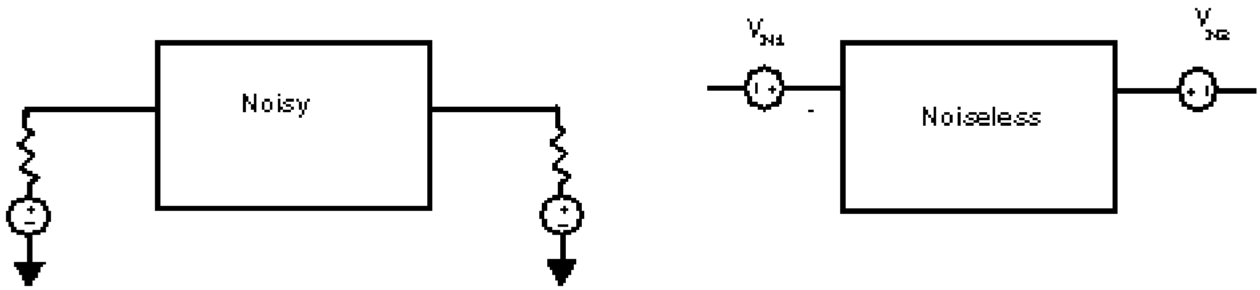
You might also want to consider correlations between noise at different frequencies. The following quantity

$$S^{\alpha}_{XY}(\omega)$$

expresses the correlation of a signal x at frequency ω with the signal y at frequency $\omega + \alpha$. For example, white Gaussian noise is completely uncorrelated with itself for $a \neq 0$. Noise in an RF system generally has $S^{\alpha}(\omega)$ non-zero when α is the fundamental frequency, for example, the LO frequency in a mixer.

Once you have measured the noise properties of a circuit, you can represent the circuit as a noiseless multiport with equivalent noise sources. For example, in Figure I-5 on page 1340, first you measure the noise voltage appearing at the excitation ports of the circuit on the left in the figure. Then, you can express the noise properties of the circuit as two equivalent frequency-dependent noise voltages V_{N1} and V_{N2}, and a complex correlation coefficient $\rho 12$.

Figure I-5 Calculating Noise Correlations and Equivalent Noise Parameters



When you know the noise at each port and its correlation, you can obtain any of various sets of equivalent noise parameters. For example, you can express noise in an impedance representation as the equivalent correlated noise voltage sources shown in Figure I-5 on page 1340, as equivalent noise resistances and the correlation parameters, and as F_{min}, R_N, G_{opt}, and B_{opt}.

Calculating Noise Correlation Parameters for a Two-Port Circuit

The following steps describe how to calculate noise correlation parameters for a two-port circuit.

1. In a terminal window, type `icms` to start the environment.
2. In the Simulation window, select *Analyses – Choose.*

   The Choosing Analyses form appears.
3. In the Choosing Analyses form, highlight *pss* and set up the PSS analysis.
4. In the Choosing Analyses form, highlight *pnoise.*

   The Choosing Analyses form changes to let you specify information for a Pnoise analysis.
5. In the Choosing Analyses form, perform the following:
   - **a.** Choose *correlations* for the *Noise Type*.
   - **b.** Choose the first output of the circuit.

This can be a port or voltage if you want to calculate noise parameters starting in the impedance representation. It can be a current if you want to use the admittance representation. For example, in a mixer, you might choose the IF output here.

c. Choose an appropriate frequency range and sweep for the analysis.

For example, for the mixer you might want to sweep 100MHz around the IF output. Choose an appropriate set of *cycles* to use for the analysis. In a mixer, you might set `maxcycles=1` or add cycle `1` if you select the IF as output. This specification calculates noise correlations between the IF output at frequency ω and the noise at the RF input frequency $\omega + \omega_0$, where ω_0 is the LO frequency.

6. In the Simulation window, choose *Simulation – Netlist and Run.*

The simulation runs.

7. To display the results, perform the following steps:

a. In the Simulation window, choose *Tools – Results Browser*.

The Results Browser appears.

b. In the Results Browser, choose the pnoise analysis and the appropriate cycle of interest. (Correlations of noise at different frequencies appear as different *harmonics* in the browser.)

A list of nodes and sources appears.

The correlation between the noise at the output you specify in the Simulation window and the noise induced at each of the nodes or sources has been calculated. You can plot it directly or further process it in the calculator.

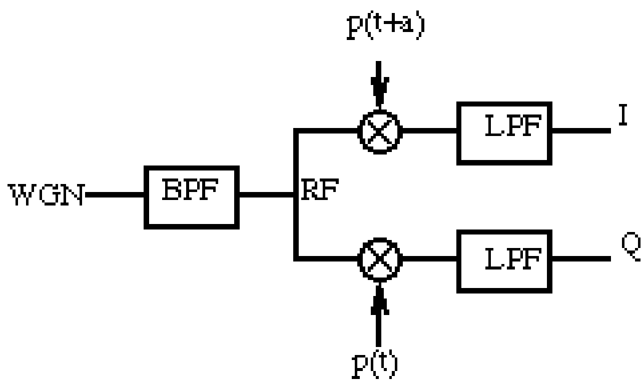
To analyze the correlations with noise at a differential input/output, subtract the cross-power spectra appearing on the two nodes.

Note: Results of the noise correlations analysis are in units of power (such as V^2/Hz). and the cross-power spectra are usually complex.

Cyclostationary Noise Example

As an example which illustrates the various aspects of cyclostationary noise, consider the simple mixer circuit shown in Figure I-6.

Figure I-6 Simple Mixer Circuit



In this simple mixer circuit, white Gaussian noise passes through a high-order band-pass filter with center frequency ω_0. Then it is multiplied by two square-waves which have a phase shift a with respect to each other. Finally the output of the ideal multipliers is put through a one-pole low-pass filter to produce I and Q outputs.

The time-domain behavior of the noise is examined first. The most dramatic effect can be seen by looking directly at the mixer outputs in Figure I-7 on page 1343. This figure shows the contributions to the time-varying noise power made by three separate source frequencies. Two of the source frequencies were selected around ω_0, the third source frequency was selected away from ω_0, slightly into the stop band of the band-pass filter. The sharp change in noise power over the simulation interval occurs because the mixers were driven with square-wave LO signals.

Figure I-7 Time-Varying Noise Power Before Low-Pass Filter

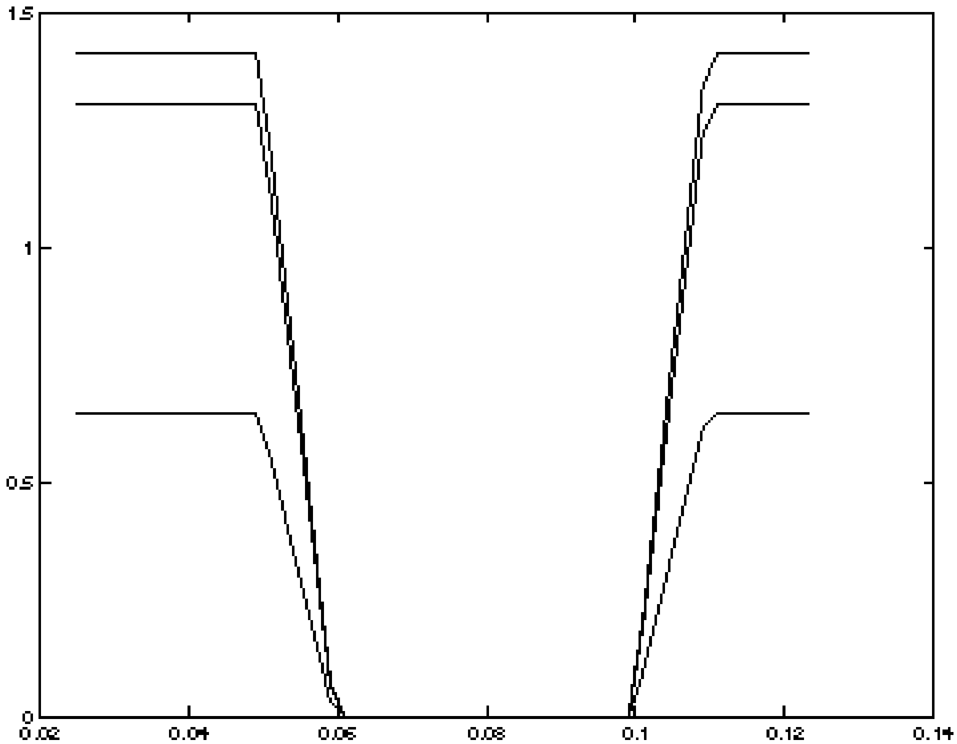
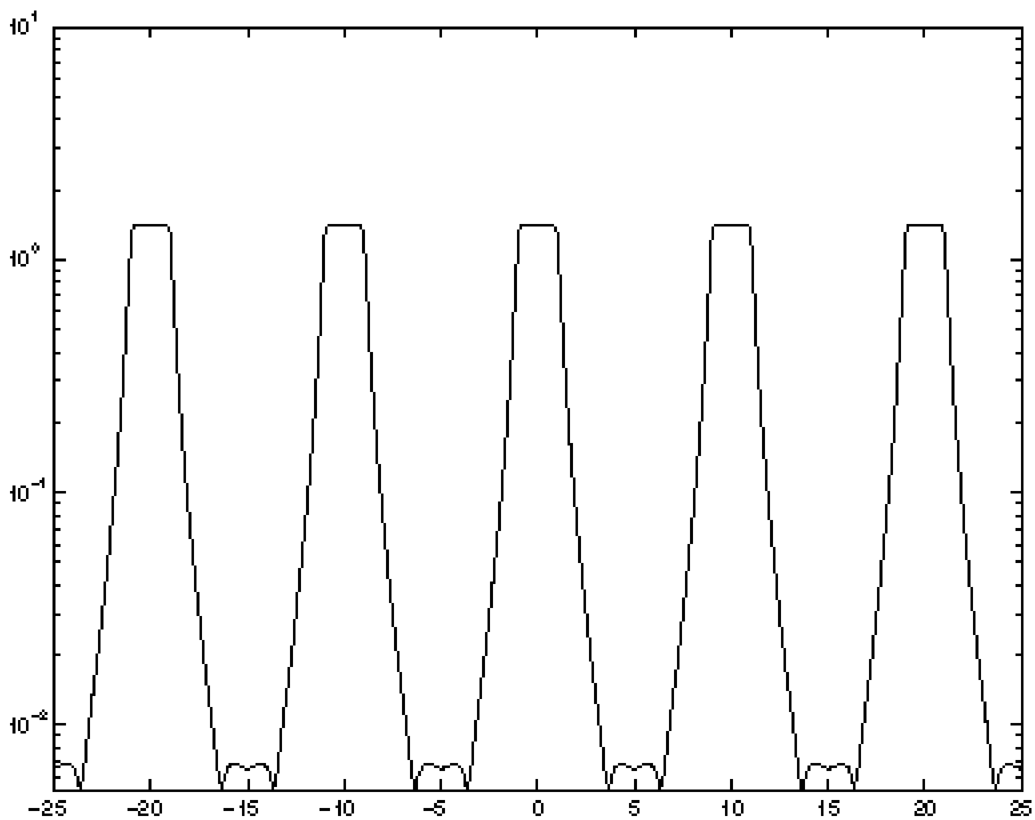


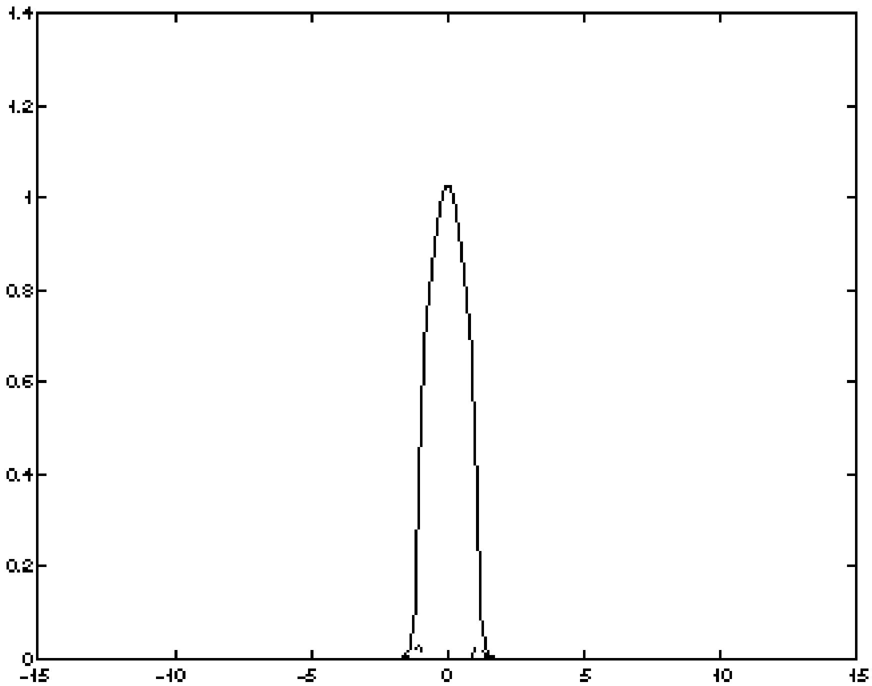
Figure I-8 on page 1344 shows the spectrum of a sampled noise process. Note the periodically replicated spectrum.

Figure I-8 Spectrum of a Sampled Noise Process



The noise behavior at the output ports is examined next. The output spectra at the I and Q outputs are shown in Figures I-9 and I-10. The noise density at I is concentrated around zero because the noise at the RF input to the mixers (band-limited around ω_0) is shifted down to zero and up to $2\omega_0$, but components not around zero are eliminated by the low-pass filter.

Figure I-9 Power Spectra With LO Tones 90^{deg} Out of Phase



More interesting is the cross-correlation spectrum of the I and Q outputs, shown as the dashed line in Figures I-9 and I-10. When the signals applied to the mixers are 90 degrees out of phase (as in Figures I-9), the cross-power spectral density of the noise at the separate I and Q outputs is small, indicating little noise correlation. If the tones are not quite out of phase (as in Figures I-10), the correlation is much more pronounced, though in neither case is it completely zero.

In Figures I-9 and I-10, the solid and dashed lines represent the following

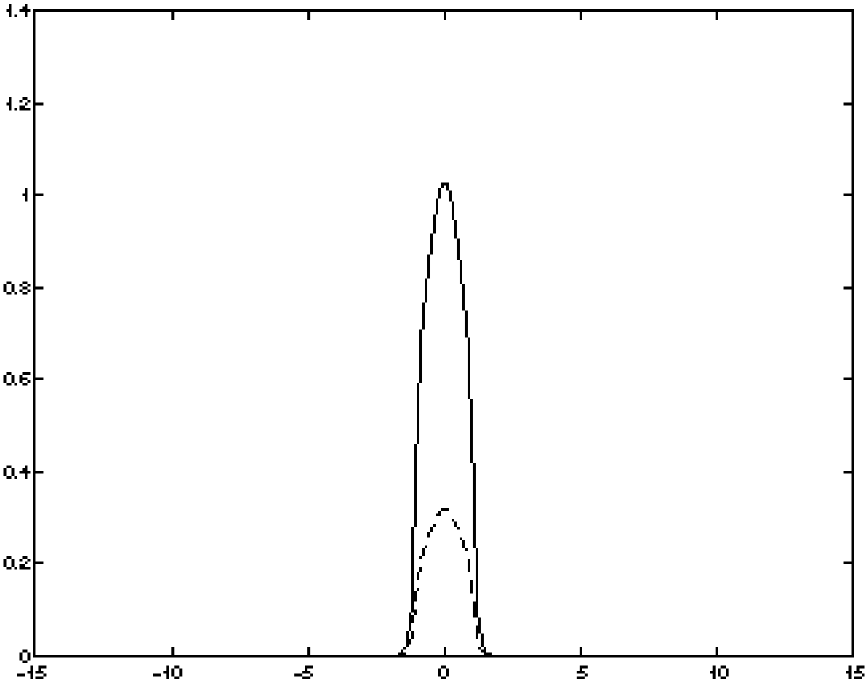
- The solid line represents the power spectrum for the I output with the function

$$S_{II}(\omega)^0$$

- The dashed line represents the cross-spectral density for I and Q with the function

$$S_{IQ}^{0}(\omega)$$

Figure I-10 Power Spectra With LO Tones 72^{deg} Out of Phase

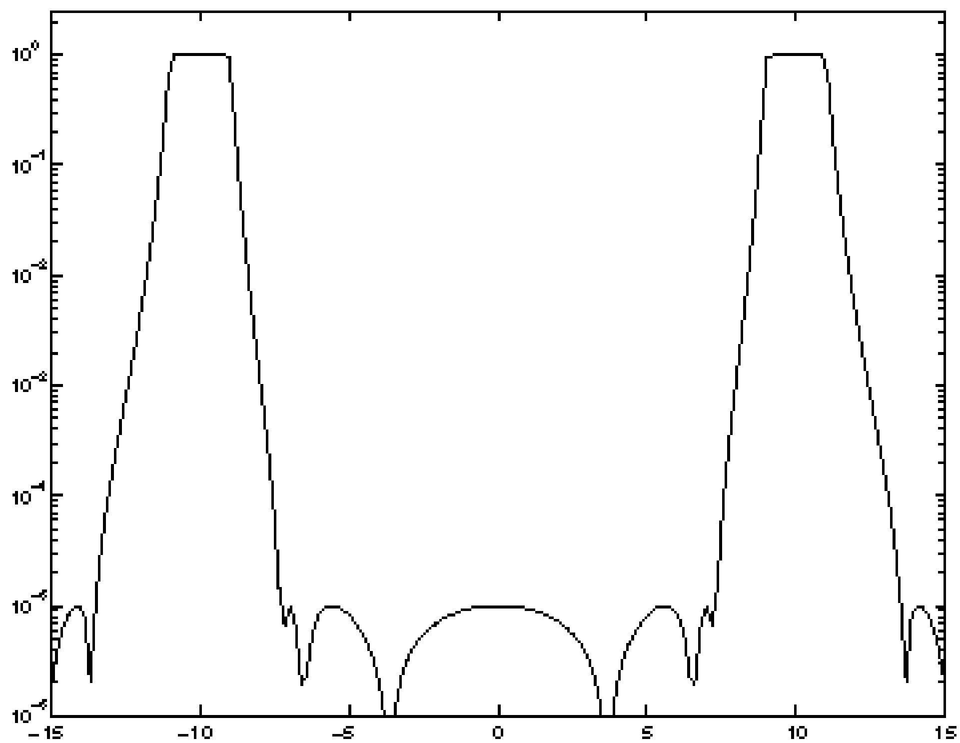


A more interesting example comes from examining the correlation between the noise at the I output and the noise at the RF input. The density function as given by

$$S_{IR}^{(1)}(\omega)$$

is significant because it represents the correlation between the noise at the I output around the baseband frequency with the noise at the RF input, ω_0 higher in frequency. The correlation is high because the noise at the RF input is centered around ω_0 and converted to zero-centered noise by the mixer.

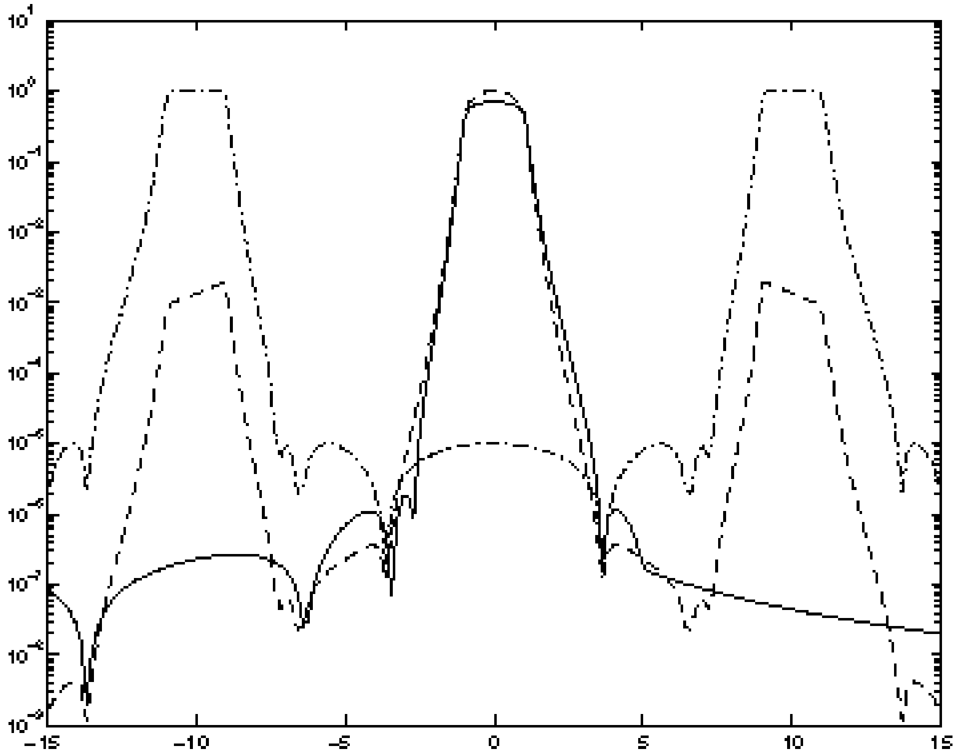
Figure I-11 Noise Spectrum at the RF Input



In Figure I-11, the noise spectrum at the RF input is given by the following function

$S^{0}_{RR}(\omega)$

Figure I-12 Noise Spectrum at Various Power Densities



In Figure I-12, the solid, dashed, and dashed-dot lines represent the following

- The solid line represents the cross power spectrum which indicates correlation between output noise power at the I output versus noise at the RF input that is one harmonic higher in frequency. This is represented by the following function

$$S^{(1)}_{IR}(\omega)$$

- The dashed line represents the noise spectrum at the I output with the following function

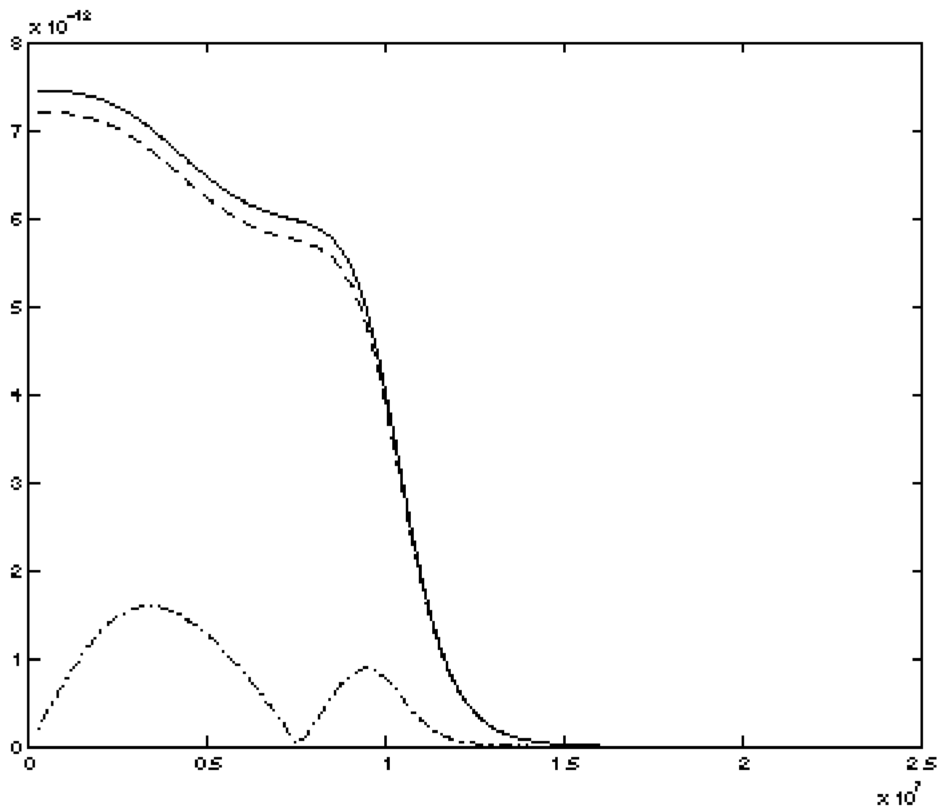
$$S^{(0)}_{II}(\omega)$$

- The dashed-dot line represents the noise spectrum at the RF input with the following function

$$S_{RR}^{(0)}$$

Finally a detailed circuit example was considered. A transistor-level image-reject receiver with I and Q outputs was analyzed. The noise spectra at the I and Q outputs were found to be very similar, as shown in Figure I-13.

Figure I-13 Power Spectral Densities of an Image-Reject Receiver

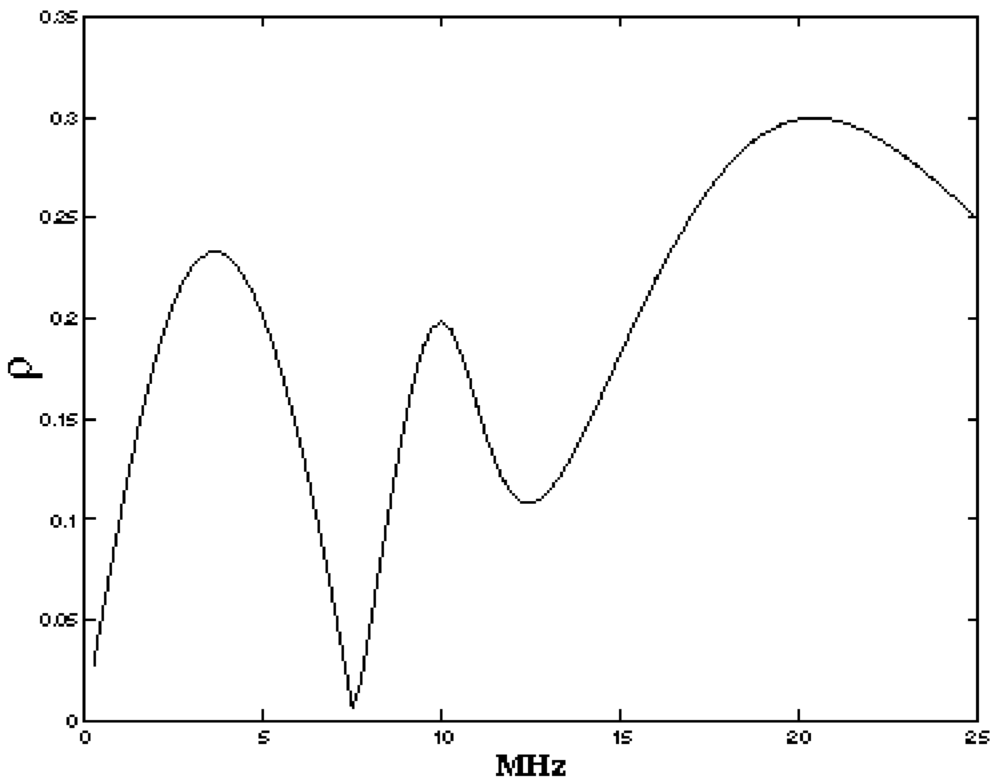


In the image-reject receiver example shown in Figure I-13, the power spectral densities are represented as follows

- I output is a solid line
- Q output is a dashed line
- IQ cross-power density is a dash-dot line

The IQ cross-power density was smaller, but not negligible, indicating that the noise at the two outputs is partially correlated. The correlation coefficient between noise at the I and Q outputs of the image-reject receiver is shown in Figure **I-14**.

Figure I-14 Correlation Coefficient for Output Noise of Image-Reject Receiver



Reference Information on Time-Varying Noise

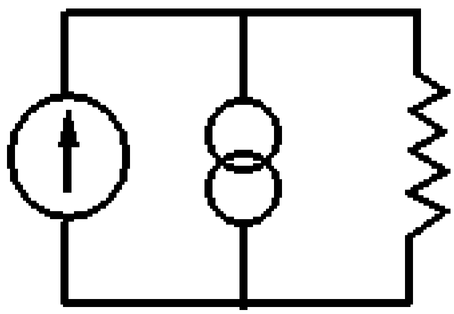
The following sections provide background and reference information on the following noise-related topics

- Thermal Noise
- Linear Systems and Noise
- Time-Varying Systems and the Autocorrelation Function
- Time-Varying Systems and Frequency Correlations
- Time-Varying Noise Power and Sampled Systems

Thermal Noise

The term *noise* is commonly used to refer to any unwanted signal. In the context of analog circuit simulation, noise is distinguished from such phenomena as distortion in the sense that it is non-deterministic, being generated from *random* events at a microscopic scale. For example, suppose a time-dependent current $i(t)$ is driven through a linear resistor, as shown in Figure I-15.

Figure I-15 Deterministic Current Source Driving a Noisy Linear Resistor

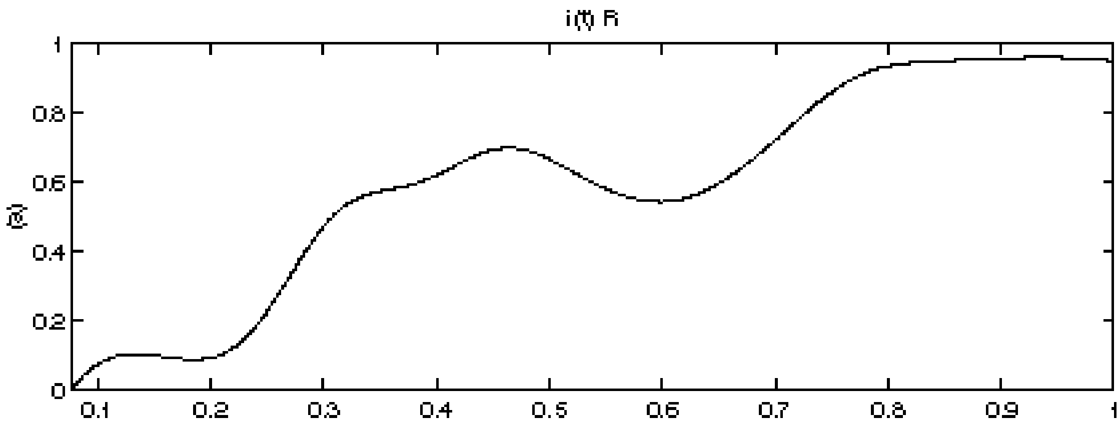


The voltage that appears across the resistor will be

$$v(t) = i(t)R + n(t)$$

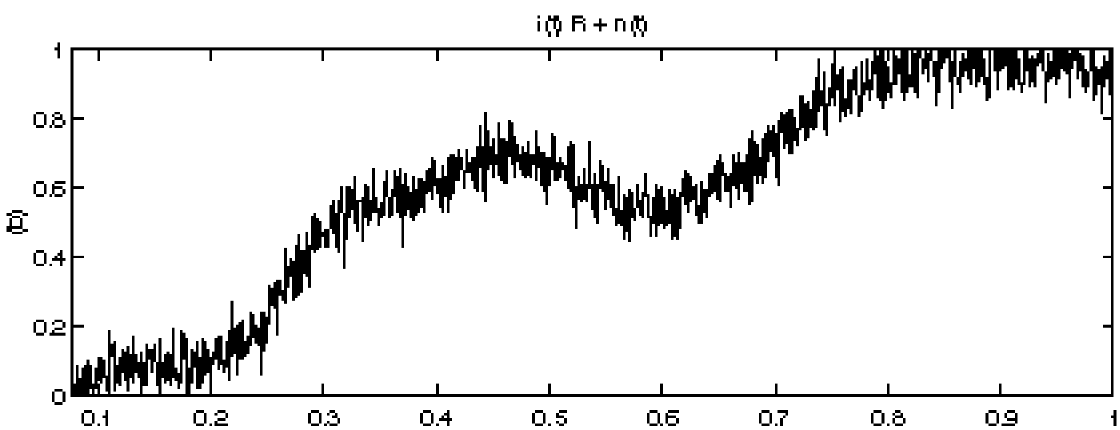
The desired signal $i(t)R$, shown in FigureI-16, is corrupted by an added noise voltage $n(t)$ that is due to resistive thermal noise. The thermal noise of the resistor is modelled by a current source in parallel with the resistor.

Figure I-16 The Desired Signal i(t)R



The total measured voltage is shown in Figure I-17.

Figure I-17 The Total Measured Voltage

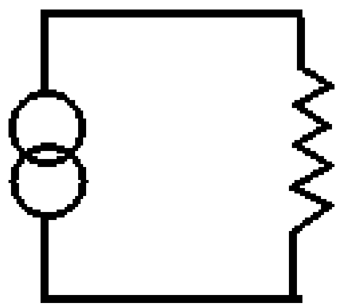


The added noise process alone, $n(t)$, is a random process and so it must be characterized in ways that are different than for deterministic signals. That is, at a time t_0 the voltage produced by the driven current can be exactly specified—it is $i_0 \sin t_0 R$. Just by inspecting Figure I-16 we can predict this part of the measured signal.

On the other hand, the exact value of the noise signal cannot be predicted in advance, although it can be measured to be a particular value $n(t_0)$. However, if another measurement is performed, the noise signal $n(t)$ we obtain will be different and Figure I-17 will change. Due to its innate randomness, we must use a statistical means to characterize $n(t)$.

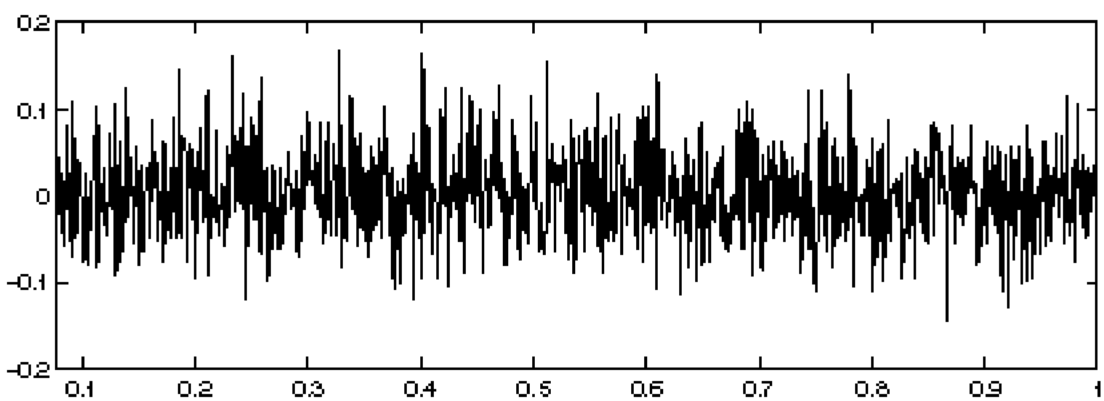
Now consider the circuit in Figure I-18, where we restrict attention to the noise source/resistor pair alone.

Figure I-18 Resistor Modeled as a Noiseless Resistance with an Equivalent Noise Current Source



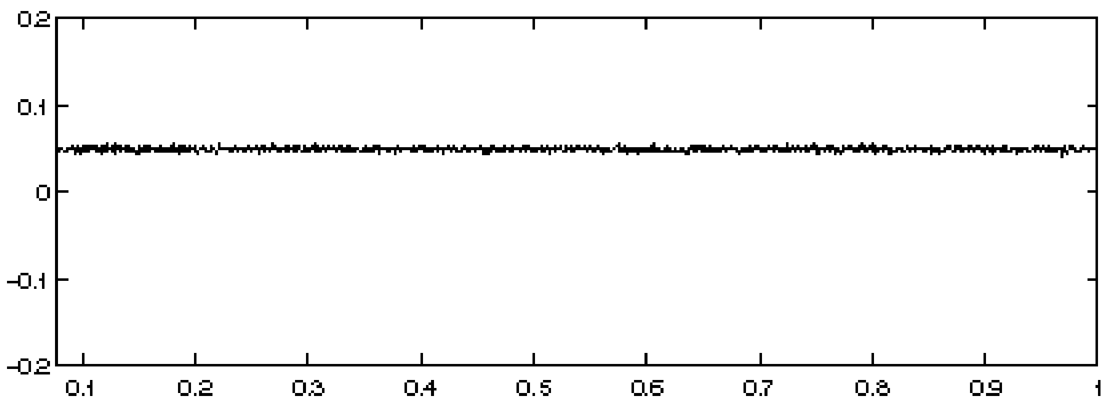
A typical measured noise current/voltage is shown in Figure I-19.

Figure I-19 Typical Measured Noise Current/Voltage



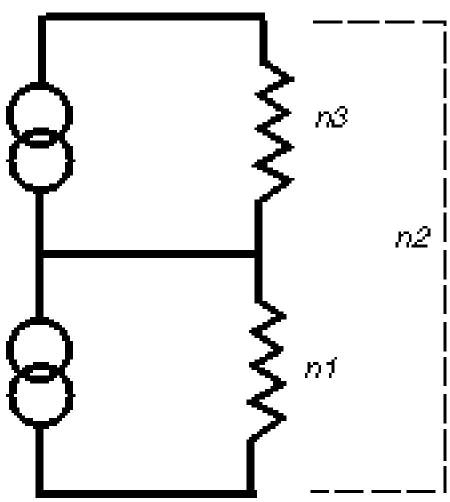
Since we cannot predict the specific value of $n(t)$ at any point, we might instead try to predict what its value would be on average, or what we might *expect* the noise to be. For example, if we measure many noise voltage curves in the time domain, $n(t)$, and average over many different curves, we will obtain an approximation to the expected value of $n(t)$ which we denote by $E\{n(t)\}$. For thermal noise, we will find that $E\{n(t)\}=0$. Therefore, instead of computing $E\{n(t)\}$, let us instead compute $E\{n(t)^2\}$, the expected noise power. An example of this sort of measurement is shown in Figure I-20. 250 measurements were needed to compute this curve.

Figure I-20 Expected Noise Power



Now suppose that we wish to tap the circuit at multiple points. Each will have its own noise characteristics, but they are not necessarily independent. Consider the circuit shown in Figure I-21.

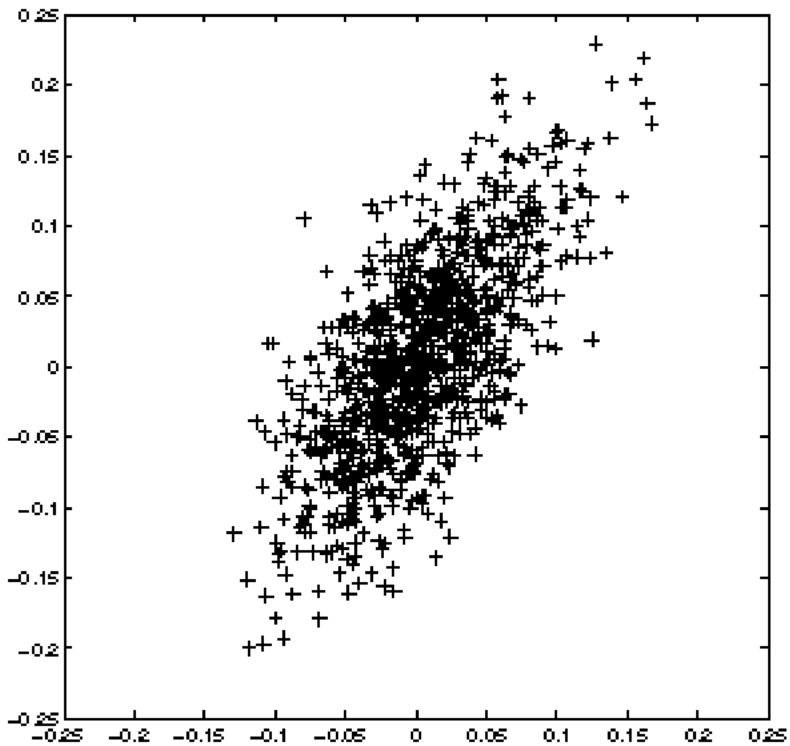
Figure I-21 Circuit Illustrating Correlated Noise



The signals $n_1(t)$ and $n_2(t)$ are obtained by measuring the voltage across a single resistor ($n_1(t)$), and across both resistors ($n_2(t)$), respectively. Just measuring $E\{n_1(t)^2\}$ and $E\{n_2(t)^2\}$ is not enough to predict the behavior of this system, because $n_1(t)$ and $n_2(t)$ are not independent.

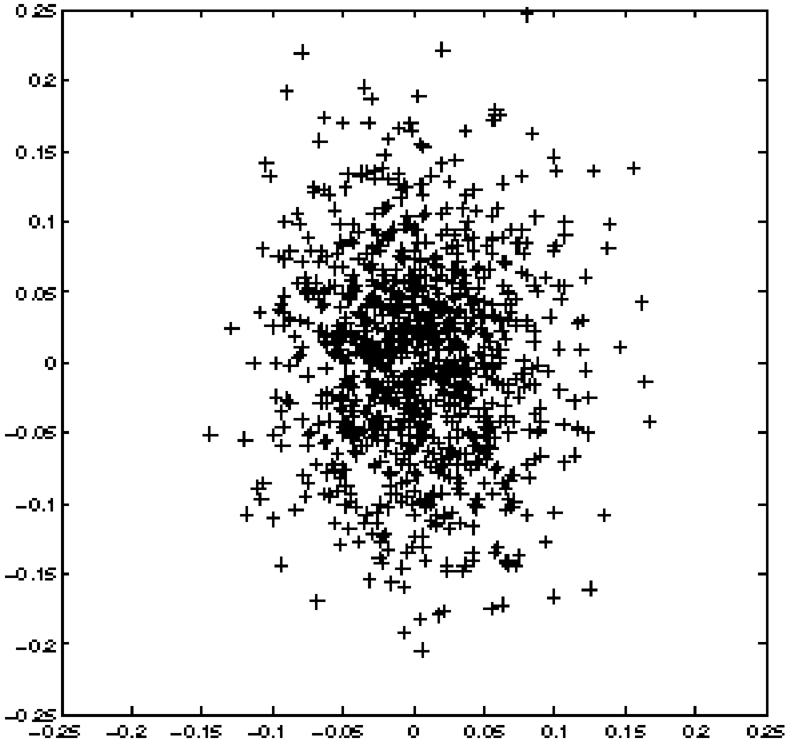
To see $n_1(t)$ and $n_2(t)$ are not independent, consider Figures I-22 and I-23. Samples of each of the processes are taken and plotted on an X-Y graph.

Figure I-22 Samples of $n_1(t)$ Plotted Versus $n_2(t)$



Because $n_1(t)$ composes part of $n_2(t)$, $n_1(t)$ and $n_2(t)$ are correlated so in Figure I-22, the X-Y plot has a characteristic skew along the $X=Y$ line, relative to the $n_1(t)$, $n_3(t)$ plot in Figure I-23,

Figure I-23 Samples of n_1(t) Plotted Versus n_3(t)



The signals $n_1(t)$ and $n_3(t)$ are uncorrelated because they represent thermal noise from different sources. The additional measurement needed to describe the random processes is the measurement of the correlation between the two processes, $E\{n_1(t)n_2(t)\}$. We can also define a time-varying correlation coefficient ρ, with $\rho \in [0,1]$, as

$$\rho(t) = \frac{E\{n_1(t)n_2(t)\}}{\sqrt{E\left\{n_1(t)^2\right\}E\left\{n_2(t)^2\right\}}}$$

A value of $\rho = 0$ indicated completely uncorrelated signals, and a value near one indicates a high degree of correlation. In this example we would find that $\rho(t) = 1/2$, representing the fact that each of the two noise sources contributes half of the process $n_2(t)$.

When there are multiple variables of interest in the system, it is convenient to use matrix notation. We write all the random processes of interest in a vector, for example

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

and then we can write the correlations as the expected value of a vector outer product, $E\{x(t)x^H(t)\}$, where the H superscript indicates Hermitian transpose.

For example, we might write a time-varying correlation matrix as

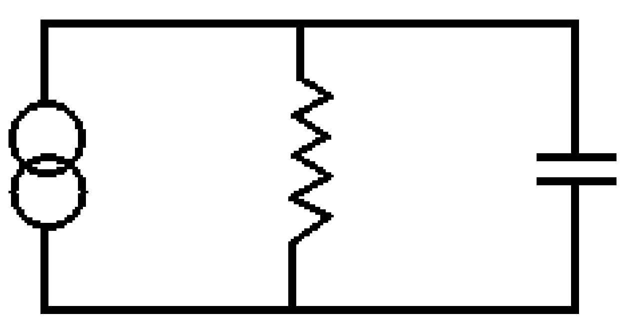
$$R_{xx}(t, t) \equiv E\{x(t)x^H(t)\} = \begin{bmatrix} E\{x_1(t)x_1(t)\} & E\{x_1(t)x_2(t)\} \\ E\{x_2(t)x_1(t)\} & E\{x_2(t)x_2(t)\} \end{bmatrix}$$

Linear Systems and Noise

The examples in the preceding sections describe how to characterize purely static systems. Now we need to add some elements with memory, such as inductors and capacitors.

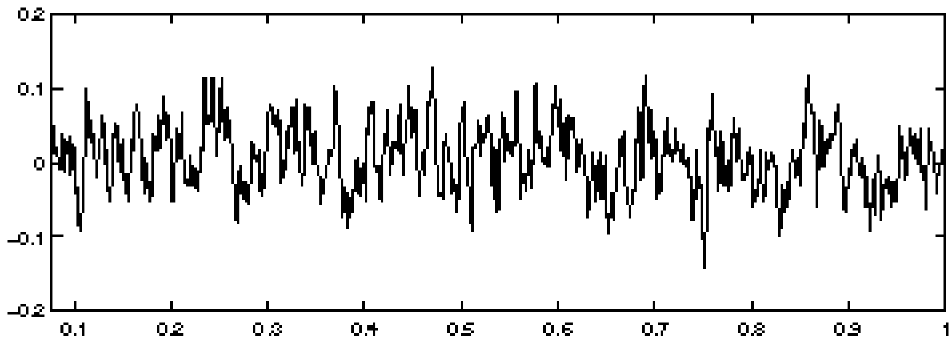
As an first example, consider adding a capacitor in parallel to the simple resistor, as shown in Figure I-24.

Figure I-24 A Simple RC Circuit



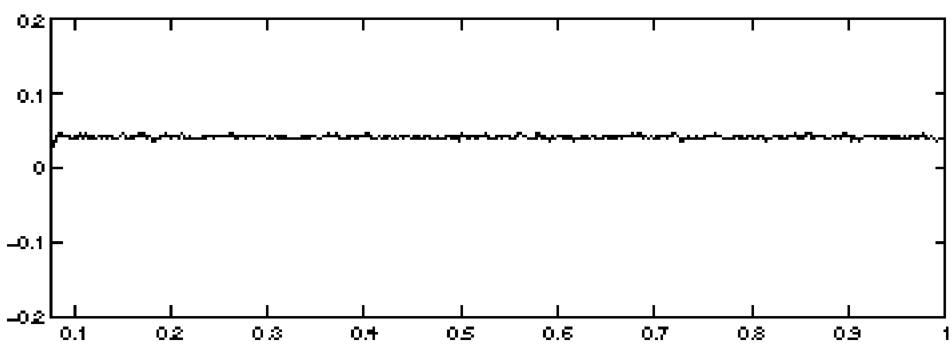
A sample of the noise process in shown in Figure I-25.

Figure I-25 Noise Process for a Simple RC Circuit



The noise looks different than the noise of the resistor alone, because the low-pass filter action of the RC circuit eliminates very high frequencies in the noise. However, we cannot see this effect simply by measuring $E\{n(t)^2\}$ as shown in Figure I-26.

Figure I-26 Expected Noise Power for an RC Circuit



The measurement of $E\{n(t)^2\}$ is independent of time for an RC circuit, just as it was for the for the resistor circuit.

Spectral Densities in Two Simple Circuits

Instead of expected noise power, let us look at the expected power density in the frequency domain. Let $n(\omega)$ denote the Fourier transform of one sample of $n(t)$. Then, $E\{n(\omega)n(\omega)^*\}$ is the expected power spectral density, which we denote by $S_n(\omega)$.

In the present case, the capacitor has a pronounced effect on the spectral density. Figure I-27 shows a computed power spectral density for the resistor thermal noise previously considered. The spectrum is essentially flat (some deviations occur because a finite number of samples was taken to perform the calculation). The flat spectrum represents the fact that in the resistor's noise, all frequencies are, in some statistical sense, equally present. We call such a process *white noise*.

Figure I-27 Power Spectral Density for Resistor Thermal Noise

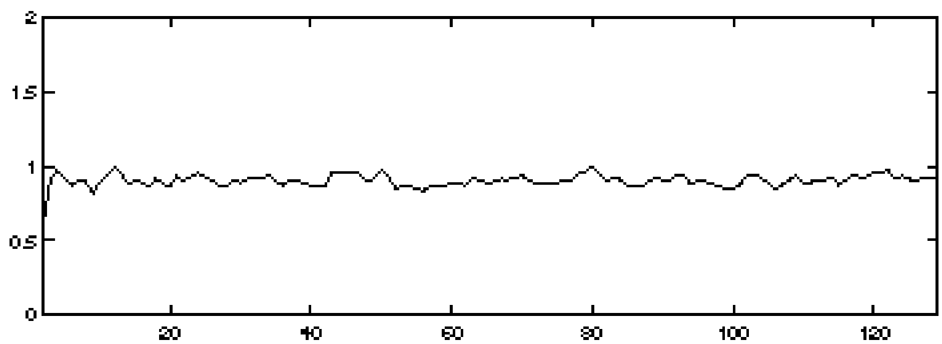
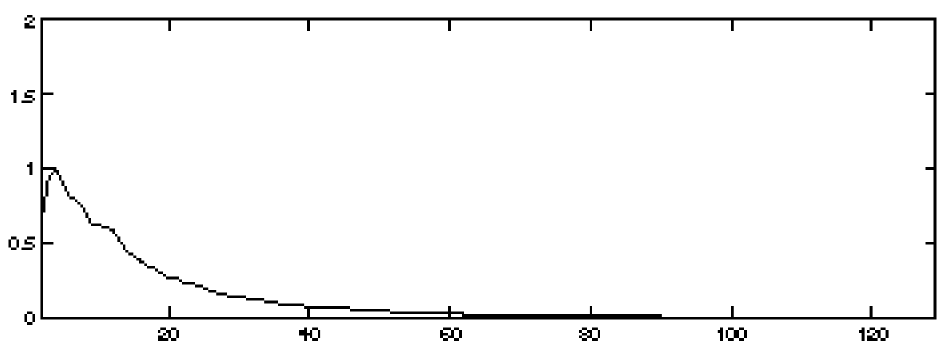


Figure I-28 shows the spectrum of the noise process after filtering by the resistor-capacitor system.

Figure I-28 Resistor-Capacitor Filtered Spectral Noise Process



It is easy to rigorously account for the effect of the RC-filter on the power spectrum of the noise signal. Suppose a random signal x is passed through a time-invariant linear filter with frequency-domain transfer function $h(\omega)$. Then the output is $y(\omega)=h(\omega)x(\omega)$.

Because expectation is a linear operator, we can easily relate the power spectral density of y, $S_y(\omega)$ to $S_x(\omega)$, the power spectral density of x, by using the definitions of y and power density. Specifically,

$$S_y(\omega) = E\{y(\omega)y(\omega)^*\} = E\{h(\omega)x(\omega)x(\omega)^*h(\omega)^*\} = |h(\omega)|^2 S_x(\omega)$$

The noise from the resistor can be considered to be generated by a noise current source i, with power density

$$S_i(\omega) = \frac{4k_B T}{R}$$

placed in parallel with the resistor With the capacitor in parallel, the transfer function from the current source to the resistor voltage is just the impedance $Z(\omega)$,

$$h(\omega) = Z(\omega) = \frac{(1/C)}{j\omega + \frac{1}{RC}}$$

and so the noise voltage power density is

$$S_n(\omega) = \frac{\frac{4k_B T}{RC}}{\omega^2 + \left(\frac{1}{RC}\right)^2}$$

Clearly the spectrum is attenuated at high frequencies and reaches a maximum near zero.

For a vector process, we may define a matrix of power-spectral densities,

$$S_{xx}(\omega) \equiv E\left\{x(\omega)x^H(\omega)\right\}$$

The diagonal terms are simple real-valued power densities, and the off-diagonal terms are generally complex-valued cross-power densities between two variables. The cross-power density gives a measure of the correlation between the noise in two separate signals at a specific frequency. We may define a correlation coefficient as

$$\rho_{ij}(\omega) \equiv \frac{S_{x_i x_j}(\omega)}{[S_{x_i}(\omega) S_{x_j}(\omega)]^{1/2}}$$

It is often more useful to examine the correlation coefficient because the cross-power density may be small. As an example, consider a noiseless amplifier. The noise at the input is simply a scaled version of the noise at the output leading to a $\rho=1$, but the cross-power density will be much smaller than the output total noise power density if the amplifier has small gain.

Note: In a numerical simulation it is important to compute *only* the correlation coefficient when the diagonal spectral densities are sufficiently large. If one of the power densities in the denominator of the correlation-coefficient definition is very small, then a small numerical error could lead to large errors in the computed coefficient, because of division by a number close to zero.

In the vector case, the transfer function is also a matrix $H(\omega)$, such that $y(\omega)=H(\omega)x(\omega)$ and so the spectral densities at the input and output are related by

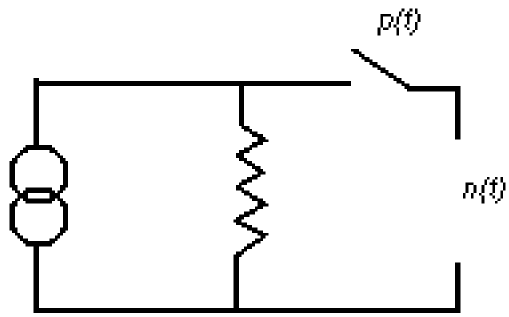
$$S_{yy}(\omega) = E\left\{H(\omega)x(\omega)x^H(\omega)H^H(\omega)\right\} = H(\omega)S_{xx}(\omega)H^H(\omega)$$

Time-Varying Systems and the Autocorrelation Function

If all the sources of noise in a system are resistors, and the circuit consists strictly of linear time-invariant elements, then the matrix of spectral densities $S_{xx}(\omega)$ is sufficient to describe the noise. However, most interesting RF circuits contain nonlinear elements driven by time-varying signals. This introduces time-varying noise sources as well as time-varying filtering. Because most noise sources are small, and generate small perturbations to the circuit behavior, for purposed of noise analysis, most RF circuits can be effectively modeled as linear time-varying systems. The simple matrix of power spectra is not sufficient to describe these systems.

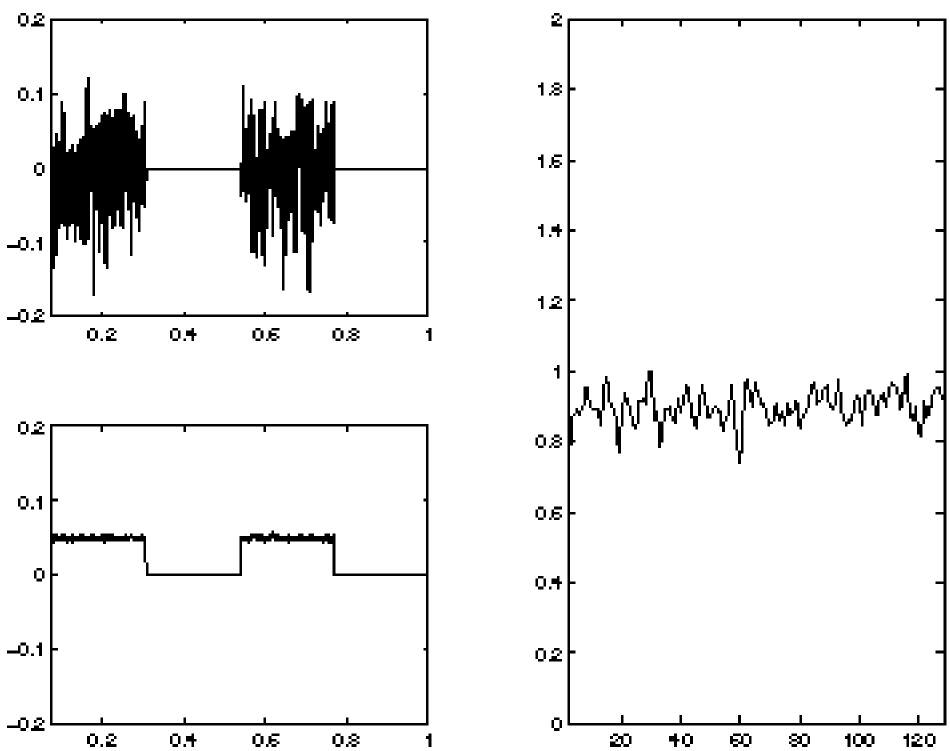
To see this, return to the simple resistor example. Suppose that a switch is connected between the resistor and the voltage measuring device, as shown in Figure I-29.

Figure I-29 SimpleTime-Varying Circuit with Switch



Further suppose that the switch is periodically opened and closed. When the switch is open, there is no noise measured. When the switch is closed, the thermal noise is seen at the voltage output. A typical noise waveform is shown on the bottom left in Figure I-30.

Figure I-30 Typical Waveforms for Noise, Noise Power and White Noise



The time-varying noise power $E\{n(t)^2\}$ can be computed and is shown in Figure I-30 on the top left, above the time-varying noise waveform. The expected power periodically switches between zero and the value expected from the resistor noise. This is different than the resistor-only and resistor-capacitor systems considered previously. Indeed, no linear time-invariant system could create this behavior. However, if we examine the power spectrum on the right in Figure I-30, we again find that it is flat, corresponding to *white* noise.

The Autocorrelation Function

At this point it is clear that $E\{n(t)\}$ and $E\{n(t)^2\}$ do not completely specify the random process $n(t)$, nor does the power spectral density. To obtain a complete characterization, consider measuring $n(t)$ at two different timepoints, t_1 and t_2. $n(t_1)$ and $n(t_2)$ are two separate random variables. They may be independent of each other, but in general they will have some correlation. Therefore, to completely specify the statistical characteristics of $n(t_1)$ and $n(t_2)$ together, we must specify not only the variances $E\{n(t_1)^2\}$ and $E\{n(t_2)^2\}$, but also the covariance $E\{n(t_1)n(t_2)\}$. In fact since $n(t)$ has infinite dimension, an infinite number of these correlations must be specified to characterize the entire random process. The usual way of doing this is by defining the autocorrelation function $R_n(t,t+\tau) = E\{n(t)n(t+\tau)\}$.

If $x(t)$ is a vector process,

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

then we define the autocorrelation matrix as

$$R_{xx}(t, t+\tau) \equiv E\left\{x(t)x^H(t+\tau)\right\} = \begin{bmatrix} E\{x_1(t)x_1(t+\tau)\} & E\{x_1(t)x_2(t+\tau)\} \\ E\{x_2(t)x_1(t+\tau)\} & E\{x_2(t)x_2(t+\tau)\} \end{bmatrix}$$

where superscript H indicates Hermitian transpose.

The diagonal term gives the autocorrelation function for a single entry of the vector, e.g, $E\{x_1(t)x_t(t+\tau)\}$. For $\tau=0$, this is the time-varying power in the single process, e.g. $E\{x_1(t)^2\}$. If the process $x(t)$ is Gaussian, it is completely characterized by its autocorrelation function $R_x(t, t+\tau)$ since all the variances and co-variances are now specified.

We can also precisely define what it means for a process to be *time-independent*, or *stationary*—A stationary process is one whose autocorrelation function is a function of τ

only, not of t. This means that not only is the *noise power* $E\{n(t)^2\}$ independent of t, but the correlation of the signal at a time point with the signal at another timepoint is only dependent on the difference between the timepoints, τ. The white noise generated by the resistor, and the RC-filtered noise, are both stationary processes.

Connecting Autocorrelation and Spectral Densities

At different points in the discussion above it was claimed that the expected time-varying power $E\{n(t)^2\}$ of the resistor voltage is constant in time, and also the power density $S_n(\omega)$ is constant in frequency. At first this seems odd because a quantity that is *broad* in time should be *concentrated* in frequency, and vice versa.

The answer comes in the precise relation of the spectral density to the autocorrelation function. Indeed, it turns out that the spectral density is the Fourier transform of the autocorrelation function, but with respect to the variable τ, not with respect to t. In other words, the measured spectral density is related to the correlation of a random process with time-shifted versions of itself. Formally, for a stationary process $R_n(t,t+\tau) = R_n(t))$ we write

$$S_n(f) = \int_{-\infty}^{\infty} e^{i\omega\tau} R_n(\tau) d\tau$$

For example, in the resistor-capacitor system considered above, we can calculate the autocorrelation function $R_n(\tau)$ by an inverse Fourier transform of the power spectral density, with the result

$$R_n(\tau) = \left(\frac{4k_B T}{C}\right) e^{-|\tau| / (RC)}$$

From inspecting this expression we can see that what is happening is that adding a capacitor to the system creates memory. The random current process generated by the thermal noise of the resistor has no memory of itself so the currents at separate time-instants are not correlated. However, if the current source adds a small amount of charge to the capacitor, the charge takes a finite amount of time to discharge through the resistor creating voltage. Thus voltage at a time-instant is correlated with the voltage at some time later, because part of the voltage at the two separated time instants is due to the same bit of added charge. From inspecting the autocorrelation function it is clear that the correlation effects last only as long as the time it takes any particular bit of charge to decay, in other words, a few times the RC time constant of the resistor-capacitor system.

Note that the process is still stationary because this memory effect depends only on how long has elapsed since the bit of charge has been added, or rather how much time the bit of charge has had to dissipate, not the absolute time at which the charge is added. Charge added at separate times is not correlated since arbitrary independent amounts can be added at a given instant. In particular, the time-varying noise power,

$$E\left\{n(t)^2\right\} = \int_{-\infty}^{\infty} S_n(\omega)d\omega$$

Time-Varying Systems and Frequency Correlations

Now we have seen that the variation of the spectrum in frequency is related to the correlations of the process, in time. We might logically expect that, conversely, variation of the process in time (that is, non-stationarity) might have something to due with correlations of the process in frequency. To see why this might be the case, suppose we could write a random process x as a sum of complex exponentials with random coefficients,

$$x = \sum_{k=-K}^{K} c_k e^{i\omega t}$$

Noting that $c_{-k}=c_k*$, the time-varying power in the process is

$$E\left\{x^2(t)\right\} = \sum_{k=-K}^{K} \sum_{l=-K}^{K} E\{c_k c*_l\} e^{i(w_k - w_l)t}$$

and it is clear that $E\{x(t)^2\}$ is constant in time if and only if

$$E\{c_k c*_L\} = \left|c_k\right|^2 \varsigma_{kl}$$

In other words, the coefficients of expansion of sinusoids of different frequencies must be uncorrelated. In general, a stationary process is one whose frequency-domain representation contains no correlations across different frequencies.

To see how frequency correlations might come about, let us return to the resistor-switch example. Let $n(t)$ denote the voltage noise on the resistor, and $h(t)$ the action of the switch, so that the measure voltage is given by $v(t) = h(t)n(t)$, where $h(t)$ is periodic with period T and frequency $\omega_0 = 2\pi/T$. The time-domain multiplication of the switch becomes a convolution in the frequency domain, $v(\omega) = h(\omega) \otimes n(\omega)$ where $\otimes$ denotes convolution. Since $h(t)$ is periodic, its frequency-domain representation is a series of Dirac deltas,

$$h(\omega) = \sum_k h_k \varsigma(\omega - k\omega_0)$$

and so

$$v(\omega) = \sum_k h_k n(\omega - k\omega_0)$$

and the spectral power density is simply

$$S_v(\omega) = E\{\langle v(\omega)v(\omega)^*\rangle\} = \sum_k \sum_l h_k h_l^* E\{n(\omega - k\omega_0)n(\omega - l\omega_0)^*\}$$

Since the process n is stationary, this reduces to

$$S_v(\omega) = \sum_k \left|h_k\right|^2 S_n(\omega - k\omega_0)$$

Since $S_n(\omega)$ is constant in frequency, $S_v(\omega)$ is also.

However, the process v is no longer stationary because frequencies separated by multiples of ω_0 have been correlated by the action of the time-varying switch. We may see this effect in the time-variation of the noise power, as in Figure I-30 on page 1362, or we may examine the correlations directly in the frequency domain.

To do this, we introduce the *cycle spectra*

$$S_{xx}^{\alpha}(\omega)$$

that are defined by

$$S_{xx}^{\alpha}(\omega) = E\left\{x(\omega)x^{H}(\omega+\alpha)\right\}$$

and are a sort of cross-spectral density, taken between two separate frequencies. $S_0(\omega)$ is just the power spectral density we have previously discussed. In fact we can define a frequency-correlation coefficient as

$$\rho_n^{\alpha}(\omega) \equiv \frac{S_n(\omega)^{\alpha}}{\sqrt{S_n(\omega)S_n(\omega+\alpha)}}$$

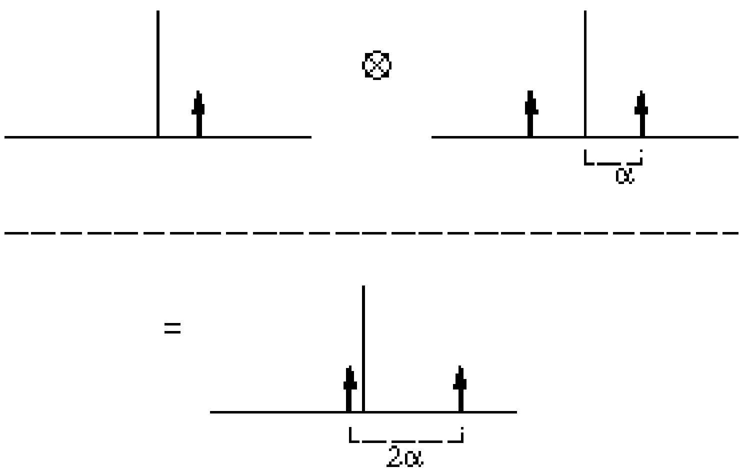
and if

$$\rho_n^{\alpha}(\omega) = 1$$

then the process n has frequency content at ω and $\omega+\alpha$ that is perfectly correlated.

Consider separating out a single frequency component of a random process and multiplying by a sinusoidal waveform of frequency α, as shown in Figure I-31. The component at ω is shifted to re-appear at $\omega + \alpha$ and $\omega - \alpha$. The new process' frequency components at $\omega - \alpha$ and $\omega + \alpha$ are deterministically related to the components of the old process located at ω. Therefore, they are correlated, and $S^{2a}(\omega)$ is non-zero.

Figure I-31 Time-Variation Introduces Frequency Correlation



Physically, what happens is that to form a waveform with a defined *shape* in time, the different frequency components of the signal must add in a coherent, or correlated fashion. In a process like thermal noise, the Fourier coefficients at different frequencies have phase that is randomly distributed with respect to each other, and the Fourier components can only add incoherently. Their powers add, rather than their amplitudes. Frequency correlation and time-variation of statistics are thus seen to be equivalent concepts.

Another way of viewing the cycle spectra is that they represent, in a sense, the two-dimensional Fourier transform of the autocorrelation function, and are therefore just another way of expressing the statistics of the process.

Time-Varying Noise Power and Sampled Systems

Again supposing the signal n to be cyclostationary with period T, for each sample phase $\xi \in [0,T)$, we may define the discrete-time autocorrelation function

$$R_n^{\xi}(p, q)$$

to be

$$R_n^{\xi}(p, p+q) = R_n(\xi + pT, \xi + (p+q)T)$$

Because the cyclostationary process R_n is periodic, by inspection

$$R_n^{\xi}(p, p+q)$$

is independent of p and thus stationary, i.e.

$$R_n^{\xi}(p, p+q) = R_n^{\xi}(q)$$

Note that

$$R_n^{\xi}(p, p) = R^{\xi}(0)$$

gives the expected noise power, $R_n(\xi,\xi)$, for the signal at phase ξ. Plotting $R^{\xi}(0)$ versus ξ will show how the noise power varies periodically with time.

The discrete-time process

$$R_n^{\xi}(p, p+q) = R_n^{\xi}(q)$$

can be described in the frequency-domain by its discrete Fourier transform,

$$R_n^{\xi}(\phi) = \sum_{q=-\infty}^{\infty} R^{\xi}(q) e^{iq2\pi\phi T}$$

Note that the spectrum of the discrete (sampled) process

$$R_n^{\xi}(\phi)$$

is periodic in frequency with period *1/T*.

All noise power is aliased into the Nyquist interval *[–1/2T, 1/2T]* (or, equivalently, the interval *[0,1/T]*). Generally it is the noise spectrum which is available from the circuit simulator. To obtain the autocorrelation function or time-varying noise power, an inverse Fourier integral must be calculated by

$$R_n^{\xi}(q) = \int_0^{1/T} R_n^{\xi}(\phi) e^{iq2\pi\phi} d\phi$$

Summary

- All useful noise metrics can be interpreted in terms of correlations. Physically these can be interpreted as the expected value of two-term products. In the case of random vectors these are expected values of vector outer products.
- The power spectral density of a variable indexed i is

  $$S_{x_i x_i}(\omega) = E\{x_i(\omega)x_i(\omega)^*\}$$

  This is what the current SpectreRF pnoise analysis computes.
- $S_{xx}(\omega)$ is constant if and only if x is a white noise process. In that case $R_{xx}(\tau) = R\delta(\tau)$ if there are no correlations in time for the process.
- The cross-power densities of two variables x_i and x_j are

  $$S_{x_i x_j}(\omega) = E\left\{x_i(\omega)x_j(\omega)^H\right\}$$

  If and only if the two variables have zero correlation at that frequency, then

  $$S_{x_i x_j} = 0$$

A correlation coefficient may be defined as

$$\rho_{ij}(\omega) \equiv \frac{S_{x_i x_j}(\omega)}{\sqrt{S_{x_i}(\omega) S_{x_j}(\omega)}}$$

and $\rho_{ij}(f) \in [0,1]$.

- The cycle-spectra

$$S_{xx}^{\alpha}(f)$$

represent correlations between frequencies separated by the cycle-frequency α

$$S_{xx}^{\alpha}(f) = E\left\{ x(\omega) x^{H}(\omega + \alpha) \right\}$$

For a single process x_i, a correlation coefficient may be defined as

$$\rho_{x_i}^{\alpha}(\omega) \equiv \frac{S_{x_i x_i}(\omega)^{\alpha}}{\sqrt{S_{x_i}(\omega) S_{x_i}(\omega + \alpha)}}$$

and

$$\rho_{x_i}^{\alpha}(f) \in [0, 1]$$

- A process is stationary if and only if

$$S_{xx}^{\alpha}(\omega) = 0$$

for all ω and all $\alpha \neq 0$, that is, if there are no correlations in frequency for the process.

In other words,

$$S_{xx}^{\alpha}(\omega) = S_{xx}(\omega)\delta(\alpha)$$

- A process is cyclostationary if

  $$S_{xx}^{\alpha} = 0$$

  for all $\alpha \neq m\omega_0$ for some ω_0 and integer m. Frequencies separated by $m\omega_0$ are correlated. A stationary process passed through a periodically linear-time varying filter will in general be cyclostationary with ω_0 the fundamental harmonic of the filter.

- We might also compute correlations between different nodes at different frequencies, with the obvious interpretation and generalization of the correlation coefficients.

J

Using Tabulated S-parameters

Many passive component models commonly used in RF design are available only as tables of S-parameter data. You can completely characterize any linear, time-invariant circuit network or component by specifying its S-parameter network at each frequency of interest. See "Using the nport Components" on page 1374 for information on how to use S-parameter tables in RF analyses.

SpectreRF uses time-domain shooting methods to achieve excellent performance on large, highly nonlinear circuits. Consequently, using S-parameter data in SpectreRF is not as straightforward as it is with frequency-domain simulators such as those based on harmonic balance. Before performing an RF simulation, such as a PSS analysis, you must first convert the frequency-domain S-parameters into an equivalent time-domain model. This conversion might be time-consuming for large, complicated data sets. However, the *nport* component lets you reuse the converted data set and thus avoid converting the data more than once. The procedure is described in "Model Reuse" on page 1379.

You might encounter three potential difficulties while converting the frequency-domain S-parameter data set into an equivalent time-domain model.

1. Some frequency-domain data sets do not have valid time-domain descriptions. Time-domain models must be stable and causal and you cannot generate a time-domain model for frequency-domain data that lacks these properties.

2. When you convert the frequency-domain S-parameter data into an equivalent time-domain model, the frequencies between the tabulated points must be interpolated. This is because the time-domain description of a frequency-domain model depends on every frequency, not just the ones given at the tabulated points. A special, robust, high-order, rational interpolation algorithm performs the interpolation. The rational interpolation process introduces some error into the final model description. See "Controlling Model Accuracy" on page 1376 to understand and control this error. it also tells you how to deal with data that contains noise that might corrupt the rational interpolation process.

3. Any algorithm that converts S-parameter data to use it in a time-domain simulator must extrapolate the data outside the range of tabulated frequencies. By definition, some frequencies are not included in a tabular data file. In addition, extrapolation might introduce nonphysical effects into the model, particularly when the S-parameter data is

given over a very narrow frequency range. See "Troubleshooting" on page 1378 for information on how to diagnose and solve any problems. It also tells how to interpret the warning messages that SpectreRF produces when a non-physical extrapolation might be occurring.

Using the nport Components

Use the *nport* components to read in S-parameter data. Follow these steps to prepare an *nport* component for use in SpectreRF simulations.

1. Select an appropriate *nport* component from the *analogLib* in the schematic. For example, you might select an *n2port* for a two-port S-parameter description. The number of ports ranges from one (*nport*) to four (*n4port*).
2. Select the *n2port* component in the Schematic window.Then choose Edit—Properties—Objects to display the Edit Object Properties form for the *n2port* component.
3. In the *S-parameters data file* field, type the S-parameter data file name. For example, *sparam1.dat*.

   If your S-parameter data is in an industry standard format other than the Spectre format, you can use the *sptr* tool to convert the data to a format that Spectre can read. See "The S-Parameter File Format Translator (SPTR)" on page 1380 for information about this tool.
4. In the *S-parameter data format* cyclic field, select an industry standard data format if it describes the format of your S-parameter data file.

   If your S-parameter data is in one of the following industry standard formats—*touchstone* or *citi*, you can select that format. If you do not select a data format, Spectre will try to determine the data format.
5. In the *Multiplier* field, enter the multiplicity factor. The default is 1.
6. In the *Scale factor* field, enter the frequency scale factor. The default is 1.
7. In the *No. of Harmonics for PSS* field, enter the number of harmonics to consider in the PSS solution. The default is 20.
8. In the *Thermal Noise* cyclic field, select *yes* or *no*.
9. In the *Thermal noise model* cyclic field, select *internal* or *external.* Selecting *internal* specifies use of the internal thermal noise model. Selecting external specifies use of the noise data in the S-parameter data file. By default, external data is used whenever it is available.

10. In the *Use smooth data windowing* cyclic field, select *yes* or *no*. Determines whether or not to use a smooth data windowing function.

11. Set the *interpolation method* cyclic field to *rational* if you are planning to use SpectreRF analyses. (Select *linear* or *spline* when you are planning to use only Spectre analyses.)

    Note: You must use rational interpolation with SpectreRF.

    When you select *rational*, four new fields are displayed.

12. In the *ROM data file* field, enter the path for the time-domain reduced order model data file (ROM) to create for the converted S-parameter data. You must enter an absolute pathname, for example, */usr/mydir/sparamtorom*. Once SpectreRF converts the S-parameter data to a time-domain model, the file (*sparamtorom* here) stores the time-domain model. Be sure the ROM data file name is distinct from the S-parameter data file name.

    Storing the time-domain model ensures that you do not have to repeat interpolation and conversion later. See “Model Reuse” on page 1379 for further details on the ROM data file.

13. In the *Relative error* field, enter the maximum relative tolerance to allow for rational interpolation errors. The default value is 0.01. When the *nport* model deviates from the supplied S-parameter data by a relative magnitude less than *relerr*, the deviation is generally ignored. Fill in the *Relative error* field as described in “Controlling Model Accuracy” on page 1376.

14. In the *Absolute error* field, enter the maximum absolute tolerance to allow for rational interpolation errors. The default value is $1e^{-4}$. When the *nport* model deviates from the supplied S-parameter data by an absolute magnitude less than *abserr*, the deviation is generally ignored. Fill in the *Absolute error* field as described in “Controlling Model Accuracy” on page 1376.

15. In the *Rational order* field, enter the order of rational function to use in fitting the S-parameter data. If you enter this argument, *relerr* and *abserr* are ignored in selecting the rational function interpolation order. If you do not enter this argument, the simulator will attempt to select an order of rational interpolation that satisfies the *abserr* and *relerr* criteria. Fill in the *Rational order* field as described in “Controlling Model Accuracy” on page 1376.

16. Click *Apply* in the Edit Object Properties form for the *nport* component and run the simulation.

17. Choose *Design—Check and Run* in the Schematic window.

18. Run the simulation.

Controlling Model Accuracy

The *nport* component has three parameters to control the accuracy of the rational interpolation process,

- *relerr* (The *Relative error* field)
- *abserr* (The *Absolute error* field)
- *ratorder* (The *Rational order* field)

You can use these parameters to trade accuracy for model size. Usually, the more stringent the accuracy requirement, the higher the model order. Higher-order models require longer simulation time. Spectre can automatically generate a model that meets a specified accuracy requirement. You can also specify the model order directly to Spectre.

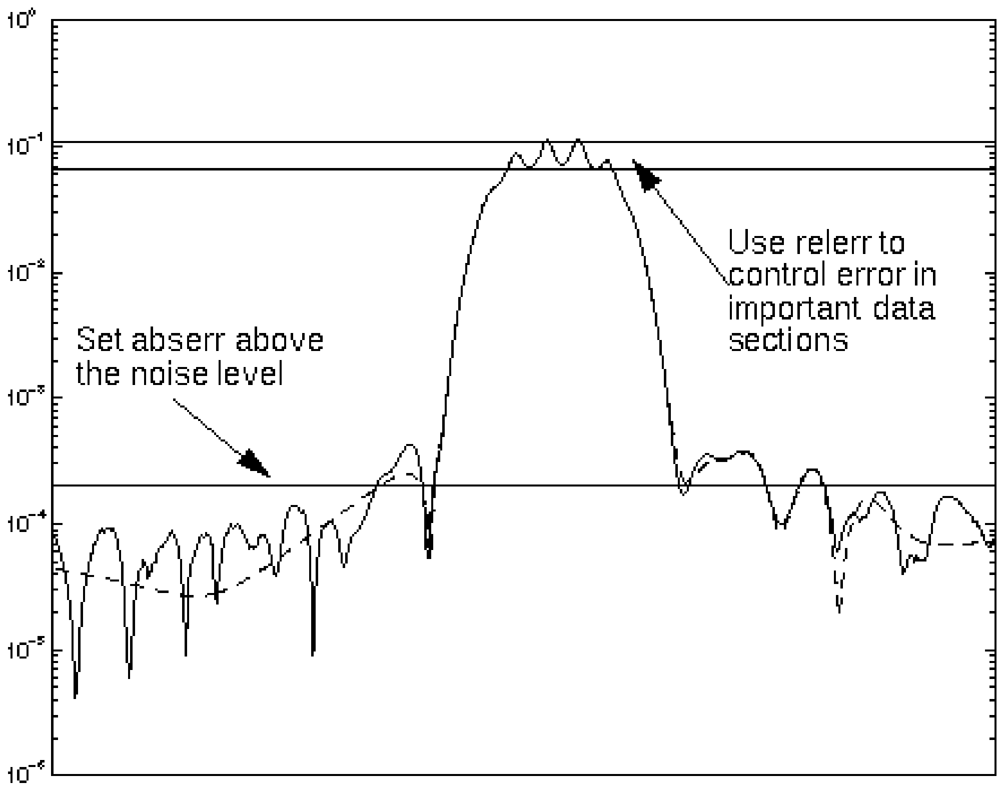
Using relerr and abserr

Let $S_{ij}(\omega)$ denote the i,j entry of the scattering parameter matrix at frequency ω and $\hat{S}_{ij}(\omega)$ the corresponding rational interpolant. The rational interpolation algorithm attempts to find an interpolant such that

$$\max_{\omega,\, i,\, j} \left|S_{ij}(\omega) - \hat{S}_{ij}(\omega)\right| < max(relerr \times \left|S_{ij}(\omega)\right|, abserr)$$

Generally, you use *relerr* for relative error control, and *abserr* for absolute error control. Consider the data shown by the solid line in the figure below. In this example, *relerr* can control the error in the passband, and *abserr* can control the error in the stopband. If the details of the filter behavior in the stopband are not of interest, *abserr* can be set to the level shown, and these details are ignored in the interpolation.

Figure J-1 Using relerr and abserr



This example has approximately a 10% ripple in the passband, so to ensure adequate error control there, set *relerr* to much less than 0.1, perhaps to 0.01.

The dashed line in the figure shows an interpolation function with the error control levels specified in the figure: *abserr*=2e^{-4}, and *relerr*=0.01. Details of the data below the *abserr* threshold are not resolved by the rational interpolation.

To recover the details in the stopband, set *abserr* to about 1e^{-5} for this example.

Note: If there is noise in the data, set *relerr* and *abserr* above the respective noise levels. Otherwise, Spectre attempts to interpolate the noise, resulting in very high order models and very long simulation times.

Remember that if *relerr* is set to zero, then, from the formula above, pure absolute error is used. Conversely, if *abserr* is set to zero, the error control is based solely on the errors relative to the magnitude of the input data.

Using the ratorder Parameter

It is usually a good idea to specify only the accuracy parameters *relerr* and *abserr*, and let Spectre automatically select the order of the rational approximation. However, if you have special information about your data set, you can direct Spectre to use a specific order of approximation in the rational interpolation. For example, if you know that your tabulated S-parameter data represents a sixth-order filter, you might instruct Spectre to use a seventh or eighth order fit. The slightly higher order gives Spectre flexibility to adjust for any noise or non-ideal behavior in the data.

When you specify the *ratorder* parameter, *relerr* and *abserr* are used to warn you if the order you selected is not sufficient to meet the accuracy requirements. Otherwise, the parameters are not used.

Troubleshooting

Some data sets can cause difficulty for the rational interpolation process. Types of data to avoid are

- Data specified only over a very narrow frequency range.

  Extrapolation of the data outside this range, needed for time-domain simulation, might be difficult and is always risky.

- Very noisy data.

  Such data might lead to large, unreliable time-domain models.

- Data on a very sparse frequency grid.

  Accurate interpolation of such data might be impossible.

- Data with long ideal delays.
- Data representing idealized lossless elements, such as lossless transmission lines.

Assessing the Quality of the Rational Interpolation

If you suspect a problem with the rational interpolation process, you can investigate it using the *sp* analysis with the following steps:

- Construct a test schematic consisting of an *nport* component with interpolation set to rational, as discussed in "Using the nport Components" on page 1374.
- Next add the appropriate number of *port* components to the schematic.

- Perform an *sp* analysis on the *nport* component and look for anomalies.

Large swings in interpolated values and S-parameter magnitudes greater than one suggest a problem. Large changes in the interpolant result from an inaccurate fit, and S-parameter values greater than one result from a non-passive (energy-generating) model that might create unstable time-domain solutions. Be particularly critical of anomalies near zero frequency (DC).

If the anomalous behavior occurs inside the frequency range given in the S-parameter data file, it usually indicates an inaccurate rational interpolation. Be sure all of the conditions given above are met. Try decreasing *relerr* or *abserr* or both, or specifying a higher-order interpolation with the *ratorder* parameter. Remember that measured data can contain fine details that might require a higher order than you might expect from a casual inspection of the data.

If the anomalous behavior occurs outside the tabulated frequency range, try changing the *abserr*, *relerr*, or *ratorder* parameters. Sometimes anomalies can be removed by using a more accurate fit. However, there are limits to the ability to extrapolate outside the frequency interval you specify. You might need to specify additional data points to fix the problem.

Model Reuse

The ROM data file feature of the *nport* component lets you reuse the same S-parameter component description across many designs, but perform the conversion to a time-domain model only once.

For a given S-parameter data set, after you have specified a location for the ROM data file, and Spectre has performed the conversion to a time-domain model and written the time-domain model file to the ROM data file, you can reuse the model file in future simulations. To reuse the model for another *nport*, enter its time-domain model file name in the *ROM data file* field on the new *nport* component's Edit Object Properties form in the new design. At this point, you no longer must specify a raw S-parameter data file.

If you accidently specify both an S-parameter data file and a ROM data file, SpectreRF reuses the model in the ROM data file if the two files are consistent. Otherwise, a new model is generated from the raw tabulated S-parameter data, and the ROM data file is overwritten with a new model. This feature lets you specify both the ROM data file and the S-parameter data file in a design when you first place the *nport* component. SpectreRF then automatically generates the time-domain model for the first simulation and reuses it for all subsequent simulations without needing to change the *nport* component parameters.

If you require a more accurate rational interpolation, then you must regenerate the model in the ROM data file by changing the *relerr*, *abserr* or *ratorder* fields as described in "Controlling Model Accuracy" on page 1376.

References

To learn technical details about how SpectreRF converts S-parameter data to a time-domain description, see the article "Robust rational function approximation algorithm for model generation," by C. P. Coelho, J. R. Phillips, and L. M. Silveira. This article appeared in the proceedings of the 36th Design Automation Conference, New Orleans, LA, June 1999.

The S-Parameter File Format Translator (SPTR)

The S-parameter data file format translator (`sptr`) is a separate program from the Spectre simulator. You can find documentation for `sptr` in the *Spectre User Guide*.

K

Measuring AM, PM and FM Conversion

Derivation

Consider a sinusoid that is simultaneously both amplitude and phase modulated as in Equation K-1.

(K-1) $$v_m(t) = A_c(1 + \alpha(t))\cos(\omega_c t + \phi_c + \phi(t))$$

In Equation K-1, A_c, ϕ_c, ω_c, are the amplitude, phase and angular frequency of the carrier, while $\alpha(t)$, and $\phi(t)$ are the amplitude and phase modulation.

When you assume that $\phi(t)$ is small for all t, this allows the narrowband angle modulation approximation as in Equation K-2. See [ziemer76].

(K-2) $$v_m(t) = A_c(1 + \alpha(t))[\cos(\omega_c t + \phi_c) - \phi(t)\sin(\omega_c t + \phi_c)]$$

Converting to complex exponentials gives Equation K-3.

(K-3) $$v_m(t) = \frac{A_c}{2}(1 + \alpha(t)) \left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + j\phi(t)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right]$$

Letting both the amplitude and phase modulation be complex exponentials with the same frequency, ω_m, gives Equations K-4, K-5, K-6, K-7 and K-8.

(K-4) $$\alpha(t) = Ae^{j\omega_m t}$$

(K-5) $$\phi(t) = \Phi e^{j\omega_m t}$$

Where

(K-6) $$A = A_{A}e^{j\phi_A t}$$

(K-7) $$\Phi = A_{\Phi}e^{j\phi_\Phi t}$$

(K-8) $$v_m(t) = \frac{A_c}{2}\left(1 + Ae^{j\omega_m t}\right)\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + j\Phi e^{j\omega_m t}\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right]$$

Assuming that both A and Φ are small and neglecting cross modulation terms gives Equation K-9.

(K-9) $$v_m(t) = \frac{A_c}{2}\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + Ae^{j\omega_m t}e^{j(\omega_c t + \phi_c)} + Ae^{j\omega_m t}e^{-j(\omega_c t + \phi_c)} + j\Phi e^{j\omega_m t}e^{j(\omega_c t + \phi_c)} - j\Phi e^{j\omega_m t}e^{-j(\omega_c t + \phi_c)}\right]$$

Simplifying gives Equation K-10.

(K-10)
$$v_m(t) = \frac{A_c}{2}\Big[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + Ae^{j((\omega_m + \omega_c)\,t + \phi_c)} + Ae^{j((\omega_m - \omega_c)\,t - \phi_c)} + j\Phi e^{j((\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((\omega_m - \omega_c)\,t - \phi_c)}\Big]$$

In Equation K-10,

- The AM terms are

$$Ae^{j((\omega_m + \omega_c)\,t + \phi_c)} + Ae^{j((\omega_m - \omega_c)\,t - \phi_c)}$$

- The PM terms are

$$j\Phi e^{j((\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((\omega_m - \omega_c)\,t - \phi_c)}$$

Rearranging Equation K-10 gives Equation K-11,

(K-11)
$$v_m(t) = \frac{A_c}{2}\Big[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + Ae^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} + Ae^{j(\omega_m + \omega_c)\,t}e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)\,t}e^{j\phi_c}\Big]$$

In Equation K-11

- The LSB terms are

$$Ae^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)\,t}e^{-j\phi_c}$$

- The USB terms are

$$Ae^{j(\omega_m + \omega_c)t} e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)t} e^{j\phi_c}$$

Assume that you perform a PAC analysis, which applies a single complex exponential signal that generates responses at the upper and lower sidebands of the ω_c signal. Assume the transfer functions are L and U, so the lower and upper sideband signals are given by Equations K-12 and K-13.

(K-12) $$l(t) = Le^{j(\omega_m - \omega_c)t}$$

(K-13) $$u(t) = Ue^{j(\omega_m + \omega_c)t}$$

Where

$$L = A_L e^{j\phi_L}$$

$$U = A_U e^{j\phi_U}$$

Matching common frequency terms between Equations K-11, K-12, and K-13 gives Equations K-14, K-15, K-16 and K-17.

(K-14) $$L = \frac{A_c}{2}\left(Ae^{-j\phi_c} - j\Phi e^{-j\phi_c}\right)$$

(K-15) $$U = \frac{A_c}{2}\left(Ae^{j\phi_c} + j\Phi e^{j\phi_c}\right)$$

(K-16) $$\frac{2}{A_c} Le^{j\phi_c} = A - j\Phi$$

(K-17) $$\frac{2}{A_c}Ue^{-j\phi_c} = A + j\Phi$$

Solving for the modulation coefficients gives Equations K-18 and K-19.

(K-18) $$A = \frac{1}{A_c}\left(Le^{j\phi_c} + Ue^{-j\phi_c}\right)$$

(K-19) $$\Phi = \frac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)$$

Thus, Equation K-18 gives the transfer function for amplitude modulation and Equation K-19 gives the transfer function for phase modulation.

Positive Frequencies

Notice that L is defined in Equation K-12 on page 1384 to be the transfer function from the input to the sideband at $\omega_m - \omega_c$, which is a negative frequency. This is usually a natural definition for use with SpectreRF's small signal analyses (depending on the setting of the `freqaxis` parameter). It can be cumbersome though when the only data available is at positive frequencies. Thus, the transfer function to $\omega_c - \omega_m$ is defined as

$$\tilde{L}$$

Then, as in Equation K-20,

(K-20) $$\tilde{l}(t) = \tilde{L}e^{j(\omega_c - \omega_m)t}$$

Since the signals are real, L is a complex conjugate of

$$\tilde{L}$$

And the reverse is also true, as in Equation K-21,

(K-21) $$L = \tilde{L}^*$$

Equations K-22 and K-23 are produced by rewriting Equation K-18 on page 1385 and Equation K-19 on page 1385 in terms of

$\tilde{L}$

(K-22) $$A = \frac{1}{A_c}\left(\tilde{L}^* e^{j\phi_c} + U e^{-j\phi_c}\right)$$

(K-23) $$\Phi = \frac{j}{A_c}\left(\tilde{L}^* e^{j\phi_c} - U e^{-j\phi_c}\right)$$

FM Modulation

For FM modulation, the phase modulation $\phi(t)$ becomes the integral of the FM modulation signal, $\omega(t)$ as shown in Equation K-24.

(K-24) $$v_m(t) = A_c \cos(\omega_c t + \phi(t))$$

Where

(K-25) $$\phi(t) = \int \omega(t) dt$$

Recall from Equation K-5 on page 1382 and Equation K-19 on page 1385 that

(K-26) $$\phi(t) = \frac{j}{A_c}\left(L e^{j\phi_c} - U e^{-j\phi_c}\right) e^{j\omega_m t}$$

Combining Equation K-25 and Equation K-26 and the differentiating both sides results in Equations K-27, K-28, and K-29.

(K-27) $$\omega(t) = \frac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)e^{j\omega_m t}$$

(K-28) $$\Omega = A_\Omega e^{j\phi_\Omega} = \frac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)$$

Or

(K-29) $$\Omega = j\omega_m\Phi$$

Simulation

The test circuit, represented by the two netlists shown in Examples K-1 and K-2, was run with SpectreRF. The test circuit consists of three, linear, periodically-varying modulators that are driven with the same input. The input is constant valued in the large signal PSS analysis, and generates a single complex exponential analysis during the PAC analysis. The idea is to compute the transfer functions from this input to the upper and lower sidebands at the output of the modulators and then use the derivation just described to convert these transfer functions into transfer functions to the AM, PM, and FM modulations and then check the simulation results against the expected results.

Notice that `freqaxis=out`. This is necessary to match the derivation. If you would rather use `freqaxis=absout`, you would have to use the complex conjugate of L as in Equation K-22 on page 1386 and Equation K-23 on page 1386.

Example K-1 Netlist for the AM, PM and FM Conversion Test Circuit

```
// AM, PM, and FM modulation test circuit

simulator lang=spectre
ahdl_include "modulators.va"

parameters MOD_FREQ=10MHz
parameters CARRIER_FREQ=1GHz
```

```
Vin (in 0) vsource pacmag=1 pacphase=0
Mod0 (unmod in) AMmodulator freq=CARRIER_FREQ mod_index=0
Mod1 (am in) AMmodulator freq=CARRIER_FREQ mod_index=1
Mod2 (pm in) PMmodulator freq=CARRIER_FREQ kp=1
Mod3 (fm in) FMmodulator freq=CARRIER_FREQ fd=MOD_FREQ

waves pss fund=CARRIER_FREQ outputtype=all tstab=2ns harms=1
xfer pac start=MOD_FREQ maxsideband=4 freqaxis=out
```

The netlist for the modulator models shown in Example K-2, has the filename `modulators.va`.

Example K-2 Netlist for the Modulator Models Written in Verilog-A

```
`include "discipline.h"
`include "constants.h"

module AMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real mod_index = 1;

    analog begin
        V(out) <+ (1+mod_index*V(in)) * cos(2*`M_PI*freq*$abstime);
        $bound_step( 0.05 / freq );
    end
endmodule

module PMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real kp = 1 from (0:inf);

    analog begin
        V(out) <+ cos(2*`M_PI*freq*$abstime + kp*V(in));
        $bound_step( 0.05 / freq );
    end
endmodule

module FMmodulator (out, in);
    input in;
    output out;
    electrical out, in;
    parameter real freq = 1 from (0:inf);
    parameter real fd = 1 from (0:inf);
    real phi;

    analog begin
        V(out) <+ cos(2*`M_PI*(freq*$abstime + idtmod(fd*V(in),0,1, -0.5)));
        $bound_step( 0.05 / freq );
    end
endmodule
```

Results

The simulations were run with various values for `pacphase` on `Vin`.

- Table K-1 shows results for the output of the AM modulator with $v_{LO} = cos(\omega_c t)$.
- Table K-2 shows results for the output of the PM modulator with $v_{LO} = cos(\omega_c t)$.

Table K-1 Results for the AM Modulator Output

pacphase	L	U	A	Φ
0	1/2	1/2	1	0
45	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2}$	0
90	$j/2$	$j/2$	j	0
180	–1/2	–1/2	–1	0

Table K-2 Results for the PM Modulator Output

pacphase	L	U	A	Φ
0	–1/2	1/2	0	1
45	$\frac{1-j}{2\sqrt{2}}$	$\frac{j-1}{2\sqrt{2}}$	0	$\frac{1+j}{\sqrt{2}}$
90	1/2	–1/2	0	j
180	$j/2$	$-j/2$	0	–1

If you repeat the simulations but replace the *cos* function in the modulators with the *sin* function, which is equivalent to changing the `LO` to $v_{LO} = sin(\omega_c t)$ or setting $\phi_c = -90$, you achieve the following results.

- Table K-3 shows results for the output of the AM modulator with $v_{LO} = sin(\omega_c t)$.

- Table K-4 shows results for the output of the PM modulator with $v_{LO} = sin(\omega_c t)$.

Table K-3 Results for the AM Modulator Output

pacphase	L	U	A	Φ
0	$j/2$	–1/2	1	0
45	$\frac{j-1}{2\sqrt{2}}$	$\frac{1-j}{2\sqrt{2}}$	$\frac{1+j}{\sqrt{2}}$	0
90	–1/2	1/2	j	0
180	$-j/2$	$j/2$	–1	0

Table K-4 Results for the PM Modulator Output

pacphase	L	U	A	Φ
0	1/2	1/2	0	1
45	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2\sqrt{2}}$	0	$\frac{1+j}{\sqrt{2}}$
90	$j/2$	$j/2$	0	j
180	–1/2	–1/2	0	–1

Finally, Table K-5 shows the results for the FM modulator with $v_{LO} = cos(\omega_c t)$. The FM modulator has a modulation coefficient of ω_m built-in, which renormalizes the results.

Table K-5 Results for the FM Modulator Output

pacphase	L	U	Ω
0	–1/2	1/2	1
45	$-\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{2\sqrt{2}}$	$\frac{1+j}{\sqrt{2}}$
90	$-j/2$	$j/2$	j
180	1/2	–1/2	–1

Conclusion

This appendix shows that the PAC analysis can be used to determine the level of AM or PM modulation that appears on a carrier. This is done by applying a small signal and using the phase of the carrier along with the transfer function to the upper and lower sidebands of the carrier to compute an AM or PM transfer function.

References

[Ziemer 76] R. Ziemer and W. Tranter. *Principles of Communications: Systems, Modulation, and Noise.* Houghton Miffin, 1976.

[Robins 96] W. Robins. *Phase Noise in Signal Sources (Theory and Application)*. IEE Telecommunications Series, 1996.

L

Using PSP and Pnoise Analyses

Overview

This appendix describes how to calculate small-signal quantities such as noise, noise figure, periodic scattering parameters, and gain in periodically-driven circuits. The appendix explains

- The concepts of periodic S-parameters,
- The concepts of noise correlation parameters
- The various definitions of noise figure and gain

The Spectre simulator provides four small-signal analyses for circuits with a DC operating point: AC, XF, Noise and SP. The SpectreRF simulator also provides four small-signal analyses for circuits with a a periodically time-varying operating point: PAC, PXF, Pnoise and PSP. Because the periodic small-signal analyses linearize the circuit about the time-varying operating point that is obtained using the PSS analysis, they can analyze frequency conversion effects.

- PAC analysis computes the small-signal response at all outputs to the small stimulus of a single group of sources.
- PXF analysis computes the transfer function from every source in the circuit to a single output.
- Pnoise analysis computes noise parameters such as noise figure as well as detailing noise contributions by devices.
- PSP analysis contains some of the capabilities of PAC, PXF, and Pnoise analyses. PSP analysis can compute periodic scattering parameters that describe the small-signal relations between several different ports in a circuit. It can also compute noise parameters, such as noise correlation matrices, equivalent noise sources, and noise figure.

Periodic S-parameters

Linear Time-Invariant S-Parameters

Designers of microwave and RF circuits typically characterize the frequency-dependent behavior of linear networks through sets of *scattering* or *S*-parameters. The notion of scattering parameters is rooted in transmission line concepts where the scattering parameter matrix relates the magnitude and phase of incident and reflected waves.

Consider an arbitrary *N*-port linear time-invariant (LTI) network, such as the two-port shown in Figure L-1 on page 1395. Each port is driven by a source of reference impedance Z_i, where the index i runs from 1 to *N*. For the remainder of this document we will assume a real valued reference impedance, R_i, for each port. In terms of the port currents I_i and voltages V_i, the *incident* and *reflected* quantities, a_i and b_i respectively, are defined for each port.

The *incident* quantity, a_i as

$$a_i = \frac{v_i}{2\sqrt{R_i}} + \frac{\sqrt{R_i}}{2} I_i$$

The *reflected* quantity, b_i as

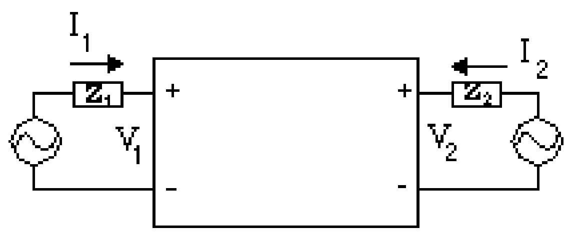
$$b_i = \frac{v_i}{2\sqrt{R_i}} - \frac{\sqrt{R_i}}{2} I_i$$

The frequency-dependent S-parameter matrix relates a and b by

$$b(\omega) = S(\omega)a(\omega)$$

These definitions are possible because for LTI systems, sources at a frequency ω generate steady-state responses, and therefore outputs, at the same frequency.

Figure L-1 Two-Port Linear Network



Frequency Translating S-Parameters

The SpectreRF small-signal analyses treat the circuit as linear time-varying (LTV). The primary difference between linear time-varying networks, those that come from circuits with a time-varying operating point, and LTI networks, those that come from circuits with a DC operating point, is that LTV networks shift signals in frequency.

For periodically linear time-varying (PLTV) systems, inputs at a frequency ω may generate circuit responses, and therefore outputs, at the frequencies $\omega + n\omega_o$, where ω_o is the fundamental frequency and n is a (signed) integer. We can adopt the S-parameter concept to PLTV systems by considering the inputs and outputs generated at the sidebands of each harmonic to be *virtual ports* of a generalized linear system.

That is, we can define $a_{i,n}$ by

$$a_{i,n}(\omega) = \frac{v_i(\omega + n\omega_o)}{2\sqrt{R_i}} + \frac{\sqrt{R_i}}{2} I_i(\omega + n\omega_o)$$

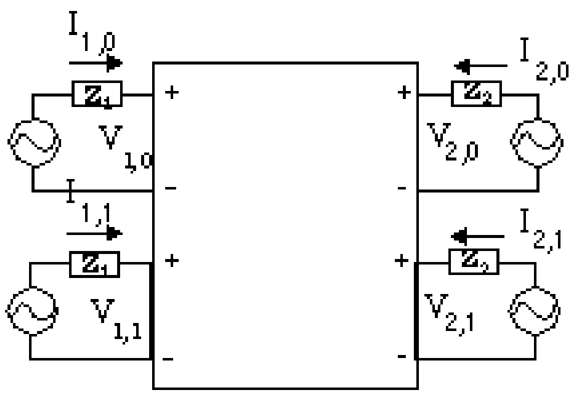
And we can define $b_{i,n}$ by

$$b_{i,n}(\omega) = \frac{v_i(\omega + n\omega_o)}{2\sqrt{R_i}} - \frac{\sqrt{R_i}}{2} I_i(\omega + n\omega_o)$$

where the integer n represents an harmonic index.

For example, as shown in Figure L-2 on page 1396, an ideal mixer may be represented as a four-port.

Figure L-2 Ideal Mixer Represented as a Four-Port



Note that each *virtual port* of a given *physical port* has the same reference impedance. The periodic S-parameter matrix is the 4×4 matrix $\tilde{s}$ that relates the extended vectors

$$\tilde{b} = \begin{bmatrix} b_{1,0} & b_{1,1} & b_{2,0} & b_{2,1} \end{bmatrix}^T, \tilde{a} = \begin{bmatrix} a_{1,0} & a_{1,1} & a_{2,0} & a_{2,1} \end{bmatrix}^T$$

by

$$\tilde{b}(\omega) = \tilde{S}(\omega)\tilde{a}(\omega)$$

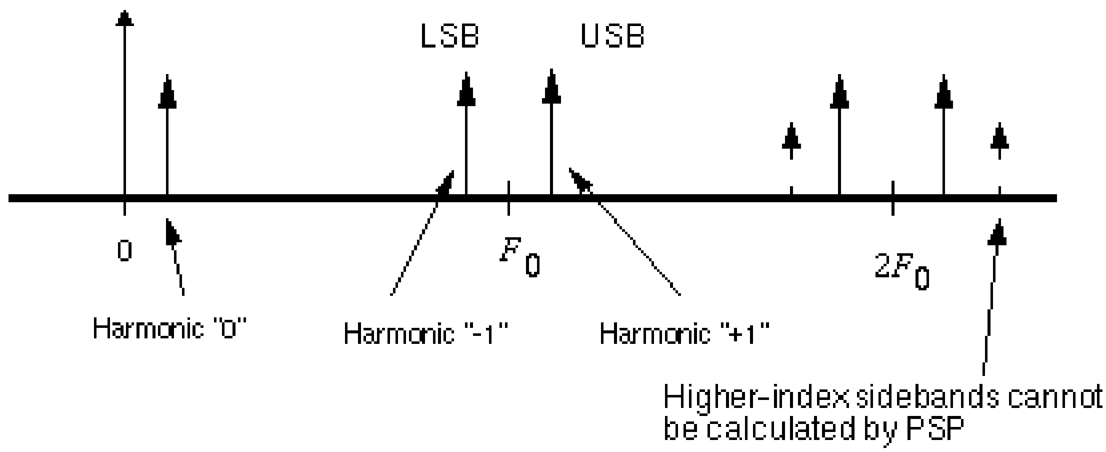
For example, consider an upconverting mixer. You might write $S(2,1|1,0)$ to represent the signal generated at port #2 (typically the output) on the upper sideband of harmonic +1 by an incident signal at port #1 (typically the input) at harmonic zero (baseband). $[S(2,1|1,0)\ (\omega)]^2$ would represent the power gain from baseband to RF at the baseband-referenced frequency ω. In the PSP results generated by spectre, $S(2,1|1,0)$ is accessible as S21~1:0.

Note that because *multiple virtual* ports are used as both inputs and outputs in PSP analysis, PSP analysis must follow an absolute indexing scheme for the small-signal responses. This is different from the relative indexing scheme used in PXF, PAC, and PSP analyses in releases 4.4.5 and earlier. See “Harmonics and Sidebands in PSP, PAC, PXF, and Pnoise Analyses” on page 1429 for a discussion of the differences. In future releases, all RF analyses will support absolute indexing of the signals.

Upper and Lower Sidebands

Each harmonic may have an input, or response, at both the upper and lower sidebands of each harmonic. In PSP analysis, the upper sideband is denoted by a positive integer, and the lower sideband as a negative integer. See Figure L-3 on page 1397.

Figure L-3 Harmonics, Sidebands, and Virtual Ports



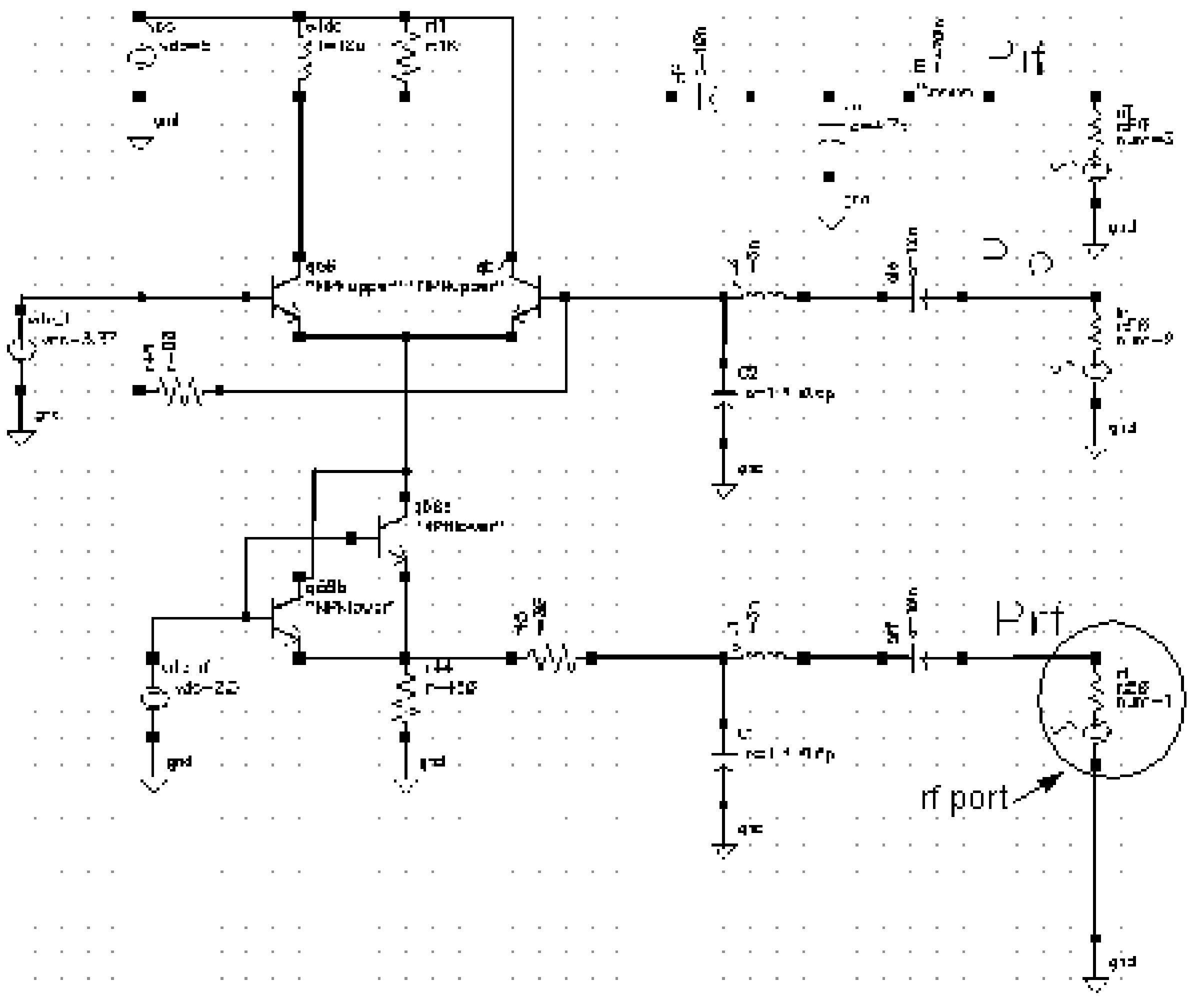
For a small-signal frequency ΔF, the upper sideband of the k^{th} harmonic is at the frequency $|k|\ F_0+\Delta F$ and the lower sideband is at $|k|\ F_0 - \Delta F$.

In general, an input (perhaps at baseband at the frequency ΔF in an upconversion mixer) can generate responses at all the frequencies $|k\ F_0+l\Delta F|$, for k and l integers. However, since small-signal analyses are linear, they can only calculate the signals at the first sideband. Thus we only need notation for the l =1 terms in PSP analysis.

PSP Analysis Example

Consider performing a PSP analysis on the *NE600* mixer schematic from the *rfExamples* library. The schematic is shown in Figure L-4 on page 1398.

Figure L-4 NE600 Mixer



Suppose the RF input signal is at 900 MHz, the LO at 1 GHz, and the IF at 100 MHz. Before the PSP analysis is performed, a PSS analysis must be run. For small-signal analysis, in many cases it would be sufficient to treat the RF input as small-signal (for example, by setting the *source type* to *DC*). However, sometimes it is important to analyze additional noise folding terms induced by the RF input, so in this example we assume the RF source is a large signal (e.g., *source type = sine*). The PSS fundamental will need to be set to 100 MHz.

Now for the sake of demonstration suppose we wish to perform the small-signal analysis from 20 MHz below the RF center frequency to 30 MHz above. To set up the analysis, we first select a *frequency sweep*. We select *sweeptype=relative*, with a range of -20 MHz to 30 MHz. This will account for inputs on the RF port in the range of 880 MHz to 930 MHz. Noise parameters such as *noise figure* will be computed in a 50 MHz band around the frequency specified by the output harmonic.

Next we select the ports and harmonics. The *input* and *output ports* are selected from the schematic as always. Selecting the harmonics is somewhat trickier.

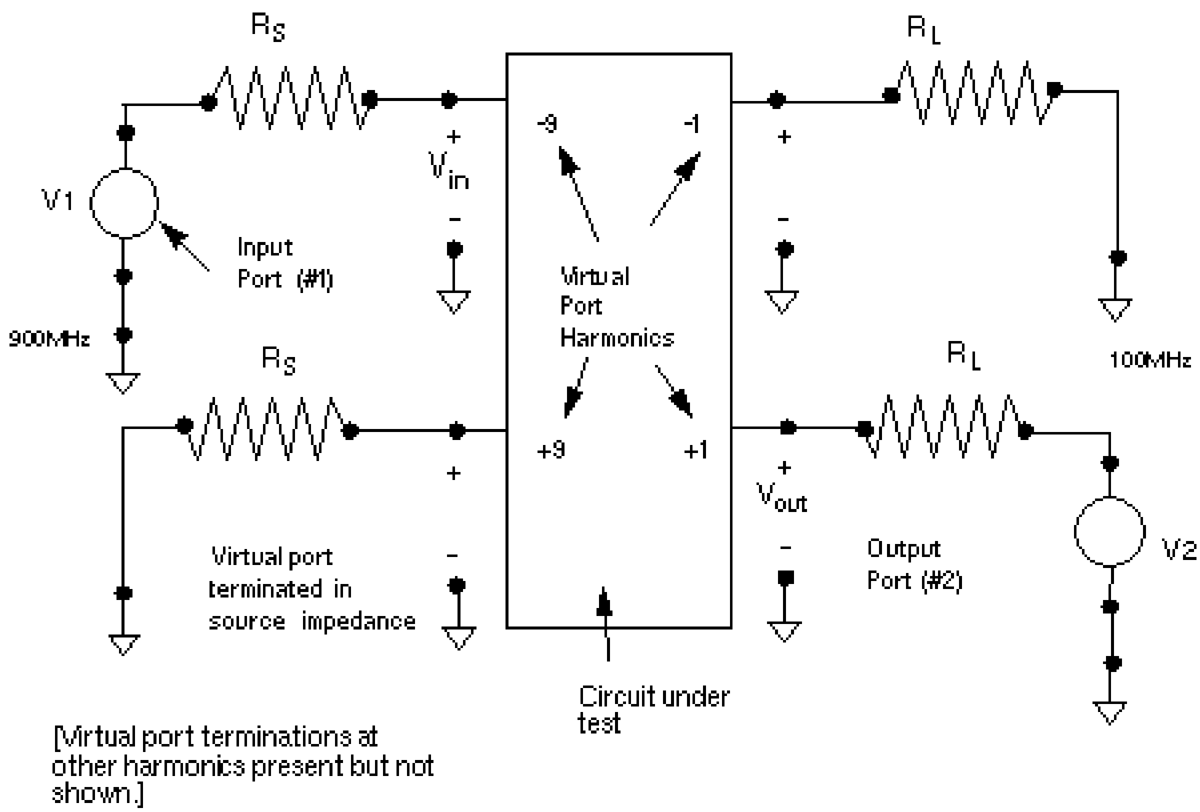
With the 1 GHz LO, small-signal inputs at around 900 MHz, or harmonics +/–9 of the PSS fundamental, will appear at around 100 MHz, or harmonics –/+1 of the fundamental. We may select harmonic 1 as the output harmonic and harmonic -9 as the input harmonic because a single complex-exponential input

$$e^{i\omega_s t}$$

input on the lower side of 900 MHz (harmonic -9) will appear as the upper sideband of harmonic 1 (around 100 MHz). -9 and 1 are separated by 10 fundamental periods, which corresponds to the LO frequency of 1 GHz. Figure L-5 on page 1400 conceptually illustrates this setup.

Once the simulation is complete, a limited set of data is accessible through direct plot, and the full data set is accessible with the results browser. The default is to output all quantities versus the input frequency, which in this case would be a sweep from -920 MHz to -870 MHz, because the input harmonic is -9 and the sweep ran from -20 Mhz to 30 MHz. The *freqaxis* parameter on the options form may be used to change the axes that are output by Spectre. In 4.4.5 the *freqaxis* parameter and *frequency sweep* specifications are solely responsible for the data’s axis generation. Setting *freqaxis=out* in this example would produce an axis running from 80 Mhz to 130 Mhz.

Figure L-5 Measuring Periodic S-Parameters



An additional port/harmonic pair can be included in the PSP analysis by using the *auxiliary port* fields.

If it is desirable to include more than three harmonics in the PSP analysis, they can be added to the list in the form below the *input./output/auxiliary*. For example, to examine additional images in the PSP analysis, +9 and -1 could be additional harmonics. S21~9:-1 would represent the transducer gain from RF-USB to IF-LSB. Note that S21:9:1 is likely to be small, because this term represents a frequency shift of 800 MHz. There are no elements in the circuit that vary at 800 MHz, so significant 800 MHz frequency translations will not be present.

Noise and Noise Parameters

Calculating Noise in Linear Time-Invariant (DC Bias) Circuits

The standard Noise analysis has two parts.

- First, the circuit is analyzed without the noise sources present in order to find a DC operating point.
- Next, the circuit is linearized around that operating point and the noise sources are turned on.

The linearized circuit is used to compute a set of transfer functions that represent the gain from each noise source to the node pair or probe that is identified as the output for the Noise analysis. All noise generators present in the circuit are automatically included in the Noise analysis, as are the noise sources of the source and load.

Calculating Noise in Time-Varying (Periodic Bias) Circuits

Noise analysis in RF circuits, where the circuit operating point is time-varying, is computationally involved, but conceptually similar to Noise analysis for circuits with a DC bias.

- Find the circuit operating point using the PSS analysis.
- Linearize the circuit around that operating point.
- Use the PXF analysis, based on the linearized circuit, to compute a set of transfer functions that relate the noise sources to the noise at the circuit output.

The treatment of the sources and the transfer functions is more complicated for RF circuits because of the time-varying operating point.

For noise sources that are bias dependent, such as shot noise sources, the time-varying operating point acts to modulate the noise sources. Active elements with a time-varying bias point can convert noise from one frequency to another, a process known as *noise folding*, regardless of the origin of the noise.

Because of these effects, noise generated in RF circuits usually has *cyclostationary* properties. Cyclostationary random processes are processes whose statistical properties are periodically time-varying. In the frequency domain, a simple way to think of cyclostationarity is as frequency-correlation. For example, noise at the input of a mixer will appear on the output, but shifted in frequency. Thus the noise at the mixer output at a given frequency will be correlated with noise at the mixer input at a frequency separated by the frequency of the local oscillator. In contrast, the noise generated by a circuit with a DC bias point is usually modeled as being uncorrelated with noise at any other frequency. SpectreRF correctly accounts for cyclostationary statistics when calculating noise, noise figure, noise correlation parameters, and equivalent noise sources.

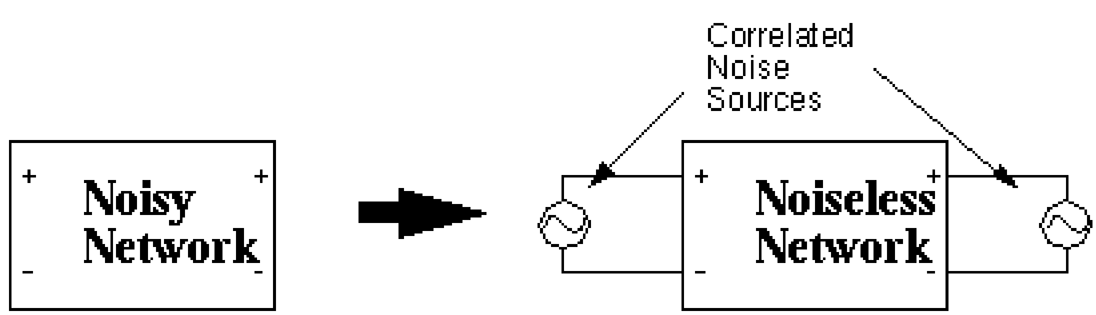
The maxsideband Parameter

All the noise computations in PSP involve noise folding effects. The *maxsideband* parameter specifies the maximum sideband included for summing noise contributions either up-converted or down-converted to the output at the frequency of interest. The contribution of the noise source to the output is modulated by the periodic transfer function. Modulation with a periodic transfer function is convolution with the discrete spectrum of the transfer function. *Maxsideband* specifies the number of sidebands to be involved in this calculation.

Noise Correlation Matrices and Equivalent Noise Sources

Noise correlation matrices represent a decomposition of a linear circuit into a noiseless linear network and correlated noise sources. For example in Figure L-6 on page 1402, a noisy linear network is decomposed into a noiseless network and equivalent noise current sources, one for each circuit port, that represent the effect of all the noise generators internal to the original network. Note that in general the equivalent noise sources will be correlated. The equivalent sources can be completely described by a (frequency-dependent) noise correlation matrix. Note that once a noise correlation matrix along with an admittance, impedance, or S-parameter matrix are known, then the properties of the noisy linear network as seen from the I/O ports is completely specified. All circuit input/output properties—gain, noise figure, etc.—can be calculated from the S-parameter and noise correlation matrices.

Figure L-6 Decomposing a Network Into Noiseless and Noisy Elements.



The SP analysis computes noise correlation parameters in the admittance representation, i.e., the sources in Figure L-6 on page 1402 are current sources. If we let I_N denote the vector of equivalent noise currents as

$$I_N = \begin{bmatrix} I_{N1} \\ I_{N2} \end{bmatrix}$$

Then the noise correlation matrixes C_y11 and C_y12 are defined as

$$C_Y 1x = \left[\frac{1}{4k \times T0\_1 \times df}\right] \times E\left\{I_N I_N^H\right\}$$

And the noise correlation matrixes C_y21 and C_y22 are defined as

$$C_Y 2x = \left[\frac{1}{4k \times T0\_2 \times df}\right] \times E\left\{I_N I_N^H\right\}$$

where

k is Boltzmann's constant

$T0\_1$ is the noise temperature of the input port

$T0\_2$ is the noise temperature of the output port

df is noise bandwidth

superscript H (I^{H}) denotes the Hermitian transpose

$E\{°\}$ denotes statistical expectation.

For example, for a two-port,

$$C_Y = \left[\frac{1}{4k \times T0\_1 \times df}\right] \times E\left\{\begin{bmatrix} I_{N1}\overline{I_{N1}} & I_{N1}\overline{I_{N2}} \\ I_{N2}\overline{I_{N1}} & I_{N2}\overline{I_{N2}} \end{bmatrix}\right\}$$

where the overbar signifies complex-conjugate. Note that this matrix must have real diagonal elements, because the diagonals represent a total noise power, but the off-diagonals, which represent the correlations, may be complex.

In the periodic case, we define the periodic admittance noise correlation matrix in precisely the same way. The situation is slightly more complicated because the *virtual ports* may lie at different frequencies in RF systems.

Letting

$$\tilde{I}_N$$

denote the extended vectors of noise currents at each of the virtual ports (each virtual port consisting of a physical port. harmonic pair), for example

$$\tilde{I}_N = \begin{bmatrix} I_{1,0} & I_{1,1} & I_{2,0} & I_{2,1} \end{bmatrix}^T$$

where the first index indexes the physical port, and the second index specifies the harmonics.

The periodic admittance noise correlation matrix

$$\tilde{C}_Y$$

is defined as

$$\tilde{C}_Y = \left[\frac{1}{4k \times T0\_1 \times df}\right] \times E\left\{\tilde{I}_N \tilde{I}_N^H\right\}$$

When you specify the PSP analysis option *donoise=yes*, then the complex noise correlation matrix of order (#active ports X #active sidebands) is computed.

Two-Port Noise Parameters

As an alternative to the noise correlation matrices that define the equivalent sources, Spectre calculates the values of the equivalent noise parameters F_{min}, R_n, G_{opt}, $B_{opt,}$ and NF_{min}. These are calculated from the two-port admittance (Y_{11}, Y_{12}, Y_{21}, Y_{22}) and noise correlation admittance (CY_{11}, CY_{12}, CY_{21}, CY_{22}) parameters. These calculations are done as part of the

SP and PSP analysis. In terms of the admittance and noise correlation matrices, the parameters are

```
Fmin = 1 + 2 CY22/|Y21| 2 (Gopt + Re{Y11 - Y21(CY12/CY22)})
Gopt = sqrt[|Y21|2 (CY11/CY22) - |Y21|2(|CY12|2/CY222) + (Re{Y11 - Y21(C12/C22)})2]
Bopt = -Im{Y11 - Y21(CY12/CY22)}
Rn = CY22/|Y21|2
Yopt = Gopt + jBopt
NFmin = 10log(Fmin)
```

Y_{opt} is the source admittance that gives the minimal noise factor F_{min} (corresponding to the source reflection coefficient Gamma opt, or Γ_{opt}) and R_n is the equivalent noise resistance. Finally, NF_{min}, the minimum noise figure, is F_{min}, the minimum noise factor, in dB.

For more information, see Janusz A. Dobrowolski, *Introduction to Computer Methods for Microwave Circuit Analysis and Design, Artech House, Boston, 1991*, page 193.

Noise Circles

Noise factor will be a function of the source admittance $Y_s = G_s + jB_s$. For a given Y_s the noise factor is

$$F = F_{min} + \frac{R_n}{G_s}(Y_s - Y_{opt})^2$$

Varying Y_s traces out circles of constant noise factor F. In the 4.4.5 release, noise circles are only available in direct plot for the SP analysis.

Noise Figure

Performing Noise Figure Computations

The Noise, SP, Pnoise and PSP analyses all provide the ability to calculate various types of noise figure. Noise and SP analyses are used for circuits with a DC bias. Pnoise and PSP analyses are used for circuits with a periodic bias. The *generic* way Spectre calculates noise figure is by first computing the noise factor F,

$$F = \frac{totalOutputNoise - outputNoiseFromLoad}{outputNoiseFromSource}$$

where the noise is specified in units of power (e.g., V^2/Hz). Noise figure is then $NF = 10 log_{10} F$.

The various definitions of noise figure differ in the following ways

- How the contributions to the total output noise are calculated (e.g. Pnoise analysis has noise folding effects, Noise analysis does not)
- What noise is considered to be due to the load
- What noise is considered to be due to the source

All the analyses share some common rules that must be followed to obtain correct answers

- You must specify a *port* (not a *vsource*, *isource*, or *ahdl* source) as an circuit *input* or *iprobe*.
- You must specify the *load* as an *output* or *oprobe*. The *load* may be a *resistor* or a *port*. Note that all noise from the source will be included in the denominator of the noise factor fraction, including excess noise, so do not specify excess noise on the input port. (Excess noise is specified with the *`noisefile`* or *`noisevec`* option.)

In rare cases there may be no load, in which case you can specify the output using a pair of nodes. Be warned, however, that if there is a load in the circuit, and a pair of nodes is specified as an output, then you will obtain different results than if a load was specified as output. This is because the load will contribute some noise to the total output noise that must be subtracted out before using the equation above to compute the noise figure. If only a pair of nodes is used, Spectre has no way of determining which of the elements in the circuit is the load (there could be multiple resistors connected to the output nodes, for example) and so cannot determine the amount of output noise due to the load.

Note that these requirements are automatically enforced in SP and PSP analyses, since the input and output sources are always ports that must be identified to the analyses.

Noise Figure From Noise and SP Analyses

The Noise and SP analyses perform noise figure computations on circuits with a DC operating point. The above prescription for noise figure computation is straightforward: the output noise, contribution from source, and contribution from load must be computed. Mathematically, if we let X_L denote the transfer function from output load to output, (at the same frequency) and X_S denote the transfer function from input source to output then

$$F(f) = \frac{N_o(f) - \left|X_L\right|^2 n_L(f)}{\left|X_s\right|^2 n_s(f)}$$

In this equation, X_S plays a role similar to transducer gain in traditional treatments of noise figure.

Pnoise (SSB) Noise Figure

Because noise in an RF circuit can originate at many different frequencies, the denominator in the noise factor computation is in a sense ill-defined. See the book *Microwave Mixers* by S. Maas for a discussion of various possible noise figure definitions.

There are three common noise factor definitions in use. The SpectreRF Pnoise and PSP analyses compute as F or NF what is referred to as conventional single-sideband (SSB) noise figure. The conventional SSB noise figure is typically useful for heterodyne receivers. To compute the conventional SSB noise figure, a reference sideband must be specified that identifies the input noise used in the denominator of the noise factor computation. Only the contribution of the noise from the input source, generated at the frequency specified by the reference sideband, is included in the noise factor denominator.

In the Pnoise context, the numerator contains the total output noise, except the noise from the output load that was generated at the output frequency. Note in particular that noise from the input source folded from all sidebands, and noise from the output load folded from the non-zero sidebands, is included in the noise factor numerator. Mathematically, Pnoise computes conventional single-sideband noise factor as

$$F_{ssb}(f_{out}) = \frac{N_o(f_{out}) - \left|X_L^{(0)}\right|^2 n_L(f_{out})}{\left|X_S^{(K_{ref})}\right|^2 n_S(f_{out} + K_{ref} f_0)}$$

where

- f_{out} is the output frequency swept by Pnoise
- $X_L^{(0)}$ is the transfer function associated with the zero sideband, from load to output
- $X_s^{(K_{ref})}$ is the transfer function associated with the reference sideband, from source to the output
- $n_L(f_{out})$ is noise generated by load at the output frequency
- $n_S\,(f_{out}+K_{ref}\,f_0)$ is the noise generated by the source at the input frequency.

In the PSP analysis context, the noise factor denominator includes only noise from the input harmonic. The numerator contains all output noise, except noise from the output load at the output harmonic. (Refer to “Harmonics and Sidebands in PSP, PAC, PXF, and Pnoise

Analyses" on page 1429 for a discussion of differences in frequency indexing in PXF and PSP analyses.)

To be mathematically precise in what follows, let

- $x_L^{(K)}$ denote the transfer function from output load, k^{th} harmonic to output
- $x_S^{(K)}$ denote the transfer function from input source, k^{th} harmonic to output
- K_o denote the output harmonic
- K_i denote the input harmonic

The conventional single-sideband noise factor is computed by PSP as

$$F_{ssb}(f_{out}) = \frac{N_o(K_o f_0 + f) - \left|X_L^{(K_o)}\right|^2 n_L(K_o f_0 + f)}{\left|X_S^{(K_i)}\right|^2 n_S(K_i f_0 + f)}$$

where

- f is the PSP relative sweep frequency
- $K_o f_0 + f$ is the output frequency
- $K_i f_0 + f$ is the input frequency
- $n_L(K_o f_0 + f)$ is the noise generated by the load at the output frequency
- $n_S(K_i f_0 + f)$ is the noise generated by the source at the input frequency

The conventional SSB noise figures computed by PSP and Pnoise analyses are the same and are computed in the same way internally, it is only the notation above that is different.

DSB Noise Figure

In some applications, such as direct conversion receivers, it is more appropriate to compute what is called double-sideband (DSB) noise figure. Double-sideband noise factor is obtained by ratioing the same numerator as for SSB to the noise from the input at the input harmonic as well as its primary image.

Double-sideband noise figure is usually 3 dB below single-sideband noise figure, except when the input signal band is converted to baseband output with DSB noise figure equal to SSB noise figure.

In double-sideband computation, the input signal band is assumed to be either down-converted or up-converted to the output signal band. Hence you should associate the appropriate harmonic number to the input and the output port in the *portharmsvec* parameter. For a mixer, their difference is the LO band. The image sideband is the sideband on the other side of the LO band. The distance from the image band to the LO band is the same as that from the LO band to the input band. For example, if the fundamental frequency is100 MHz, the LO frequency is 1 GHz, and the RF input frequency is 900 MHz, then the LO band is 10, the RF input band is 9 and the image band is 11.

Double-sideband noise figure is computed by as

$$F_{dsb}(f_{out}) = \frac{N_o(K_o f_0 + f) - \left|X_L^{(K_o)}\right|^2 n_L(K_o f_0 + f)}{\left|X_S^{(K_i)}\right|^2 n_S(K_i f_0 + f) + \left|X_S^{(K_{\mathrm{image}})}\right|^2 n_S(K_{\mathrm{image}} f_0 + f)}$$

where

$n_S(K_{\mathrm{image}} f_0 + f)$ is the noise generated by source at the image input frequency obtained according to the above description.

All other quantities are the same as those in F_{ssb}. Note that both the PSP and Pnoise analyses compute double-sideband noise figure.

IEEE Noise Figure

Sometimes it is desirable to define noise figure quantities where we assume that the noise from input images that are potentially filtered is not present. The IEEE definition of noise figure in mixers differs from the conventional definition in that it does not include the contribution to the output noise from the image sideband in the numerator of the noise factor. Spectre eliminates all image harmonics/sidebands from the output noise in the noise factor numerator when computing *Fieee*. Using the above notation, *Fieee* is

$$F_{ieee} = \frac{N_o(K_o f_0 + f) - \left|X_L^{(K_o)}\right|^2 n_L(K_o f_0 + f) - \sum_{K \neq K_i} \left|X_S^{(K)}\right|^2 n_S(K f_0 + f)}{\left|X_S^{(K_i)}\right|^2 n_S(K_i f_0 + f)}$$

Note that both the PSP and Pnoise analyses compute *Fieee*.

Figure L-7 on page 1410, Figure L-8 on page 1411, and Figure L-9 on page 1412 summarize the treatment of the input source and output load for the various noise figure definitions.

Figure L-7 Input Source Treatment for Denominator in Noise Factor Computations

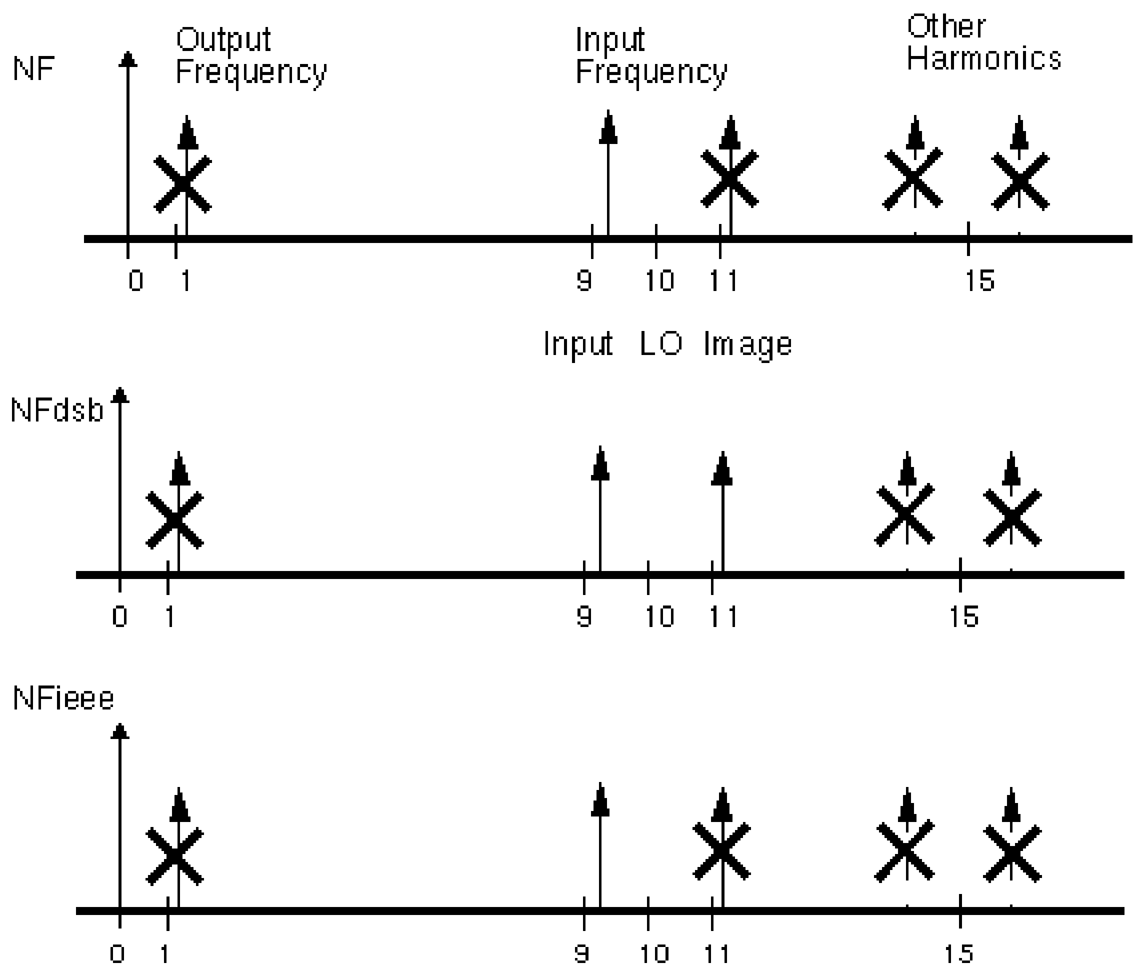


Figure L-8 Output Load Treatment for Numerator in Noise Factor and Noise Figure Computations

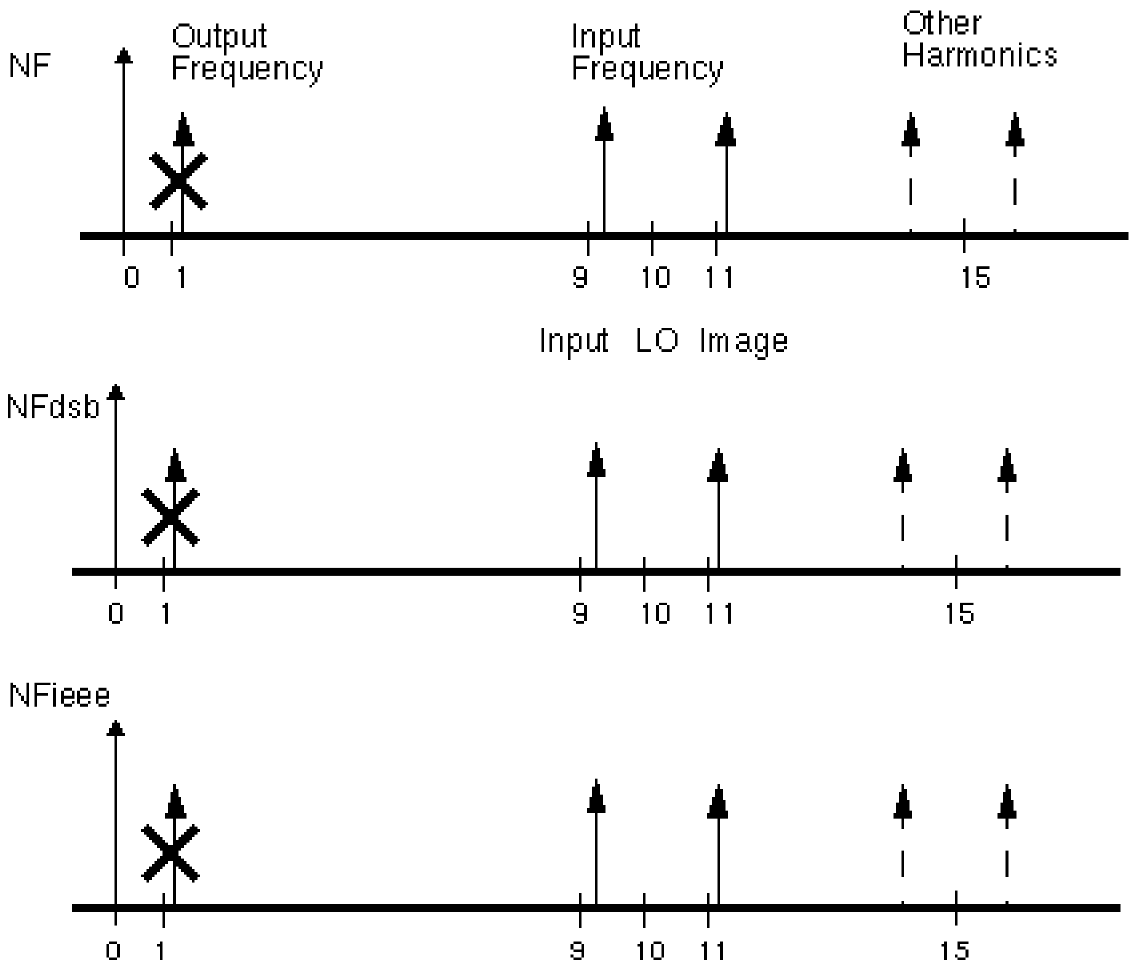
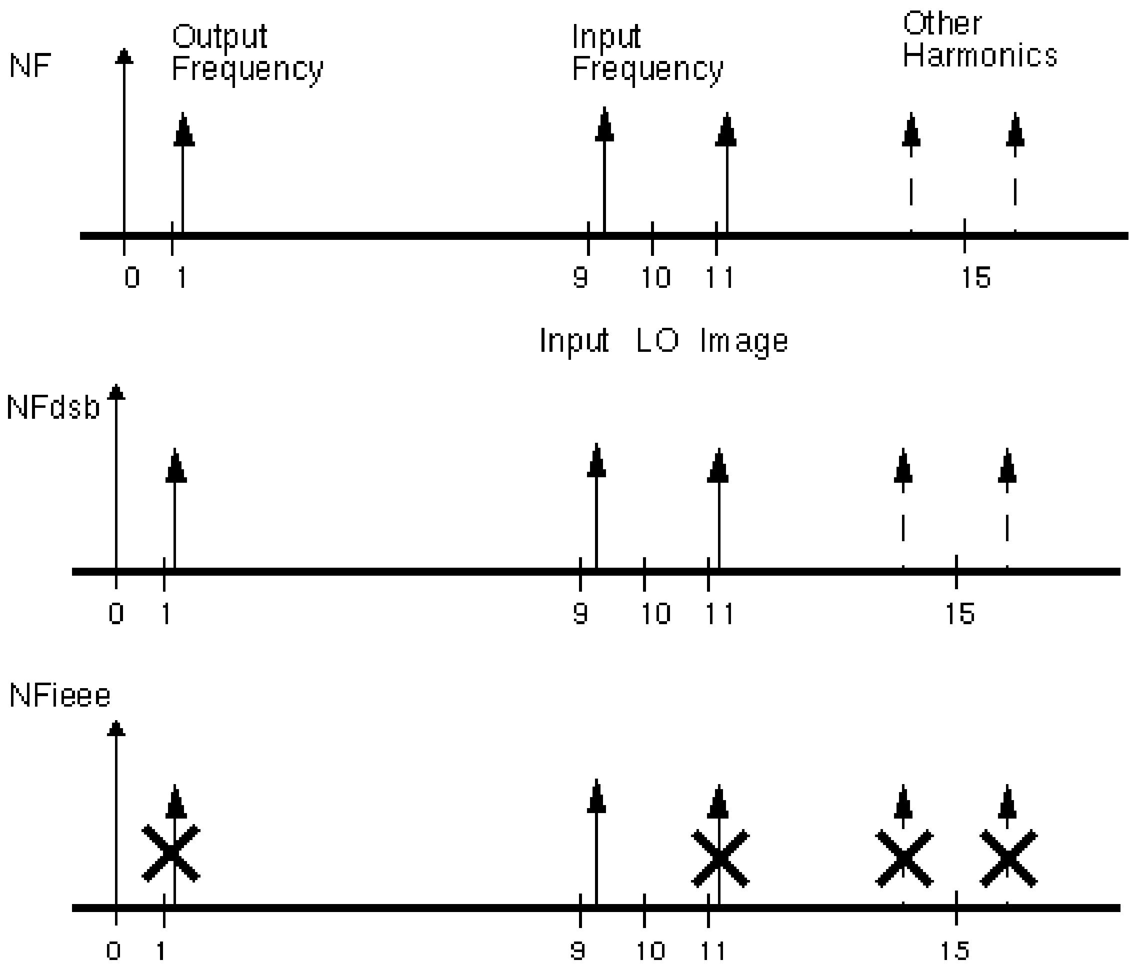


Figure L-9 Input Source Treatment for Numerator in Noise Factor Computations



Noise Computation Example

Now consider performing noise computations as part of the PSP analysis of the *ne600* mixer example presented earlier (See "PSP Analysis Example" on page 1397).

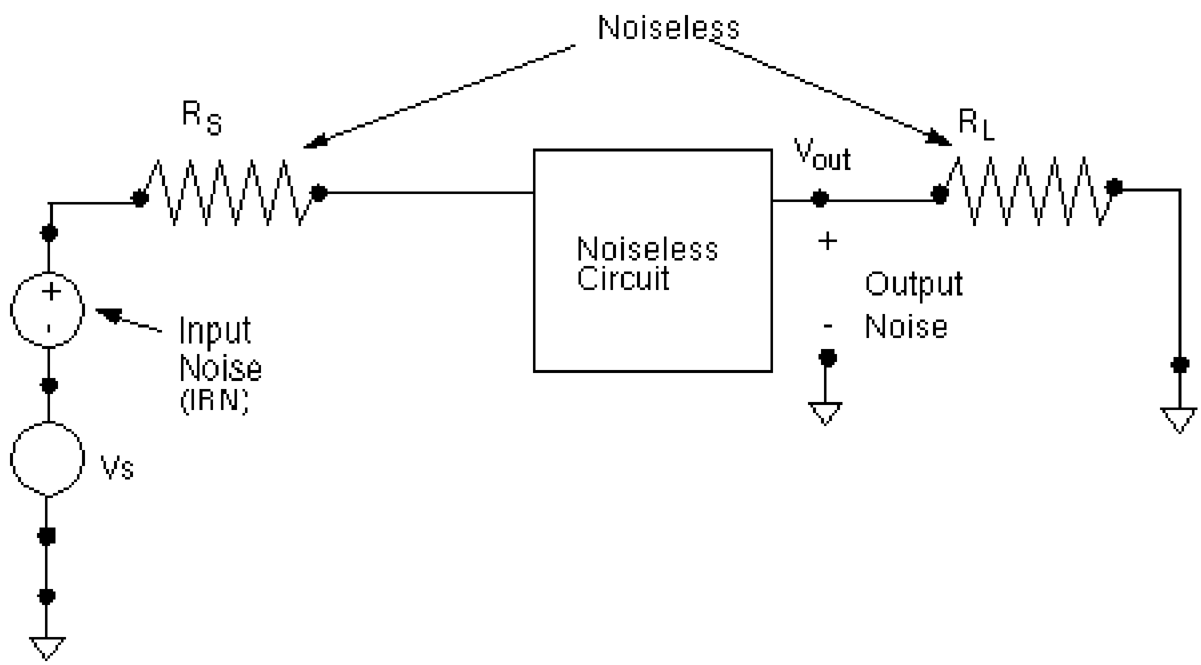
To compute noise parameters, set *donoise=yes* on the PSP analysis form and select a reasonable number for *maxsideband*. In this case, *maxsideband* should certainly be greater than 10 and probably greater than 50 or so. Setting *maxsideband=50* would account for noise folding from up to 5 GHz in frequency. The simulation can now be run as before. In addition to computing the periodic S-parameters, the periodic noise correlation matrices will also be computed, as will noise figure and the equivalent noise parameters.

Recall that the output axes for PSP analysis are set by the *`freqaxis`* option and are shared by all quantities calculated by PSP analysis. This can seem odd in the case of noise figure, where the output noise is actually analyzed for the frequency range of 80 Mhz to 130 MHz. This is because the output harmonic is 1, representing a center frequency of 100 Mhz for the relative sweep, and the sweep ran from -20 MHz to 30 MHz. Setting *`freqaxis=out`* in this example would produce an axis running from 80 Mhz to 130 Mhz.

Input Referred Noise

For a given output noise spectrum, the equivalent input noise, *input referred noise* or IRN, is the noise that, if it were generated by the circuit element specified as input, would produce the same output noise. This assumes that the circuit loading conditions, etc., are unchanged. Note that the equivalent input noise will include noise from the source and load if they are noisy in the original circuit. For example, consider a circuit with a voltage source as input, as in Figure L-10 on page 1414. If a voltage source inserted in series with the input source generates the input referred noise, and the rest of the circuit is noiseless, the noise observed at the output will be the same as in the original noisy circuit.

Figure L-10 Equivalent Input Noise



Using Input Referred Noise

Input referred noise, or IRN, gives a direct estimate of how much the noise in a circuit corrupts signals passing through, since the amplitude of the noise can be directly compared to the amplitude of signals on the input. In principle, it can also be used to build macromodels of the circuit. If a source of the same type (vsource, isource, or port) has the `noisefile` argument set to a file to which the input referred noise has been written, and all the noise elements in the circuit turned off (perhaps as may happen when replacing the circuit with an S-parameter macromodel) then the noise that appears as the circuit output will be the output noise of the original circuit. Note that Spectre `noisefile` arguments are usually given in terms of power, such as V^2/Hz, whereas the simulator usually outputs the noise in units of signal amplitude (such as $V/sqrt(Hz)$).

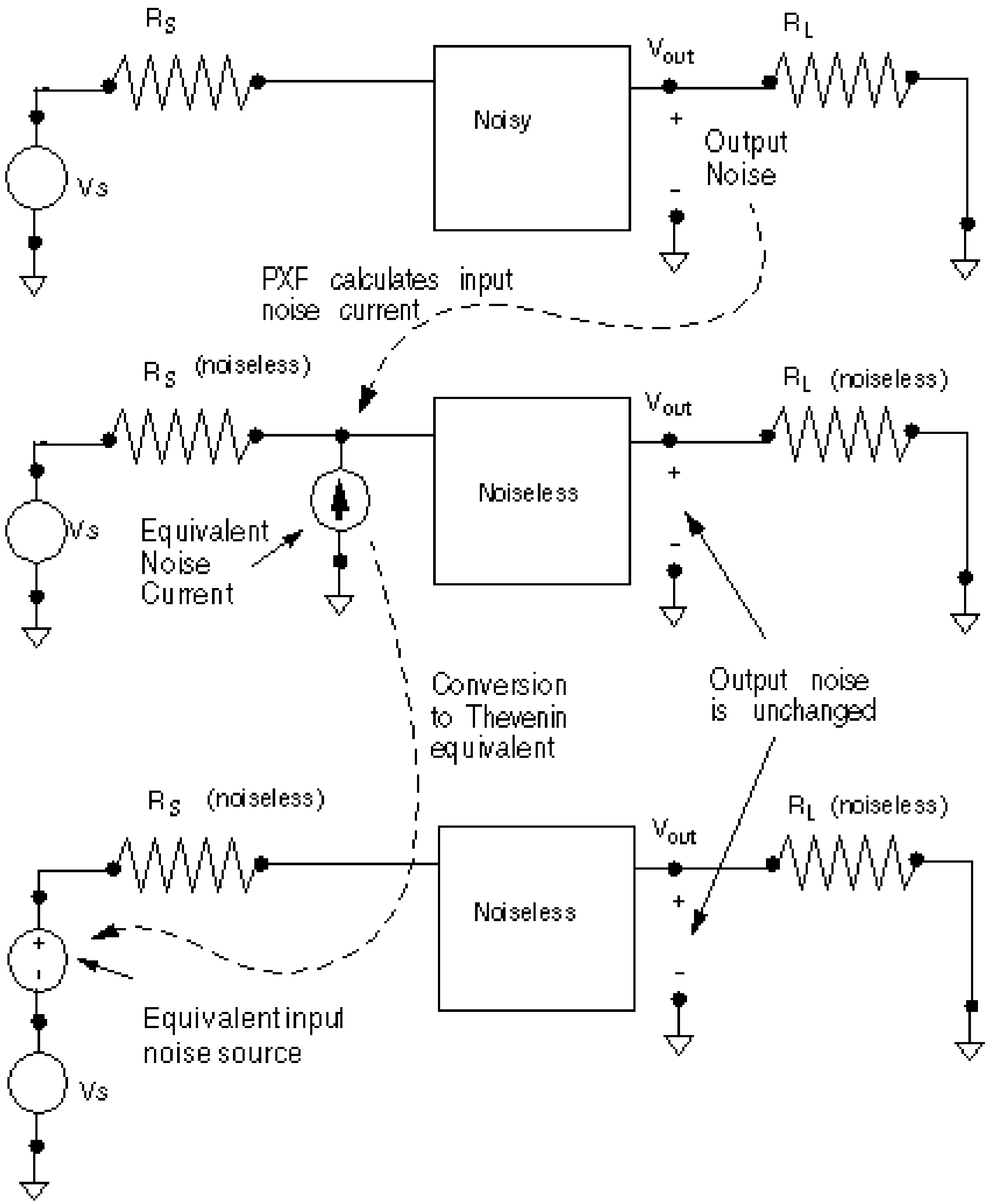
It is also possible to calculate noise figure from input referred noise, however, it is essential to be sure that any noise contribution from the load, etc., has first been properly treated. For this reason, it is recommended to use the Spectre built-in noise figure calculation, since the contribution of load noise, noise from other images, etc., is automatically accounted for. (See "The noise figure computed by <Procedure X> seems inaccurate, inconsistent, or just plain wrong." on page 1425 for more information.)

How IRN is Calculated

Spectre calculates input referred noise from the results of the Noise or Pnoise analysis and an XF or PXF analysis. The XF or PXF analysis needed is fortunately the same analysis needed to compute the noise.

As an example of an input noise calculation, consider Figure L-11 on page 1416, which shows the calculation of the equivalent input noise when a port component drives the circuit. First the total output noise is calculated. Note that this noise generally includes contributions from the circuit load. Next the results of the PXF analysis are used to express this noise as an equivalent current source attached to the circuit input. The amplitude of the current source is the amplitude of the output noise divided by the transfer function from an imaginary current source connected to the circuit input to the output node. Finally, the equivalent current source is converted into the equivalent input voltage noise by converting to Thevenin form.

Figure L-11 IRN Calculation



Relation to Gain

It is common to think of the input noise as the output noise divided by the gain,

$$IRN = \frac{OutputNoise}{G}$$

where the output noise and input noise are measured in volts/sqrt(Hz). This is a true statement if the proper gain is used. The gain reported by Pnoise analysis is in fact the ratio between the output noise and the equivalent input noise as computed in the previous section. However, the gain reported by Pnoise analysis is not necessarily the gain useful for any other purposes (see “Gain Calculations in Pnoise” on page 1422).

Referring Noise to Ports

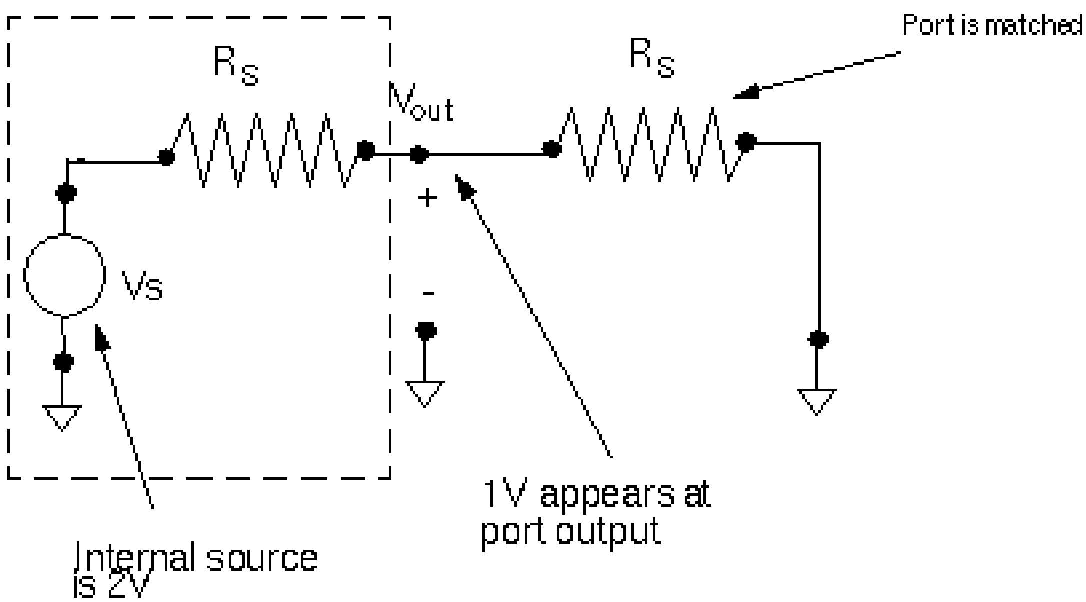
A port component in Spectre is a voltage source combined with a resistive impedance. Ports exist to enable easy generation of test fixtures, accurate noise figure calculations, and S-parameter calculations. Noise and S-parameter calculations in Spectre can (usually) only be properly performed when ports are used to drive the input and output. While the port is electrically equivalent to a source + resistor, the excitations are specified in a somewhat different way.

The port component is designed so that when a 1 V source is specified, 1 V appears at the port output when the circuit impedance is matched to the port reference impedance. The port output, however, is not the internal node that connects the voltage source and resistor, but rather the exterior resistor terminal that actually connects to the circuit under test. Thus, as shown in Figure L-12 on page 1418, the voltage on the internal voltage source must be twice the voltage specified to appear at the port output.

One implication of this convention is that when a port is specified as an input in Pnoise analysis it is the noise referred to the port that is computed, not the noise referred to the internal voltage source. In other words, the input equivalent noise reported is the noise that would be fed back to spectre as a *noisefile* argument on the port to produce the same output noise. The noise referred to the port will have amplitude half of the noise referred to the input voltage source.

Likewise, the gain reported by Spectre is the gain that would produce the equivalent input noise, referred to the port. It will be half the gain from the input voltage source.

Figure L-12 Port Driving Matched Circuitry



Gain Calculations

Definitions of Gain

To understand gain calculations in SpectreRF it is necessary to specify which of several possible gains are being calculated.

Figure L-13 Equivalent Input Noise

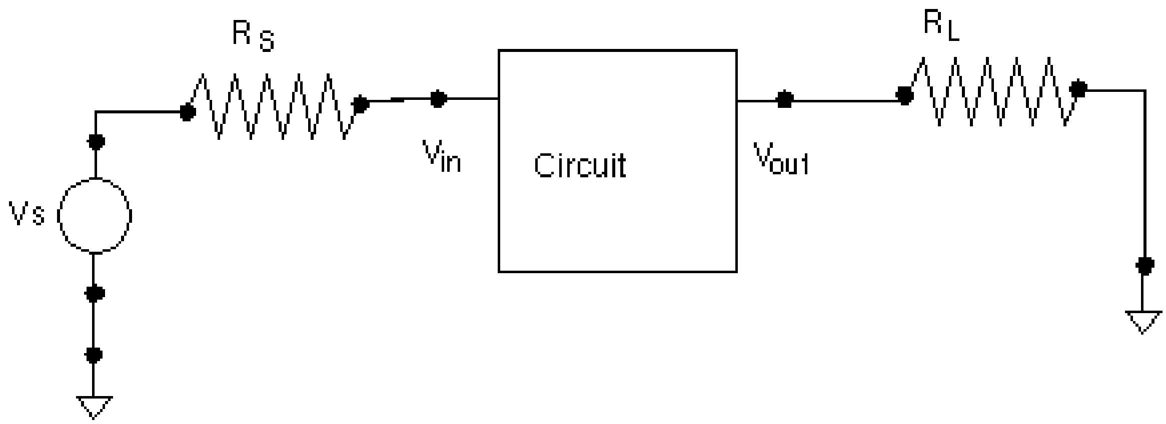


Figure L-13 on page 1419 shows a circuit configured for a gain calculation. Source and load impedances are present. These loads might represent other circuit components or *testbench* circuitry. The circuit is driven with the source *Vs* and the output voltage *Vout* measured. One possible gain is the gain Gs.

$$Gs = \frac{Vout}{Vs}$$

from the voltage source to the circuit output.

However, Gs is not the actual gain of the circuit, it is the gain of the circuit and the testbench put together. For high frequency circuits, this gain in fact cannot be measured on the bench, only in the artificial world of a circuit simulator. For these reasons, the proper way to calculate gain is the use a port component, which allows the user to specify the impedance of the driving circuitry. The Spectre PSP analysis computes the gain,

$$G = \frac{Vout}{Vin}$$

which is the voltage gain from the circuit input to the circuit output.

G is calculated for historical reasons, and is not necessarily a useful number, but once an SP or PSP analysis is performed, several other gains can be defined and computed. Varying source and load impedance can also be considered. In the 4.4.5 release, the SP direct plot form displays various gains. The PSP direct plot form does not automatically compute the various gains but they can be computed from the basic definitions using the waveform calculator.

Some gains of interest in two-port circuits are:

- G_A (Available Gain)
- G_P (Power Gain)
- G_T (Transducer Gain)
- G_{umx} (Maximum Unilateral Transducer Power Gain)
- G_{max} (Maximum Available Gain)

GA (Available Gain), the power gain obtained by optimally (conjugately) matching the output of the network.

$$G_A = \frac{|S_{21}|^2(1-|\Gamma_S|^2)}{|1-S_{11}\Gamma_S|^2(1-|\Gamma_2|^2)}$$

where

$$\Gamma_2 = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1-S_{11}\Gamma_S}$$

and Γ_S is the source reflection coefficient,

$$\Gamma_S = \frac{Z_S - Z_{S,\,ref}}{Z_S + Z_{S,\,ref}}$$

with Z_S is the source impedance and $Z_{S,\,ref}$ is the reference impedance for the input port.

GP (Power Gain), the power gain obtained by optimally (conjugately) matching the input of the network.

$$G_P = \frac{|S_{21}|^2(1-|\Gamma_L|^2)}{|1-S_{22}\Gamma_L|^2(1-|\Gamma_1|^2)}$$

where

$$\Gamma_1 = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1-S_{22}\Gamma_L}$$

and Γ_L is the load reflection coefficient.

GT (Transducer Gain), the ratio of the power dissipated in the load to the power available from the source,

$$G_T = \frac{(1-|\Gamma_S|^2)|S_{21}|^2(1-|\Gamma_L|^2)}{|(1-S_{11}\Gamma_S)(1-S_{22}\Gamma_L)-S_{12}S_{21}\Gamma_S\Gamma_L|^2}$$

Gumx (Maximum Unilateral Transducer Power Gain)

$$G_{umx} = \frac{|S_{21}|^2}{(1-|S11|^2)(1-|S22|^2)}$$

Gmax (Maximum Available Gain), the transducer power gain when there exists a simultaneous conjugate match at both ports.

For $Kf > 1$

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}[Kf{-}SQRT((Kf)^2-1)]$$

For $Kf < 1$

$$G_{max} = \frac{|S_{21}|}{|S_{12}|}$$

Where K_f is the stability factor, K_f

$$K_f = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |D|^2}{2|S_{22}||S_{12}|}$$

and

$$D = S_{11}S_{22} - S_{21}S_{12}$$

Gain Calculations in Pnoise

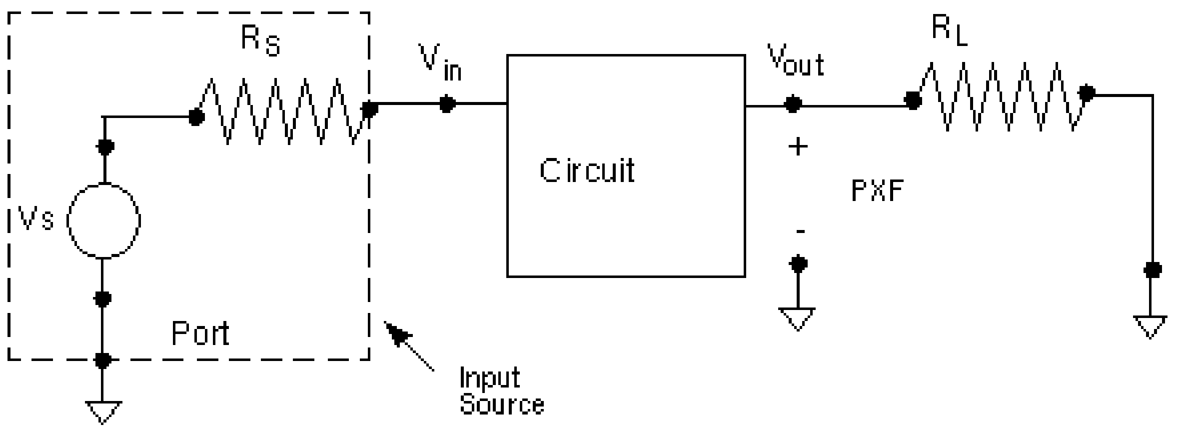
When ports are used to drive the circuit input and a Pnoise analysis is performed, Pnoise analysis reports a quantity called `gain` that can be misleading. Consider the circuit in Figure L-14 on page 1423 where a port component is used as the input to the circuit, and specified as such by using the iprobe option to Pnoise.

The goal of Pnoise analysis is usually to calculate the noise at the circuit output, as shown in Figure L-14 on page 1423. Pnoise analysis also calculates an approximation to the gain G as a by-product of the input-referred noise calculation. It turns out that if it is assumed that the circuit input impedance is matched, i.e., the impedance seen by the port component shown as the dashed box in Figure L-14 on page 1423, is the same as the source impedance, then the gain needed for the input-referred noise calculation is also the gain G.

However, the Pnoise analysis is based on a series of PXF analyses and when the input is not matched, then a single PXF analysis is not sufficient to calculate G. PSP analysis is needed to accurately calculate G independent of match conditions.

Figure L-14 Pnoise with Port Component

.



Coincidentally, because of the matched-input assumption made both in Pnoise analysis and in the definition of the port, the gain G from circuit input to circuit output that is reported by Pnoise analysis happens to be exactly twice the gain from the source to the output, Gs. This can be seen by noting that if the circuit input is matched, then the circuit acts as a voltage divider, and the voltage at Vin is

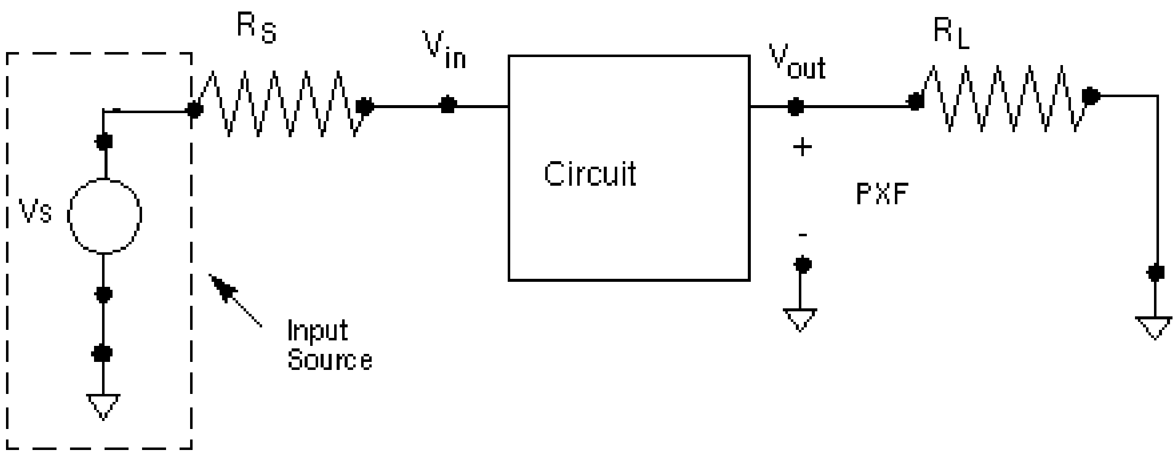
$$Vin = \frac{Zin}{Zin + Rs}Vs = \frac{Rs}{Rs + Rs}Vs = \frac{Vs}{2}$$

Thus in this special case, Vout/Vin is twice Vout/Vs. The *`gain`* reported by Pnoise is twice Gs regardless of the match conditions.

To alleviate this confusion, in future releases the *`gain`* number will not be reported by Pnoise analysis when the input is driven by a port.

Now consider when a separate vsource and resistor are used to drive the circuit, as shown in Figure L-15 on page 1424. In this case, the voltage source would be specified as the input probe. Spectre computes the gain from the input probe to the output, which in this case is the gain from Vs to the output, Gs, since Vs was specified as the input source.

Figure L-15 Pnoise With vsrc Component



Phase Noise

The Direct Plot form inthe analog design environment plots *output noise* and *phase noise*. The phase noise form is designed for use in oscillator noise computations.

The *output noise* plots are the Total Output Noise (`out`) data, taken directly from the psf files with no modification.

The *phase noise* plots show the noise power relative to the carrier power. Technically the plot is not phase noise, simply normalized output noise. However, for oscillators, close to the fundamental frequency, the noise is mostly phase noise.

The normalization is done relative to the power in the fundamental component of the noise-free oscillation as calculated by the PSS analysis. If you look at the PSS analysis results in the frequency domain, the fundamental will have a particular amplitude $V1$, which means that the fundamental component of the oscillation is $V1\cos(2pf_c + j)$, for some φ, where f_c is the fundamental frequency.

Defining

noise power = out^2,

where out is the *Total Output Noise*

carrier fundamental power = $(V_1)^2/2$

The normalized power is $\text{normalized power} = (\text{out2})/[(V_1)^2/2]$

Thus the formula used by Direct Plot in Artist 4.4.3 and later to plot the normalized power (*phase noise*) in `dB10` is:

$$\text{normalized power in dB} = 10\log10\{(\text{out}^2)/[(V_1)^2/2]\}$$

For more information on phase noise, see Appendix A, “Oscillator Noise Analysis,” which summarizes how to get good phase noise calculations.

Frequently Asked Questions

The noise figure computed by <Procedure X> seems inaccurate, inconsistent, or just plain wrong.

The key to noise figure computation is to remember that three pieces of information are needed:

- The total noise at the output
- How much of that noise is due to the output load
- How much of that noise is due to the input source

Noise figure is the log of noise factor, and noise factor is fundamentally the total output noise less the noise due to the load ratioed to the noise due to the input. The easiest way to be sure of getting the correct noise figure is to use ports as input and output sources and loads and to have the Spectre simulator perform the noise figure computation.

SpectreRF correctly accounts for the noise due to the load, as well as the different possible treatments of noise from the input source that result in different types of noise figure that are used in RF circuits (customary SSB, IEEE SSB, DSB, as discussed in “Noise Figure” on page 1405). See “Performing Noise Figure Computations” on page 1405 for information on setting up noise figure computation.

Most mistakes in noise figure calculations arise when improperly accounting for items two and three in the list above. Hand computations of noise figure require that you correctly account for noise from the load and source, which can be difficult. For example, some users have tried to use the information from the noise summary table to calculate noise figure. The noise summary table reports total noise contributions, from all sidebands present in the Pnoise analysis, sorted by contributor. If the total noise reported in the noise summary is used as the numerator for the noise factor computation the wrong answer is obtained because the noise due to the load has not been subtracted off. Likewise, the noise summary table does not sort information by sideband. To compute SSB noise figure, the denominator of the noise factor fraction must be only the noise from the reference sideband, not the total from all sidebands as reported in the noise summary. Noise figure calculations based on the reported input-equivalent noise, or on output noise and gain, suffer similar problems.

You can compute the noise figure by hand, but you must take care. For example, use the following steps to compute the customary single-sideband noise figure from raw data.

- Perform a Pnoise analysis with the *saveallsidebands* parameter set to *yes*.
- Using the results browser or direct plot, obtain the total noise at the output. Call this *OutputNoise*.
- Now use the results browser to obtain the amount of total noise due to the load contributed from the zero sideband. Call this *LoadNoise*.
- Finally, again using the results browser, obtain the amount of noise due to the source contributed from the reference sideband. Call this *sourcenoise*.

The noise factor is then

$$F = \frac{OutputNoise - LoadNoise}{SourceNoise}$$

Be sure to use the same units on all quantities. Note that, because in this procedure the noise due to the source and load are obtained by hand inspection, the procedure can be used to compute noise figure even if ports are not used to drive the circuit. Hand calculations are not recommended because they are tedious and prone to error.

In versions of the Analog Circuit Design Environment prior to 4.4.5, it was possible to perform noise figure computations in ways that incorrectly or incompletely specified the sources/input or load/output. Since, at that time, there was no way for the simulation environment to track down which component was contributing, for example, the *LoadNoise*, this part of the noise factor might have been neglected.

The rule of thumb is: *if you have not specified your sources and loads to the simulator, the simulator does not know what the loads and sources are*, so the simulator may not be able to compute noise figure properly. Generally, the input to the circuit must be a *port*, and *not* some other electrically equivalent combination of components. The output may be a resistor probe, or it may be a port. In up-to-date versions of software, 4.4.2 and later, using ports (or possibly a resistor as the output probe) will insure accurate computations. Inspect the netlist if you are unsure about the identity of the components presented to the simulator.

Finally, if you ever have any doubt about the propriety of the procedure you are using, use the results browser to access the noise figure directly from the simulation data. This data is correct in all software versions.

Why is the gain reported by Pnoise twice the gain that I expect?

- First, unlike some circuit simulators, Pnoise and PSP compute the gain of the circuit from its input to its output, instead of from an external source to the output. See "Definitions of Gain" on page 1418 to see if the gain you expect is actually the gain that you want.
- Second, Pnoise makes certain assumptions about input matching in order to calculate this gain. See "Gain Calculations in Pnoise" on page 1422 for further explanations of the way Pnoise calculates gain.

In general, you should not use the gain calculated by Pnoise unless you know your circuit is input-matched. Use PSP instead.

Does SpectreRF compute single-sideband (SSB) or double-sideband (DSB) noise figure?

The SpectreRF Pnoise analysis computes SSB noise figure, (except in certain special cases). PSP can compute both SSB and DSB noise figure. See "Noise Figure" on page 1405.

Why does the axis for noise or noise figure have negative frequencies?

The axis labeling in Spectre is fairly independent of the actual computation. In all cases, Spectre computes single-sided noise (that is, there is no need to add noise from *negative* frequencies to get the total noise power or noise figure).

Recall that PSP analysis performs many computations at once—periodic S-parameters, noise correlation matrices, noise figure, etc. In 4.4.5, for historical reasons, all these quantities are stored as a sweep with a single axis. An axis appropriate for visualizing s11 may not be good to display NF.

To obtain strictly positive axis labels for noise quantities in PSP, choose a positive integer for the output harmonic, use *sweeptype=relative*, and set *freqaxis=out* on the PSP analysis options form.

How does match affect the noise and gain calculations in SpectreRF?

If the input impedance is not matched to the port impedance, the gain and input referred noise, or IRN, reported by Pnoise analysis is probably not the gain you want. out, F, NF, and in (input-referred noise) will still be correct.

PSP analysis computes the correct G, *i*n (input-referred noise), F and NF regardless of match.

Will Spectre include noise parameter data from S-parameter files?

Spectre supports writing and reading S-parameter files, but as of the 4.4.5 release it does not write noise parameter data into files, nor can noise parameter data be read from file and included in a simulation.

Known Problems and Limitations

Dubious AC-Noise Analysis Features

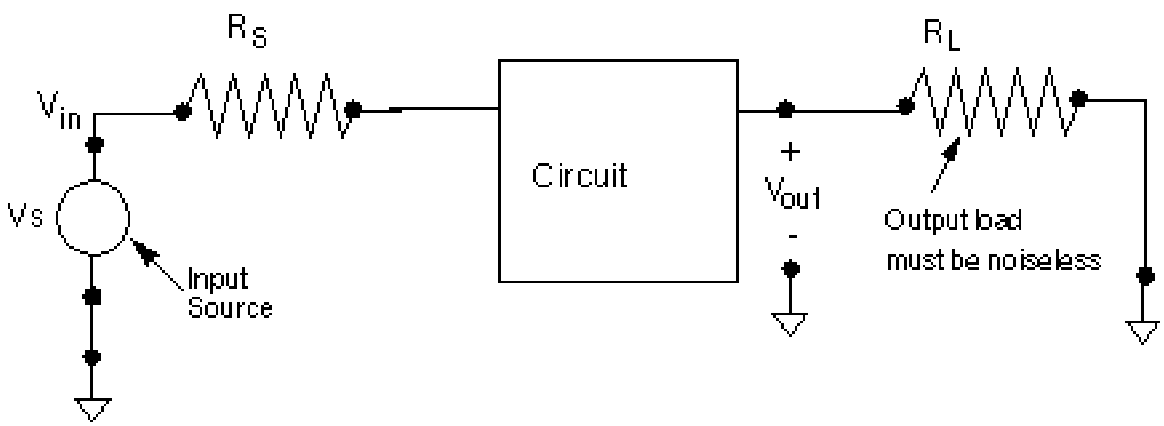
When computing noise figure in circuits with a DC bias point in releases prior to 4.4.5, the noise figure available from the direct plot form was obtained from knowledge of the total output noise computed with a Noise analysis, the source impedance as entered on a form, and circuit gain data obtained in an auxiliary AC analysis. Simulators other than Spectre still require this procedure.

In 4.4.5, if you use ports as the output load and input source, the noise figure is obtained directly from the Spectre simulator data. An artifact of the previous use model is that the noise figure direct plot still requires an AC analysis to run, but this analysis has no effect on the computed noise figure if ports are used as sources and loads.

If ports are not used as the input source and load (for example, using a *vsin* component as the input and a pair of nodes as the output) it is still possible to compute noise figure in the Analog Circuit Design Environment, but the procedure is not recommended as it can be tricky to obtain the desired result (see “Performing Noise Figure Computations” on page 1405 and “The noise figure computed by <Procedure X> seems inaccurate, inconsistent, or just plain wrong.” on page 1425).

To obtain reasonable noise figure calculations using this (unsupported) procedure, you must configure the circuit as shown in Figure L-16 on page 1429. Be sure to select the proper nodes as input and output (note that a *port* component cannot be used in this topology because the *Vin* node cannot be accessed). The contribution of the output load to the total noise must be eliminated by making the node noiseless, for example by setting `Generate noise?` in a resistor component to `no`.

Figure L-16 Alternative (Undesirable) Noise Figure Computation Setup



Note that you cannot use this topology to compute noise figure in Pnoise, SP or PSP analyses.

Gain in Pnoise and PSP Analyses Inconsistent

The `gain` computed by the Pnoise analysis is really a number that relates equivalent input and output noise. PSP analysis computes a somewhat different quantity that can be interpreted as a circuit gain. If the input impedance of the circuit is matched, the PSP and Pnoise gains will match. See the "Gain Calculations" on page 1418 for details on computing circuit gains.

Harmonics and Sidebands in PSP, PAC, PXF, and Pnoise Analyses

Frequent users of SpectreRF may already be familiar with the concepts of *harmonics* and *sidebands*.

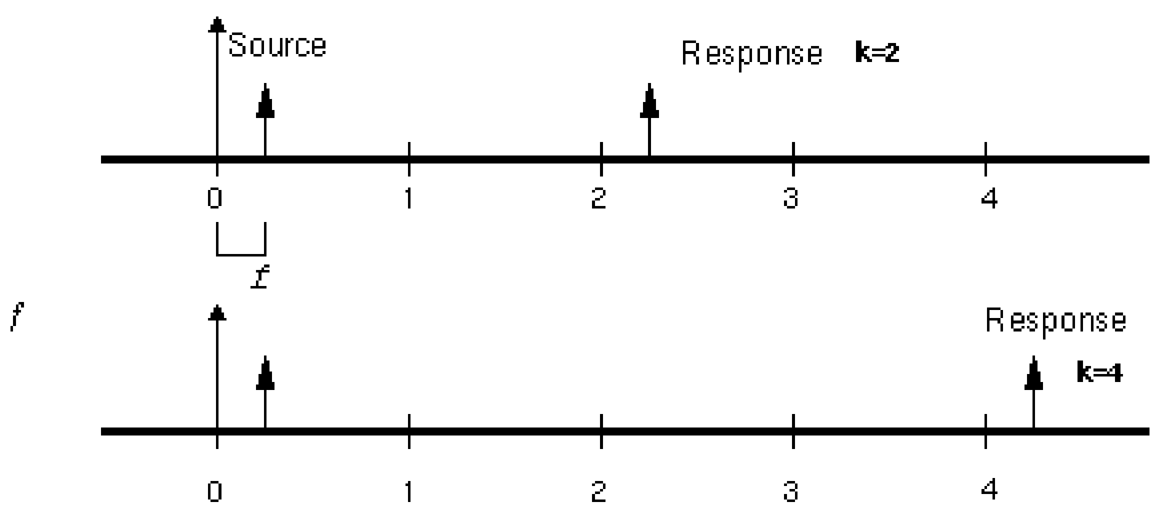
- Typically, the term *harmonic* is used to refer to a signal at a multiple of the carrier or local oscillator frequency.
- If an additional input is applied, such as for RF data, additional responses appear as *sidebands*.

In small-signal analysis, only the first-order sidebands are computed.

In the context of small-signal analysis, a frequency is specified by an index k and a frequency offset f. These numbers specify a frequency $kf_0 + f$. In an absolute indexing scheme, such as used in PSP analysis, if the frequency f is restricted to lie in an interval of size f_0, then for any given frequency, the numbers k and f are unique.

For example, consider an upconversion mixer, as shown in Figure L-17. Two possible computations are shown. In both cases, the source is at k=0, baseband, and relative offset f. In the top half of the figure, the response is at k=+2, frequency f, and in the bottom half, k=4, offset f.

Figure L-17 Absolute Indexing Schemes in SpectreRF Small-Signal Analyses



In contrast, PAC, PXF, and Pnoise analyses use a relative indexing scheme. In a relative indexing scheme, a single frequency may be referred to in different ways depending on context.

Figure L-18 Relative Indexing in SpectreRF

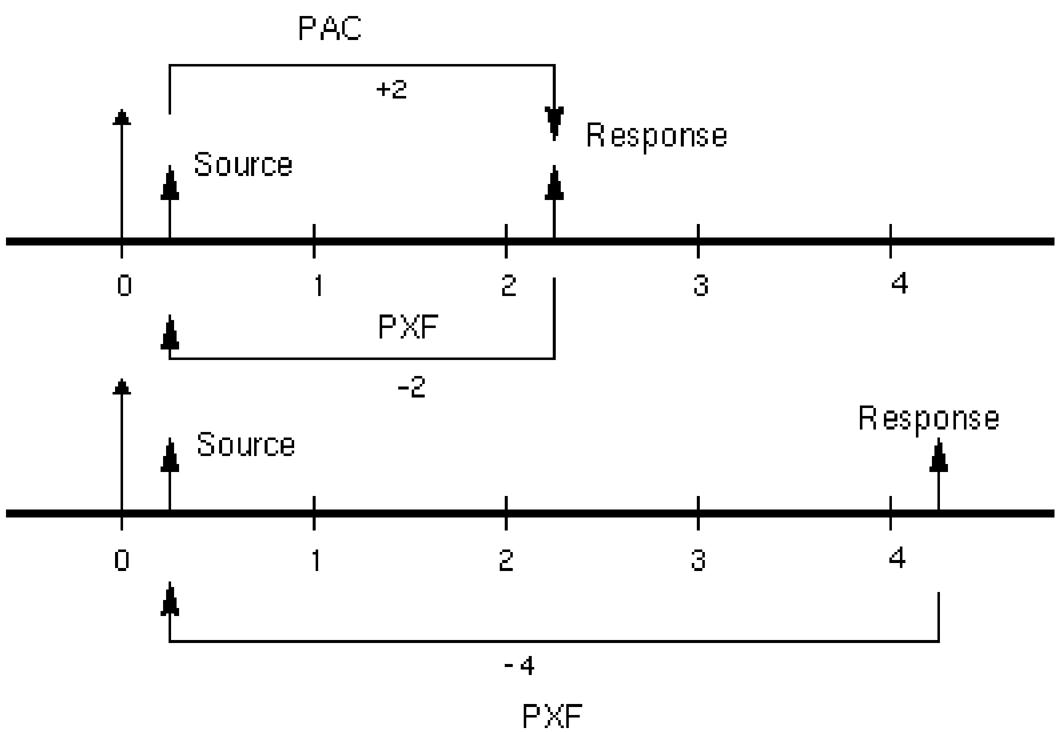


Figure L-18 on page 1431 shows how relative indexing may change depending on context. In the top half of the figure, a source may be specified in PAC analysis using either *sweeptype=absolute* and frequency $-f_0 + f$, or *sweeptype=relative*, *relharmnum=-1*, and frequency f. The response is shown at frequency $f_0 + f$, which is $k=2$, because $f_0 + f = -f_0 + f + 2f_0$.

If instead a PXF analysis was used to analyze the circuit, then the location of the *response* is specified with either *sweeptype=absolute* and frequency $f_0 + f$, or *sweeptype=relative* and *relharmnum=+1*. The source is at $k=-2$. Now consider the bottom half of the figure. The response is shown at the frequency $3f_0 + f$. If PXF analysis is used to analyze the circuit, the source lies at $k=-4$.

Note that if all the frequencies involved in the above examples were shifted upwards by f_0, then the indexing in PSP analysis would change by $+1$, but the indexing of the transfer functions in PXF and PAC analyses would remain unchanged, even though different frequencies, and thus physical different transfer functions, will be computed.

The names used for the responses and transfer functions in the SpectreRF environment are not entirely consistent in release 4.4.5 and earlier. The SpectreRF interface asks for *sidebands* in order to specify which of the transfer functions associated with the relative index k are wanted. This is reasonable because the small-signal responses appear as sidebands to the harmonics of the fundamental frequency of the PSS steady-state solution. However, the data is output and displayed with the label *harmonic*. Thus the index naming is not consistent. However, since the transfer function associated with the relative index k results from the small-signal being convolved with the k^{th} harmonic of the steady-state solution, the *harmonic* label is not entirely inappropriate either.

Index

Symbols

Numerics

A

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W