Spectre® Circuit Simulator RF Analysis Theory

Product Version 19.1
January 2020

Contents

Preface

Spectre$^®$ circuit simulator or Spectre$^®$ Accelerated Parallel Simulator (Spectre$^®$ APS) base products with the Spectre RF analysis option provide simulation capabilities for RFIC designers. The simulators:

■ Support efficient calculation of the operating point, transfer function, noise, and signal distortion of common RF and communication circuits, such as mixers, LNA, oscillators, sample and holds, and switched capacitor filters.

■ Support a multi-technology simulation (MTS) mode that enables the simulation of systems consisting of blocks designed with different processes, such as RF System-in-Package (SIP).

This user guide assumes that you are familiar with:

■ RF circuit design

■ SPICE simulation

■ Virtuoso$^®$ Analog Design Environment (ADE)

The Spectre$^®$ circuit simulator, the high performance Spectre APS, and the Spectre RF analysis option are part of the MMSIM (multi-mode simulation) portfolio and are accessible using individual a la carte licenses or MMSIM tokens. Note that mixing of tokens and a la carte license is not allowed. Table 3-1 on page 11 and Table 3-2 on page 12 show the capabilities offered with the base products Spectre simulator or Spectre APS and the Spectre RF analysis option.

Table 3-1 Spectre$^®$ Circuit Simulator with Spectre RF Analysis Option

Features
Transient Noise Analysis
DC, small-signal analyses and transient
Monte Carlo, DC mismatch, Parametric sweep
RF Harmonic Balance

Features
RF Shooting Newton
Large-signal Analysis - Periodic and Quasi-periodic (HB, PSS and QPSS)
Noise Analysis - Periodic and Quasi-periodic (pnoise, QPNoise, sampled, jitter)
Small-signal Analysis - Periodic and Quasi-periodic (PAC, PXF, PSP, QPAC, QPXF, QPSP)
Periodic Stability Analysis (PSTB)
Envelope Analysis (AM, PM, FM, Autonomous)
Rapid Distortion Analyses - Perturbation-based IP2 and IP3
Co-simulation with Simulink® from The MathWorks
MMSIM Toolbox for MATLAB® from The MathWorks

Table 3-2 Spectre® Accelerated Parallel Simulator with Spectre RF Analysis Option

Features
All analysis and features in Spectre
RF Harmonic Balance
RF Shooting Newton
Large-signal Analysis - Periodic and Quasi-periodic (HB, PSS and QPSS)
Noise Analysis - Periodic and Quasi-periodic (pnoise, QPNoise, sampled, jitter)
Small-signal Analysis - Periodic and Quasi-periodic (PAC, PXF, PSP, QPAC, QPXF, QPSP)
Periodic Stability Analysis (PSTB)
Envelope Analysis (AM, PM, FM, Autonomous)
Rapid Distortion Analyses - Perturbation-based IP2 and IP3

Cadence offers multi-core simulation up to 64 cores for RF analysis, enabled with the base product Spectre APS along with Spectre RF analysis and the Spectre CPU Accelerator options.

Please contact your account representative for more details on the licensing and packaging.

Using License Queuing

You can turn on license queuing by using the `lqtimeout` command line option:

```
spectre +lqtimeout time_in_seconds
```

If a license is not available when you begin a simulation job, the Spectre circuit simulator and Spectre APS wait in queue for a license for the specified time. If you specify the value 0 for this option, the Spectre circuit simulator waits indefinitely for a license. The `lqtimeout` option has no default value for the standalone Spectre circuit simulator. If you invoke Spectre through the Analog Design Environment, the default value for `lqtimeout` is 900 seconds.

You can use the `lqsleep` option to specify the interval (in seconds) at which the Spectre circuit simulator should check for license availability. The default value for `lqsleep` is 30 seconds.

```
spectre +lqsleep interval
```

For more information on any of the above options, see `spectre -h`.

Suspending and Resuming Licenses

You can direct Spectre and Spectre APS to release licenses when suspending a simulation job. This feature is aimed for users of simulation farms, where the licenses in use by a group of lower priority jobs may be needed for a group of higher priority jobs. To enable this feature, simply start Spectre with `+lsuspend` command-line option. In the Solaris environment, press `ctrl+z` to suspend the Spectre license. All licenses are checked in. To resume simulation, press `fg`. These keystrokes may not work if you have changed the default key bindings.

For information on tracking token licensing, see the *Virtuoso® Software Licensing and Configuration Guide*.

In Virtuoso® Analog Design Environment, the `lqtimeout` and `lqsleep` options are controlled by the following options:

```
spectre.envOpts lsuspend boolean t
spectre.envOpts licQueueTimeOut string "900"
spectre.envOpts licQueueSleep string "30"
```

Related Documents for Spectre

This user guide contains information about the functionality. The following documents provide more information about SpectreRF and related products.

■ The Spectre circuit simulator is often run within the analog circuit design environment, under the Cadence design framework II. To see how the Spectre circuit simulator is run under the analog circuit design environment, read the *Virtuoso Analog Design Environment User Guide.*

■ To learn more about specific parameters of components and analyses, consult the Spectre online help (`spectre -h`).

■ To learn more about the equations used in the Spectre circuit simulator, consult the *Spectre Circuit Simulator Components and Device Models Manual*.

■ The Spectre circuit simulator also includes a waveform display tool, Virtuoso Visualization and Analysis tool, to use to display simulation results. For more information about the tool, see the *Virtuoso Visualization and Analysis User Guide*.

■ For more information about using the Spectre circuit simulator with Verilog-A, see the *Verilog-A Language Reference* manual.

■ For more information about RF theory, see *Spectre Circuit Simulator RF Analysis Theory*.

■ For more information about how you work with the design framework II interface, see *Cadence Design Framework II Help*.

■ For more information about specific applications of Spectre analyses, see *The Designer's Guide to SPICE & Spectre*[1].

Third Party Tools

To view any `.swf` multimedia files, you need:

■ Flash-enabled web browser, for example, Internet Explorer 5.0 or later, Netscape 6.0 or later, or Mozilla Firefox 1.6 or later. Alternatively, you can download Flash Player (version 6.0 or later) directly from the Adobe website.

■ Speakers and a sound card installed on your computer for videos with audio.

---

1. Kundert, Kenneth S. *The Designer's Guide to SPICE & Spectre.* Boston: Kluwer Academic Publishers, 1995.

Typographic and Syntax Conventions

This list describes the syntax conventions used for the Spectre circuit simulator.

`literal`	Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names, filenames and paths, and any other sort of type-in commands.
argument	Words in italics indicate user-defined arguments for which you must substitute a name or a value. (The characters before the underscore (_) in the word indicate the data types that this argument can take. Names are case sensitive.
|	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character.
[ ]	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
{ }	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.
...	Three dots (...) indicate that you can repeat the previous argument. If you use them with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.

Note: The language requires many characters not included in the preceding list. You must enter required characters exactly as shown.

1

Basic Reference Information

Spectre$^{\circledR}$ circuit simulator RF analysis (Spectre RF) adds capabilities to the Spectre circuit simulator (Spectre) that are particularly useful to analog and RF designers, including the ability to

■ Efficiently and directly compute a steady-state solution

■ Characterize circuits that translate frequency

You can use Spectre RF simulation in combination with both the accurate Fourier analysis capability of Spectre and the Verilog$^{\circledR}$-A behavioral modeling language to simulate RF circuits. This book describes the theory of operation of Spectre RF simulation.

Spectre RF simulation brings many concepts to the Spectre simulator.

■ Periodic Steady-State analysis — PSS (a large-signal analysis)

■ Periodic Small-Signal analyses — PAC, PSP, PXF, and Pnoise

■ Quasi-Periodic Steady-State analysis — QPSS (a large-signal analysis previously called pdisto analysis)

■ Quasi-Periodic Small-Signal analyses—QPAC, QPSP, QPXF, and QPnoise

■ Envelope analysis

■ The Harmonic Balance simulation engine

Periodic Analyses

Note: Modulated periodic noise, time domain noise and jitter analyses have been deprecated. For more information on the current use model of periodic noise, refer to the *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide*.

Periodic Steady-State (PSS) analysis is a large-signal analysis that directly computes the periodic steady-state response of a circuit with a simulation time that is independent of

the time constants of the circuit. PSS quickly computes the steady-state response of circuits that exhibit extremely long time constants, such as high-Q filters and oscillators.

Periodic Small-Signal (PAC, PSP, PXF, Pnoise and PSTB) analyses are similar to the conventional small-signal analyses (AC, SP, XF, Noise and STB) but you can apply them to periodic circuits where frequency conversion plays a critical role. The conventional small-signal analyses (AC, SP, XF, Noise and STB) linearize about the DC or time-invariant operating point and they do not include frequency conversion effects.

After a PSS analysis, the circuit is linearized about a periodic (time varying) operating point with frequency conversion effects included. After the PSS analysis, you can perform one or more of the periodic small-signal analyses. Example circuits where you might use the periodic small-signal analyses include conversion gain in mixers, noise in oscillators, and switched-capacitor filters.

Quasi-Periodic Analyses

Quasi-Periodic Steady-State (QPSS) analysis is a large signal analysis you can use for circuits with multiple large tones. QPSS analysis computes the steady-state responses of a circuit driven by two or more signals at unrelated frequencies. You select one sinusoidal or pulse signal as the large signal. Any additional signals, called moderate signals, must be sinusoids.

Quasi-Periodic Small-Signal (QPAC, QPSP, QPXF, and QPnoise) analyses are similar to the periodic small-signal analyses (PAC, PSP, PXF, and Pnoise) but they extend to circuits where frequency conversion and intermodulation effects both play a critical role.

After a QPSS analysis to determine the quasi-periodic operating point and to linearize the circuit about the quasi-periodic operating point, you can perform one or more of the quasi-periodic small-signal analyses. It is the quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion and intermodulation. Example circuits where you might use the quasi-periodic small-signal analyses include conversion gain in mixers, noise in oscillators, switched-capacitor filters and other periodically or quasi-periodically driven circuits.

Envelope Analysis

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

The Simulation Engines

Spectre RF provides a choice of simulation engines between the *shooting method* and the *harmonic balance method* (HB) with most analyses. The harmonic balance engine complements the capabilities of the shooting method.

Harmonic Balance (HB) Engine

The harmonic balance engine supports frequency domain harmonic balance analyses. It provides efficient and robust simulation for linear and weakly nonlinear circuits. The harmonic balance engine is supported on the Solaris, Linux, HP and IBM platforms for both 32 and 64 bit architectures. See "The Harmonic Balance Engine" on page 30 for more information on this simulation engine.

Time Domain (TD) Engine

The Spectre RF simulator has traditionally used an engine known as the *shooting method* [kundert90] to implement periodic and quasi-periodic analyses and the envelope analysis. The shooting method is a time domain method and it is used in most descriptions and examples in this manual.

Periodic Steady-State Analysis

Periodic Steady-State (PSS) analysis directly computes the periodic steady-state response of a circuit. Spectre RF simulation has traditionally used an engine known as the *shooting method* [kundert90] to implement PSS analysis. Now, Spectre RF supports a choice of simulation engines between the shooting method and the harmonic balance method. See "The Harmonic Balance Engine" on page 30 for more information.

The Shooting Method

The shooting method is used in the majority of examples in this manual. The shooting method is a time domain method that operates by efficiently finding an initial condition that directly results in steady state, as illustrated in Figure 1-1.

Figure 1-1 The Shooting Method

The signal on the left starts at point v_i that does not result in periodicity.

For the signal on the right, the starting point was adjusted by the shooting method to directly result in periodic steady state.
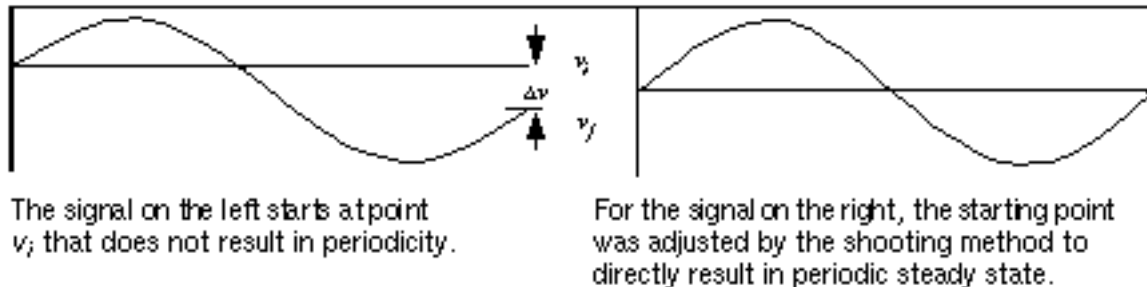
The shooting method is an iterative method that begins with an estimation of the desired initial condition. The shooting method computes the initial condition that results in the signals being periodic as defined by $v_f - v_i = \Delta v = 0$. The circuit is evaluated for one period starting from the initial condition. The final state (v_f) of the circuit is computed along with the sensitivity of the final state with respect to the initial state ($\partial^{11} v_f / \partial v_i$). The nonperiodicity ($\Delta v = v_f - v_i$) and the sensitivities are used to compute a new initial condition. If the final state is a linear function of the initial state, then the new initial condition directly results in periodicity. Otherwise, additional iterations are needed.

Shooting methods require few iterations if the final state of the circuit after one period is a near-linear function of the initial state. This is generally true even for circuits that react in a strongly nonlinear fashion to large stimuli (such as the clock or local oscillator) applied to the circuit. Because the circuit is assumed to be periodic over the shooting interval, then the stimulus must also be periodic over that same period. The shooting method integrates over a whole number of periods of the stimulus, which minimizes the nonlinearity in the relationship between the initial and final state and minimizes the number of iterations needed for convergence. Typically, shooting methods need about five iterations on an average circuit and have little difficulty with the strongly nonlinear behavior that occurs within the shooting interval. This is a strength of shooting methods over other steady-state methods such as harmonic balance [kundert90]. See "The Harmonic Balance Engine" on page 30 for information on the Spectre RF harmonic balance engine.

One reason why shooting methods are not commonly used is that they can become quite slow on larger problems. Shooting methods form a full $N \times N$ sensitivity matrix, where N is the number of equations needed to represent the circuit. During the course of the algorithm, this matrix must be inverted. This requires N^3 operations. For a large circuit, doubling the size of the circuit increases the simulation time by 8. The cubic rise in computational complexity as a function of circuit size makes this algorithm impractical for larger circuits.

Spectre RF simulation uses proprietary algorithms that eliminate the superlinear rise in computation time with problem size [telichevesky95]. Typically, the time required to directly

compute the periodic steady-state response of a circuit is about the same as the time required to compute 4 to 5 periods of the circuit's transient response. As a result, the PSS analysis is capable of directly finding the periodic steady state of very large circuits.

Distributed Components and Hidden State in PSS Analysis

PSS analysis does not support distributed components or components with hidden state. The difficulty with distributed components results from the fact that the state of a distributed component is a waveform, not a simple number, as with capacitors and inductors. To know the state of a distributed component, it is necessary to know the voltage and current along the whole length of the component. The sensitivity of the final state with respect to the initial state would be an infinite dimensional matrix. All of this greatly complicates the shooting method. This limitation prevents the use of ideal delays, transmission lines, microstrip lines, and nports with the PSS analysis.

The Transmission Line Modeler (LMG), which is available with Spectre RF, generates transmission line models that you can use in Spectre RF simulations. See the *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide* for more information.

The problem of components with hidden state results more from practical difficulties than from computational difficulties. Some components have an internal state that is hidden from the PSS analysis. Because the PSS analysis cannot access this hidden state, it cannot set it at the beginning of the period, nor can it compute the sensitivity of the final state to the initial state. Components that contain hidden state include the z-blocks and some components described using the Verilog-A$^®$ behavioral modeling language. Components described in the Verilog-A language that use values computed from a previous time step and saved in local variables to affect the behavior at a later time step are said to have hidden internal state. (Conversely, there is no problem if the only state is that held on internal or external nodes using the *integ* and *dot* operators.) You cannot use Verilog-A language components that use internal hidden state with the PSS analysis.

Fundamental Assumptions for PSS Analysis

There are two fundamental assumptions that apply to PSS analysis

■   Periodicity

■   Linearity

Understanding these two assumptions and their consequences allows you to anticipate whether you can apply PSS successfully to your problem.

Periodicity Assumption

PSS analysis assumes that during the shooting interval, all stimuli are periodic and that the circuit supports a T-periodic response, where T is the period specified for the PSS analysis. If the circuit is driven with more than one periodic stimulus, then the frequencies must all be commensurate or co-periodic, and T must be the common period or some integer multiple of the common period. Efficiency of the simulation drops when the T is long compared to the periods of the stimuli.

In rare cases, circuits with a periodic stimulus generate subharmonics, which PSS can handle if you set the period to be that of the subharmonic. In other cases, periodically driven circuits respond chaotically, such as do delta-sigma modulators. In this case, you must use transient analysis rather than PSS analysis.

If the period T used by PSS is not an integral multiple of the period specified for each *built-in* time-varying independent source present in the circuit, the simulator issues a warning message and skips the PSS analysis. If you use the Verilog-A language to describe independent sources, then *you* must ensure that sources are co-periodic with the analysis period T. Failure to do so generally prevents PSS analysis from converging, although PSS analysis rarely converges to an incorrect result.

If the circuit is driven by T-periodic stimuli but does not have a T-periodic solution (for example, if the solution is chaotic), PSS analysis fails to converge. In this case, you should use transient analysis to simulate the circuit. Occasionally a circuit generates subharmonics, as is the case with frequency dividers. In this situation, you should specify the PSS period to be equal to that of the subharmonics.

If PSS analysis fails to converge because the circuit does not have a T-periodic solution (which might result if the circuit is not driven by T-periodic stimuli), it is considered a user error, so you are responsible for finding and fixing the problem.

Linearity Assumption

In the circuit under simulation, the relationship between the initial and final points over the shooting interval should be near-linear. The more nonlinear the relationship between initial and final points, the greater number of iterations needed by the PSS analysis, which causes the PSS analysis to take longer. If the relationship is sufficiently nonlinear, PSS analysis might not converge.

If convergence is achieved, then there are no further consequences (no degradation of accuracy). Failing to converge in this situation is considered a failure of the simulator and should be reported to Cadence. (Send a description of the problem along with the input files [netlist, Verilog-A files, model files] and the log file to `spectrerf@cadence.com`). Occasionally delaying the starting time of the shooting interval can improve convergence.

Arrange for the shooting interval to start at a time when signals are quiescent or changing slowly.

Periodic Small-Signal Analyses

The conventional Spectre small-signal analyses (AC, SP, XF, and Noise) are useful for characterizing a wide variety of analog circuits [kundert95]. For example, you can use them to determine the frequency response, input and output impedances, loop gain characteristics, and supply rejection of circuits such as amplifiers and filters.

However, from an RF perspective, an important shortcoming of the Spectre small-signal analyses is that you cannot apply them to circuits that strongly exhibit frequency translation, either as a desired or inadvertent consequence of the design of the circuit. Commonly, you find these circuits in both analog and RF designs.

■   Circuits designed to translate from one frequency to another include

  ❏   mixers

  ❏   detectors

  ❏   samplers

  ❏   frequency multipliers

  ❏   phase-locked loops

  ❏   parametric oscillators

  Such circuits are commonly found in wireless communication systems.

■   Circuits that translate energy between frequencies as a side effect include

  ❏   oscillators

  ❏   switched-capacitor and switched-current filters

  ❏   chopper-stabilized and parametric amplifiers

  ❏   sample-and-hold circuits

You cannot simulate any of the circuits listed in the previous paragraphs with conventional small-signal analyses because they exhibit a nonnegligible amount of frequency translation. The conventional analyses begin by linearizing the circuit about a quiescent operating point (the DC solution). A linear time-invariant model of the circuit is then constructed by the small-signal analyses and the steady-state solution is computed using phasor analysis. These conventional analyses cannot handle frequency translation because they analyze a linear

time-invariant representation of the circuit, and linear time-invariant networks do not exhibit frequency translation.

Spectre RF simulation provides the *periodic small-signal analyses*: PAC, PSP, PXF, and Pnoise [telichevesky96b]. These analyses start by linearizing the circuit about the *periodically time-varying operating point* computed by a preceding PSS analysis. A periodically time-varying circuit does exhibit frequency translation, so the periodic small-signal analyses can accurately model frequency translation effects. Instead of using conventional small-signal analyses on amplifiers and filters, you can now make measurements on periodic circuits that exhibit frequency translation using the periodic small-signal analyses.

Applying a periodic small-signal analysis is a two-step process.

■   First, the small input or noise signals are ignored and a PSS analysis computes the periodic steady-state response to the remaining large signals (such as the clock or the LO).

   During the course of the PSS analysis, the circuit is linearized about the periodic large-signal operating point.

■   Then the subsequent periodic small-signal analyses use this periodic operating point to predict the response of the circuit to a small sinusoid at an arbitrary frequency. After you know the periodic large-signal operating point, you can perform any number of periodic small-signal analyses.

For the two-step process of applying the periodic small-signal analyses to be accurate, the input signals applied in the second step must be small enough so that the circuit does not respond to these signals in a nonlinear fashion in any significant way. This is not true of the signals applied in the first step. The only restriction on the large signals used in the initial PSS analysis is that they must be periodic.

This two-step process is widely applicable because most circuits that translate frequency are designed to react in a strongly nonlinear manner to one stimulus (the LO or the clock), while at the same time they react in a nearly linear manner to other stimuli (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 very wide dynamic range.

Because the periodic small-signal analyses are performed on a linear representation of the circuit, the computed response is a linear function of the stimulus, regardless of the size of the stimulus. In the real circuit, the input must be small enough to avoid violating the assumptions of the small-signal analysis. However, after the circuit has been linearized, the amplitude chosen for the small stimuli is arbitrary. Typically the stimulus is given an amplitude

of 1 and a phase of 0 (zero) to allow transfer functions to be computed directly. In this regard, the periodic small-signal analyses are similar to conventional small-signal analyses.

Conventional small-signal analyses do not model frequency translation, so between any input and output there is only one frequency-dependent transfer function. However, with the periodic small-signal analyses, there are many transfer functions between any single input and output. In fact, there are as many transfer functions as there are harmonics in the periodic operating point, which is always zero, one, or infinite.

In practice, usually only one or two transfer functions are interesting. For example, when analyzing the down-conversion mixers found in receivers, the desired transfer function is the one that maps the input signal at the RF to the output signal at the IF, which is usually the LO minus the RF. The Spectre RF simulator using the shooting method, unlike harmonic balance simulators, internally computes the response of the circuit in the time domain and converts the results into the frequency domain using the new Fourier Integral approach to Fourier analysis [kundert94].

The accuracy of the internal time-domain solution is completely independent of the number of harmonics requested. In addition, because Spectre RF analyses use the Fourier Integral rather than Discrete Fourier Transforms the accuracy of the frequency-domain results are also independent of the number of harmonics requested. With Spectre RF analyses, you are free to request as few or as many harmonics as desired without concern for the accuracy of the final result. The only exception to this general rule is the periodic noise analysis, Pnoise. Pnoise analysis models the noise folding that occurs in periodically varying circuits. If you request fewer harmonics, fewer folds are included in the accumulated total.

Another important advantage that the combination of the PSS analysis using the shooting method and the periodic small-signal analyses has over methods based on harmonic balance is that it is efficient even if the circuit is responding in a strongly nonlinear manner to the LO or the clock. As a result, you can apply Spectre RF analyses to such strongly nonlinear circuits as switched-capacitor filters, switching mixers, chopper-stabilized amplifiers, PLL-based frequency multipliers, sample-and-holds, and samplers [konrath96]. Note that the combination of the PSS analysis using the harmonic balance method and the periodic small-signal analyses efficiently analyze weakly nonlinear circuits.

Fundamental Assumptions for the Periodic Small-Signal Analyses

There are two fundamental assumptions that apply to the periodic small-signal analyses

■   Linearity

■   Frequency

Understanding these assumptions and their consequences lets you anticipate whether you can apply the periodic small-signal analyses to your problem.

Linearity

The periodic small-signal analyses all assume that the circuit responds linearly to the sinusoidal (PAC or PXF) or noise (PNoise) stimulus, or it involves both sinusoidal and noise (PSP) stimuli. There is no such assumption concerning the periodic signals (such as the LO or clock) applied in the initial PSS analysis. Spectre RF simulation is not capable of computing the distortion caused by the small signals, although you can use the small signals to measure distortion caused by the large signals present in the PSS analysis.

If the signals present in the circuit during a periodic small-signal analysis are large enough to drive the circuit to nonlinear behavior, then the results computed by simulation differ from the results produced by the circuit. If you consider PAC and PXF analyses to be computing transfer functions, which by definition assume arbitrarily small input signals, then this is not an issue. However, Spectre RF simulation can compute inaccurate results on strongly nonlinear circuits that exhibit very high levels of noise, such as when a circuit exhibits high levels of jitter (low-Q, highly nonlinear circuits such as relaxation and ring oscillators).

Analysis Frequency

Internally, the periodic small-signal analyses compute transfer functions using time domain techniques. The time-steps used in these time-domain computations are the same as those used in the preceding PSS analysis. In order for the transfer function to be accurate, the period of the small sinusoidal stimulus must be large compared with the largest time-step used during the PSS analysis.

If the analysis frequency of the periodic small-signal analysis is too high, the accuracy of the results degrade. The Spectre RF simulator warns you when it determines that you are requesting a frequency that is too high. You can use the *maxacfreq* parameter of the PSS analysis to specify the highest frequency that Spectre RF simulation can use in subsequent periodic small-signal analyses. PSS analysis then chooses the time-step in order to assure the results computed by the small-signal analyses are accurate. Specifying a very large *maxacfreq* causes both the PSS and small-signal analyses to run slowly and requires a large amount of memory.

Quasi-Periodic Steady-State Analysis

The Quasi-Periodic Steady-State (QPSS) analysis calculates the response due to multiple input frequencies. All of the input signals are treated in the same way as the PSS drive source

so that the calculated output includes all the intermodulation distortion effects caused by frequency translation of all harmonics of the input signals. You can analyze the response of a circuit to a sum of signal sinusoids with PSS analysis by including them as components of the PSS drive source, but the resulting sum must be periodic. Spectre RF now supports a choice of simulation engines between the shooting method and the harmonic balance method. See "The Harmonic Balance Engine" on page 30 for more information.

Because QPSS analysis allows arbitrary signal inputs, including sums of sinusoids that may not be periodic, you can think of it as a quasi-periodic extension of PSS analysis. You can also think of QPSS analysis as an extension of PAC analysis that allows signal inputs capable of producing third-order products.

Choose QPSS analysis when you need to compute the steady-state responses of a circuit driven by two or more signals at unrelated frequencies. Unlike PSS analysis, QPSS analysis does not require that multiple periodic stimuli be coperiodic. When you use QPSS analysis, you declare one signal as large. That signal can be a sinusoid or a pulse. Any additional-signals, called moderate signals, must be sinusoids. You must specify the number of harmonics to be simulated on all moderate tones.

Quasi-Periodic Noise Analysis

The Quasi-Periodic Noise (QPnoise) analysis is similar to the conventional Pnoise analysis, except that, in addition, it includes frequency conversion and intermodulation effects. You can use QPnoise analysis to predict 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.

In addition, QPnoise analysis also includes the effect of a quasi-periodically time-varying bias point on the noise generated by the various components in the circuit. 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.

Envelope Analysis

Envelope (ENVLP) analysis allows RF circuit designers to efficiently and accurately predict the envelope transient response of the RF circuits used in communication systems. For example, you can apply ENVLP analysis to efficiently and accurately analyze modulation signals in large communication circuits.

Important applications of ENVLP analysis include

■ Predicting the spectral regrowth of amplifiers or mixers

■ Designing feedback loops

■ Predicting the transient behavior of switched capacitor filters

■ Simulating large transients in phase lock loops

■ Helping the oscillator designer to identify the load pull effect for the communication systems with VCO and power amplifier

As a typical example, a designer might be interested in simulating a receiver signal path involving a mixer, in particular, to predict the spectral regrowth of the mixer. As shown in Figure 1-2, the inputs to a mixer can be one complete digital low-frequency (not necessarily periodic) modulation and one high frequency LO. The result of the mixing is a modulated high-frequency signal as shown in Figure 1-2. However, due to the nonlinearity of the mixer, unwanted harmonics might be generated and it is important to predict the signal level at these unwanted harmonics in order to validate the design.

Figure 1-2 Time-Domain Modulation
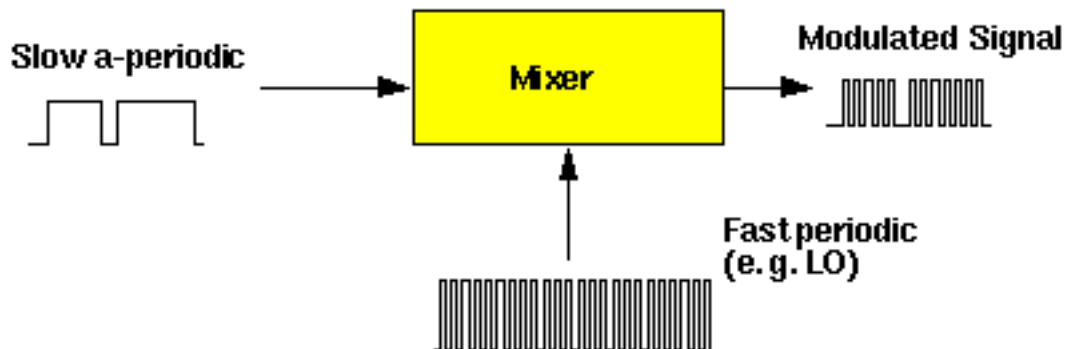
Figure 1-3 shows a typical scenario in the receiver signal path. Due to the nonlinearity of the mixer, it is important to predict the resulting spectral regrowth. Spectral regrowth is extraordinarily expensive to simulate using the traditional transient analysis because spectral regrowth requires a very long time interval to resolve the required frequency resolution.

Figure 1-3 Spectral Regrowth Simulated Using Envelope Analysis
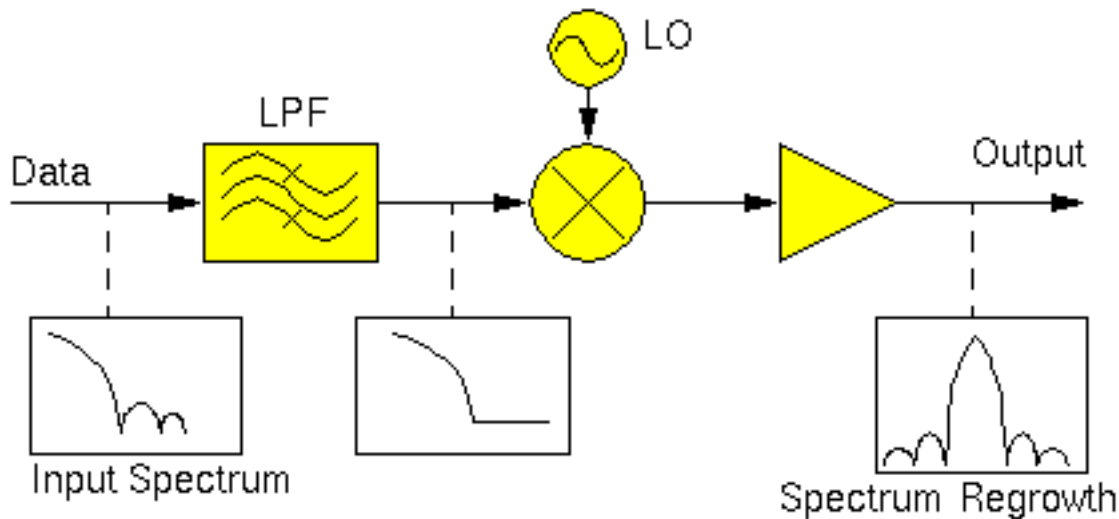
ENVLP analysis overcomes this difficulty with traditional transient analysis. ENVLP analysis reduces simulation time without compromising accuracy by exploiting the property that the behavior of a circuit in a given high frequency clock cycle is similar, but not identical, to its behavior in the preceding and following cycles. In particular, the envelope of the high-frequency clock by accurately computing the circuit behavior over occasional cycles, which accurately capture the fast transient behavior. The slow varying modulation is accurately followed by a smooth curve. As a result, you can obtain the spectrum of the circuit response by combining the spectrum of the smooth curve and the spectrum of occasional clock cycles.

The RF analysis types such as PSS (periodic steady-state analysis) [1] or QPSS (quasi-periodic steady-state analysis) [2] might not work directly because the modulation signal might be neither periodic nor quasi-periodic.

For switched-capacitor filters, the clock signal is referred to as the clock. For some other applications the clock signal might be referred to differently.

■   For mixers, the clock signal is called the LO

■   For detectors, the clock signal is called the carrier

The clock signal is normally the most rapidly changing signal in the circuit and thus causes the most nonlinearity.

Spectre RF now supports a choice of simulation engines between the shooting method and the harmonic balance method.

The Harmonic Balance Engine

The harmonic balance (HB) engine supports the frequency-domain harmonic balance analyses and complements the existing analyses which use the time-domain shooting method (TD) engine. The harmonic balance engine performs efficient and robust simulation of linear and weakly nonlinear circuits on all platforms for both 32 and 64 bit architectures.

The harmonic balance engine (HB) is available for

■ Driven PSS, PAC, PXF, PSP, and Pnoise analyses

■ Driven QPSS, QPAC, QPXF, QPSP, and Qpnoise analyses

■ Autonomous PSS, PAC, PXF, and Pnoise analyses

   These autonomous harmonic balance analyses are good candidates for simulating mildly-nonlinear oscillators with resonators, for example, LC oscillators, negative-gain oscillators, and crystal oscillators.

■ Envelope Analysis ENVLP

   For linear and weakly non-linear circuits, the harmonic balance Envlp analysis has better performance than the shooting method Envlp analysis.

■ The HB engine also supports corresponding swept harmonic balance analyses.

The HB engine is supported on the Solaris, Linux, HP, and IBM platforms for both 32 and 64 bit architectures.

HB Method Theory

In the time-domain method, the signal waveform $u(t)$ is solved according to Circuit Equation 1-1 where s is the source.

(1-1) $$i(u(t)) + \frac{d}{dt}q(u(t)) = s(t)$$

At the steady state, the solution $u(t)$, the current i, and the charge q are represented by harmonics as shown in Equations 1-2, 1-3 and 1-4 where ω is the fundamental frequency.

(1-2) $\quad u(t) = \sum_k U(k)e^{j \cdot k\omega \cdot t}$

(1-3) $\quad i(t) = \sum_k I(k)e^{j \cdot k\omega \cdot t}$

(1-4) $\quad q(t) = \sum_k Q(k)e^{j \cdot k\omega \cdot t}$

Coefficients $U(k)$, $I(k)$, and $Q(k)$ are Fourier transforms of $u(t)$, $i(t)$, and $q(t)$ respectively at the kth harmonic. Notice that $I(k)$ and $Q(k)$ are functions of $U(k)$ determined by $i(u)$ and $q(u)$.

Equivalent to the time-domain circuit Equation 1-1 on page 30, the circuit equation in the frequency-domain is shown in Equation 1-5 where $S(k)$ is the Fourier transform of periodic source s.

(1-5) $\quad I(k) + j \cdot k\omega \cdot Q(k) = S(k)$

The fundamental concept of the harmonic balance (HB) method is to solve for $U(k)$ (the frequency-domain waveform) according to Equation 1-5, rather than solving for $u(t)$ (the time-domain waveform) according to Equation 1-1. With the frequency-domain solution $U(k)$ obtained from Equation 1-5, the HB method uses Equation 1-2 to solve for the time-domain waveform $u(t)$.

The HB method is very efficient for simulating circuits such as low-noise amplifiers that have only low order harmonics. In these cases, the HB solution is small because only a few $U(k)$ coefficients are needed to give the waveform $u(t)$ accurately. Mixers with a low moderate-tone power level can also be represented with low order harmonics. In general, problems related to multi-tone simulation in QPSS can be reduced in scope or even eliminated by using the HB method.

The HB method also effectively supports frequency-dependent components. These are usually provided as frequency dependent scalar data such as in an S-parameter format file.

HB Convergence Criteria

HB is designed to solve nonlinear equations. For example, you might have the equation

(1-6) $\qquad f(x) = 0$

which can be solved by using the Newton method:

(1-7) $\qquad x_{k+1} = x_k - \left[J_f(x_k) \right]^{-1} f(x_k)$

When the following two conditions are met the iterations are assumed to have converged.

(1-8) $\qquad \left| f(x_{k+1}) \right| < tolerance\_residual$

(1-9) $\qquad \left| x_{k+1} - x_k \right| < tolerance\_delta$

To understand Spectre RF HB convergence, it is useful to know of the following two norms:

■   `Resd_Norm`

The `Resd_Norm` measures

(1-10) $\quad \left| f(x_{k+1}) \right|$

by taking on the value

(1-11) $Resd\_Norm = \left| Cr(f(x_{k+1})) \right| / tolerance\_residual$

in which *Cr* is a function designed to get the relative errors.

Equation 1-11 on page 33 gives a rough determination of the relationship between *Resd_Norm* and *tolerance_residual*.

■ Delta_Norm

The Delta_Norm measures

(1-12) $\left| x_{k+1} - x_k \right|$

by taking on the value

(1-13) $Delta\_Norm = \left| Cd(x_{k+1} - x_k) \right| / tolerance\_norm$

in which *Cd* is a function designed to get the relative errors.

Equation 1-13 on page 33 gives a rough determination of the relationship between *Delta_Norm* and *tolerance_delta*.

In the HB Newton method, iterations are assumed to have converged when

(1-14) $Resd\_Norm < 1$

and

(1-15) $Delta\_Norm < 1$

The *tolerance_residual* is determined by the product of the `reltol` and `residualtol` parameters.

The *tolerance_delta* is determined by the product of the `reltol`, `lteratio`, and `steadyratio` parameters.

Occasionally, *Resd_Norm* becomes smaller than 1 but *Delta_Norm* remains large. To achieve convergence in a case like this, consider setting a bigger `steadyratio` value so that the *tolerance_delta* is larger. This approach is recommended because

■   The infinity norm is defined as

(1-16) $|x|_\infty = Max(x_1, x_2, ..., x_n)$

so that the norm of the vector x depends on its maximum element.

■   In HB, the high order harmonics of some nodes are occasionally quite small. During iterations, these small high-order harmonics vary greatly, which makes *Delta_Norm* large. Because these harmonics contribute little to the whole circuit state, you can set a larger steadyratio, which solves the problem.

HB Large Signal Simulation for Driven Circuits

The following parameters are provided to specify the HB method for large signal simulations of driven circuits.

Basic HB Parameters for Driven HB Analysis

flexbalance=no	The HB engine shares the same PSS and QPSS analysis statement with the time-domain engine. To run the harmonic balance engine, set **flexbalance**=yes in the analysis statement, or in the graphical user interface, select *Harmonic Balance* for *Engine*.
harms and **maxharms**	Determines how many harmonics are used to expand the waveform in Equation 1-2.
	In PSS, specify the maximum harmonic with **harms**. In QPSS, specify the maximum harmonic for each tone with **maxharms**.
	It is important to point out that **harms** in PSS and the first value in the **maxharms** vector in QPSS are output parameters in the time-domain method but they are input parameters in the HB method and have a direct impact on the accuracy and performance of the HB simulation.
	The best choice for the **harms** or **maxharms** value depends on the signal waveform and circuit nonlinearity. The faster the signal varies with time, or the more nonlinear the circuit, the more harmonics are required to accurately represent the solution. In multi-tone mixers, the large LO tone has a higher power level than the moderate RF tones and causes more nonlinear effects, so you should usually use more harmonics for the large LO tone than for the moderate tones.
	The best **harms** or **maxharms** value also depends on the order of nonlinear effect that you want to study. For example, for a mixer IP3 measurement, the **maxharms** value for the moderate tones must be 2 or more to capture the IM3 mode at frequency $2\omega_1-\omega_2-\omega_{LO}$.

Basic HB Parameters for Driven HB Analysis, *continued*

tstab	**tstab** is a valid parameter for the initial transient analysis in HB. The default **tstab** value for both PSS and QPSS is one cycle of the signal period. For QPSS analysis, you can choose a specific tone for the **tstab** integration and only one tone is allowed. One additional cycle is used for the FFT. When you set **tstab** to 0, the DC results are used as the initial condition for HB.

The following are example HB statements.

```
pss0 pss fund=1G harms=10 flexbalance=yes

qpss0 qpss funds=["LO" "RF1" "RF2"] maxharms=[5 3 3]
flexbalance=yes
```

Optional HB Parameters for Driven HB Analysis

oversamplefactor In the HB method the nonlinear parts of functions $i(u)$ and $q(u)$ are evaluated at a series of sample time points $t_n \cdot I(k)$ and the $Q(k)$ values are obtained from Fourier transforms of $i(t_n)$ and $q(t_n)$. In a regular Fourier transform, t_n equals the number of harmonics N_k. For highly nonlinear functions $i(u)$ or $q(u)$, if $i(t)$ or $q(t)$ is under-sampled by t_n, aliasing effects cause inaccuracy in $I(k)$ or $Q(k)$. Aliasing can be prevented by increasing N_k but this also increases the number of unknown $U(k)$. Another way to prevent aliasing is to use the same N_k but to oversample $i(t)$ and $q(t)$ by $m \cdot N_k$ time points, where the integer m is the oversample factor. Notice that although the oversampling method does not change the solution size, it does increase the size of the Fourier transform and the number of evaluations of $i(t_n)$ and $q(t_n)$.

In the PSS or QPSS analysis statement, use **oversamplefactor**=m to specify the oversample factor. Spectre RF oversamples $i(u)$ and $q(u)$ by a factor of m for each tone. As a result, the size of the Fourier transform is increased by $m^{number\text{-}of\text{-}tones}$. For multi-tone cases, you can also specify **oversample**=$[m_1, m_2, m_3, ...]$ to oversample the ith tone by a factor of m_i. The size of the Fourier transform is therefore increased by $m_1 \cdot m_2 \cdot m_3 \cdot ...$. The default oversample factor is 1.

Note that even with oversampling, you must make sure the maximum harmonic meets the accuracy requirement on results of interest.

Optional HB Parameters for Driven HB Analysis, *continued*

itres	Controls the accuracy of the linear solution at each Newton update. It controls the convergence of the Newton iteration. The value is between [0, 1] and the default is 0.9. When you use the HB method for highly nonlinear circuits, Cadence recommends using the default value. For nearly linear circuits, using a small **itres** (for example,$1e^{-2}$) can speed convergence.
maximorder and **selectharm**	These two parameters determine the harmonics selected for multi-tone QPSS analysis. By default, Spectre RF uses a square cut determined by **maxharms**. A QPSS simulation with M tones will solve for all frequencies $\omega = k_1\omega_1 + k_2\omega_2 + ...$ where $k_i \leq H_i$ for each tone and H_i is the maximum harmonic for tone i. For a diamond cut $\sum k_i \leq P$ where P is the maximum order set by the **maximorder** parameter. The **selectharm** parameter determines the type of cut. The options for cut are: **box**, **diamond**, **axis**, or **funnel**.

> ■ The default cut is **box**.
>
> ■ A **diamond** cut selects a diamond cut with order P.
>
>   **Note:** The diamond cut must not be used in a QPSS analysis that is followed by a QPAC, QPXD, QPNOISE, or QPSP analysis because some sidebands are not included in the small signal calculation.
>
> ■ An **axis** cut selects harmonics along an axis. This means that there are no intermodulation harmonics, there are only harmonics of type $\omega = k_1\omega_1$ where $k_i \leq H_i$.
>
> ■ A **funnel** cut combines the **axis** and **diamond** cuts. The **funnel** cut selects harmonics along an axis and intermodulation harmonics of order P or less.

freqdivide	The frequency division ratio of a large signal in QPSS analysis. The default is 1. Specify **freqdivide** when the circuit has a frequency divider on a large signal.

The ADE Use Model

All the analysis features mentioned above are integrated into the Analog Design Environment (ADE). You can switch between the time domain shooting (TD) engine and the HB engine in the PSS, QPSS, and ENVLP Choosing Analyses forms. For example, Figure 1-4 shows the Choosing Analyses form set up for a large signal HB Quasi-Periodic Steady State Analysis. Notice the *Engine* field choices labeled *Shooting* and *Harmonic Balance*.

Figure 1-4 The HB QPSS Choosing Analyses Form

For a detailed design example, refer to Lab 10, *IP3 Calculation* (QPSS with Shooting or Harmonic Balance Engine), in the workshop *Mixer Design Using SpectreRF*. The database for this workshop is available at

```
<instdir>/tools/spectre/examples/SpectreRFworkshop
```

HB Large Signal Simulation for Autonomous Circuits

The harmonic balance (HB) engine is supported for oscillators and phase-noise simulation. HB is a frequency domain technique that solves the spectrum of each node voltage unlike the time-domain engines (such as shooting PSS) which solve the waveform of each node voltage over the period.

The harmonic balance method is a good candidate for simulating mildly-nonlinear oscillators with resonators, such as LC oscillators, negative-gain oscillators, and crystal oscillators. The time-domain shooting PSS method is a good candidate for simulating strongly-nonlinear resonatorless oscillators, such as ring oscillators, relaxation oscillators, or oscillators containing digital control components.

Basic HB Parameters

The following parameters are provided to specify the HB method for autonomous circuit large signal simulations.

Basic Parameters for Autonomous HB Analysis

flexbalance	Harmonic balance engine shares the same analysis form with time-domain engine and is invoked by setting flag **flexbalance**=yes in the analysis statement.
harms	Determines how many harmonics will be used to expand the waveform in eq. (2). Similar to that in PSS analysis for driven circuit, maximum number of harmonics is specified by **harms**. It is input parameter in HB method and has direct impact on the accuracy and performance of the simulations.
oscmethod	Currently, there are two methods implemented: **onetier** and **twotier**. They are set by parameter **oscmethod**. In **onetier** method, the frequency and voltage spectrum are solved simultaneously in one single set of nonlinear equations. In **twotier** method, the nonlinear equations are split into two sets: the inner set of nonlinear equations solves the spectrum of node voltage equations; the outer set of nonlinear equations solves the oscillation frequency. Physically it is equivalent to add a sinusoidal voltage probe to a pinning node and adjust its frequency and amplitude until it has no effect on the oscillator. Spectre RF automatically chooses the pinning node. By default, Spectre RF runs **onetier** first for n iterations; if necessary, it runs **twotier** for next n iterations (n is set by **maxperiods**). Users can also run only **onetier** or **twotier** by specifying **oscmethod**. **Twotier** has larger convergence zone as its convergence is slightly more robust. This method is however computationally more expensive than **onetier** method.

Basic Parameters for Autonomous HB Analysis, *continued*

tstab	Specifies the length of the transient analysis used to assist the harmonic balance analysis. Spectre RF runs a transient analysis of the length specified by **tstab** and then switches to HB. Stable oscillation in the transient analysis can help HB to converge. Stable oscillation can be achieved by

❑ Setting an initial condition for a particular node (for example: `ic net01=5.0`)

❑ Setting an initial condition for the inductor or capacitor in the resonator in OSC (for example: `L14 (Xtal02 net01) inductor l=6.m ic=0.5m`)

❑ Adding a damped current source in parallel to the resonator in OSC (for example: `Ikicker (net01 net02) isource type=sine freq=1.0G ampl=1m damp =1.0G`)

❑ Adding a voltage pulse (for example: `vdd (vdd 0) vsource type=pulse val0=0.0 val1=3.3 rise=1n`)

Optional Parameters for Autonomous HB Analysis

oversamplefactor	The usage of **oversamplefactor** parameter is similar to the one in driven HB analysis. See "**oversamplefactor**" on page 36.

The ADE Use Model

All the analysis features mentioned above are integrated into the Analog Design Environment (ADE). You can use the *Engine* field to switch between the time domain shooting (TD) engine and the HB engine in the PSS, QPSS, and ENVLP Choosing Analyses forms. For example, Figure 1-5 shows the Choosing Analyses form for a large signal HB autonomous PSS Analysis.

Figure 1-5 The Autonomous HB PSS Choosing Analyses Form

For a detailed design example, refer to Lab 1, *Phase Noise Simulation with Shooting or Harmonic Balance Engine*, in the workshop *VCO Design Using SpectreRF*. The database for this workshop is available at

`<instdir>/tools/spectre/examples/SpectreRFworkshop.`

HB Small Signal Setup

HB small signal analysis runs automatically if the large signal analysis uses the HB method. Hence, HB small signal analyses do not require special setup and use the same statement as a time-domain small signal analysis. The only difference is that the **maxsideband** setting is ignored in HB small signal analysis when the value is larger than the **harms** and **maxharms** value of the large signal analysis.

There is no specific small signal set up for HB in ADE.

HB Envelope Analysis

See The Time Domain and Harmonic Balance ENVLP Algorithms on page 278 for information about the HB ENVLP analysis.

The ADE Use Model

All the analysis features mentioned above are integrated into the Analog Design Environment (ADE). In the Choosing Analyses form, two buttons are available for you to switch between the time domain shooting (TD) engine and HB engine in the ADE PSS, QPSS and ENVLP Choosing Analyses forms. Figure 1-5 shows the HB Envlp GUI in ADE.

All the analysis features mentioned above are integrated into the Analog Design Environment (ADE). You can use the *Engine* field to switch between the time domain shooting (TD) engine and the HB engine in the PSS, QPSS, and ENVLP Choosing Analyses forms. For example, Figure 1-5 shows the Choosing Analyses form for an HB Envlp analysis.

Figure 1-6 The HB ENVLP Choosing Analyses Form

The following figure shows the multi-carrier HB envelope analysis set up in ADE.

Figure 1-7 The HB ENVLP Multi-Carrier Choosing Analyses Form

For a detailed design example, refer to Lab 6, *Envelope Following Analysis*, in workshop *PA Design Using SpectreRF*. The database for this workshop is available at

`<instdir>/tools/spectre/examples/SpectreRFworkshop.`

Multi-Thread Acceleration for Harmonic Balance Analysis

You can speed up HB large signal and small signal simulations if you have multiple CPUs. The earlier environment variable `CDS_SPECTRERF_MULTITHREAD` is no longer supported. Now, you can simply use the Spectre option `+mt=`.

For example, if you have the number of CPUs as 4, you can give:

`spectre +mt=4 input.scs`

Distributed Component Support

The distributed component models, which are usually provided in S-parameters scalar format, are directly computed in the frequency domain for HB analysis. The Spectre `nport` primitive is supported. If you specify the *rational fitting* method for the `nport`, an equivalent circuit is still used in HB and it might cause convergence issues due to an unstable model of the rational fitting method. Consequently, Cadence recommends using the default fitting method `spline` or `linear`.

Other distributed components such as `mtline` and `delay` are also supported.

Hidden State Issue

The fundamental hidden state issue in Spectre RF still exists with the harmonic balance engine. Hidden states are not exposed to solution, so it is difficult to determine how sensitive other signals are to changes in those hidden states. This is required for predicting new iterations.

Frequently Asked Questions and Answers

What are the parameters that affect accuracy in HB?

Maximum harmonic (**harms** in PSS and **maxharms** in QPSS) has the most impact on the accuracy of HB analysis. When you use too few harmonics, the spectrum outside of the maximum harmonic is folded back into harmonics inside by aliasing effects, causing errors. To obtain accurate results, set the maximum harmonic large enough to cover the signal bandwidth of interest.

The **reltol** or **errpreset** parameters also affect simulation accuracy. HB uses the same convergence criteria as the TD shooting method.

Spectre RF offers both time-domain shooting and HB options. Which one should I use for my circuits?

The HB method is very efficient for simulating weakly nonlinear circuits such as LNAs. Only a few harmonics are needed to accurately represent the solution. For highly nonlinear circuits with sharply raising or falling signals, the TD shooting method is usually more appropriate. However, HB might still have an advantage when exploring design trade-offs using a few harmonics where accuracy is not the primary concern.

For multi-tone cases such as mixers, HB is significantly more efficient than the shooting method with QPSS analysis due to its natural representation of circuit equations.

The HB method is better than the TD method for handling frequency-dependent components. In post-layout circuits with many linear elements, HB has better performance than TD shooting.

Should I use HB PSS or HB QPSS for circuits driven by multi-tone stimulus?

Because HB multi-tone simulation does not have the convergence and speed issues encountered in TD shooting QPSS analysis, use HB QPSS analysis to simulate multi-tone circuits. HB PSS analysis using the *beat frequency* as the fundamental is very inefficient for multi-tone cases. When source frequencies are closely spaced, their common frequency is so low that you must use hundreds or even thousands of harmonics. In this situation, you should use QPSS analysis.

When should I use oversample factor?

In general, oversampling is not needed. However, for extremely nonlinear circuits driven by sources at a very high power level, oversampling might help with convergence. If the circuit has a frequency divider on the large signal, you might want to specify the oversample factor the same as the divide ratio.

How should I choose the number of harmonics for an HB envelope analysis?

The default value is 3 harmonics. The number of harmonics you choose depends on the linearity in one cycle of the fast signal (`LO` or `clock`). For linear cases, even 1 harmonic is enough for accurate results. However, for nonlinear cases, such as a circuit with a square clock, you might need 15 or even more harmonics.

In HB autonomous analysis, why would I set the OSC Newton Method rather than letting the simulator take care of it?

The default value for `oscmethod` is `onetier`. If Spectre RF fails to converge then it automatically switches to the `twotier` method. In these cases it might be more efficient to use the `twotier` method at the beginning by specifying `oscmethod=twotier`.

Perturbation Based Measurements

The perturbation-based approach solves weakly nonlinear circuits based on Born approximation. The perturbation method does not require explicit high order device derivatives and Its implementation is straightforward and is equivalent to successive small signal analysis. We analyze convergence properties for perturbation-based analyses and discuss the connection between the perturbation method and Volterra series.

Rapid IP2 and IP3 measurements are calculated with 2nd and 3rd order Born approximation under weakly nonlinear conditions. Using the diagrammatic representation of nonlinear interaction, both Volterra series and Born approximation can be constructed in a systematic way. The method is generalized to calculate other high order nonlinear effects such as the IMn product. All equations are formulated as RF harmonics and they can be implemented in both the time and frequency domains.

Specialized PAC and AC Analyses for Measuring Distortion

Spectre RF supplies four specialized PAC and AC analyses based on perturbation methods that provide a basic set of rapid distortion measurements

■   Compression Distortion Summary

■   Rapid IP3

■   IM2 Distortion Summary

■   Rapid IP2

These four perturbation-based analyses characterize intermodulation distortion and compression distortion for RF circuits such as mixers and amplifiers.

Distortion is an important issue in RF circuits such as mixers, low-noise amplifiers (LNAs) and power amplifiers (PAs). Compression and intermodulation distortion are the two aspects of distortion that are of greatest concern. At high frequencies, and particularly with narrowband circuits, it is common to characterize the distortion produced by a circuit in terms of a compression point (CP) or an intercept point (IP). These metrics characterize the circuit rather than the signal, and as such it is not necessary to specify the signal level at which the circuit is characterized.

About the IM2 Intermodulation Distortion Summary

The 3rd order intermodulation distortion occurs when input signals at frequencies f_1 and f_2 mix together to form the response at $2f_1 - f_2$ and $2f_2 - f_1$. When f_1 and f_2 are close enough

in frequency, the intermodulation products $2f_1 - f_2$ and $2f_2 - f_1$ will fall within the bandwidth of the circuit and will interfere with the input signal.

Intermodulation distortion is typically measured in the form of an intercept point. As shown in Figure 1-8, you can determine the 3rd order intercept point (IP3) by plotting the power of the fundamental and the 3rd order intermodulation product (IM3) versus the input power. You should plot both input and output power in some form of dB. Thus, when you measure IP3, the fundamental power curve is extrapolated from where the curve has a slope of 1 over a broad region. The 3rd order intermodulation product is extrapolated from a point where its curve has a slope of 3 over a broad region. Extrapolate both curves from a low power level and identify the intercept point as the point where they cross.

For Rapid IP3 analysis in Spectre RF, the IP3 is obtained at an input power where the user is confident of 1st order and 3rd order linearity.

Figure 1-8 IP3 Measurement
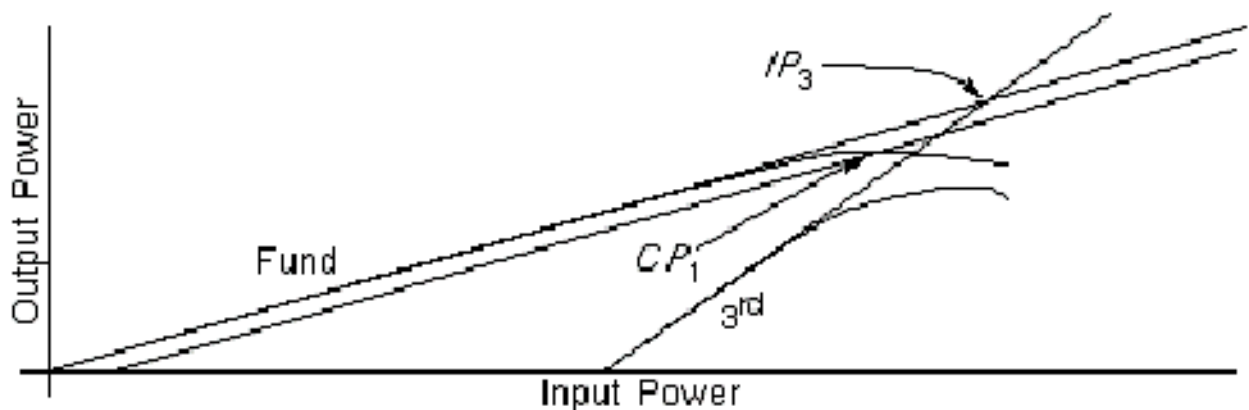
For some applications, such as direct conversion mixers, IP2 is important to characterize the circuit for the 2nd order intermodulation products (IM2) $f_1 - f_2$ or $f_2 - f_1$ which will fall within the bandwidth of the output signal. The definition of IP2 is similar to IP3. The difference is that the 2nd order intermodulation product is extrapolated from a point where its curve has a slope of 2.

For Rapid IP2 analysis in Spectre RF, the IP2 is obtained at an input power where you are confident of the 1st order and 2nd order linearity.

Rapid IP2 analysis produces the value of IP2 for the whole circuit, while the IM2 Distortion Summary analysis produces a list of the various distortion contributors and how much distortion they contribute to the output. That is, IM2 Distortion Summary computes the

contribution of each nonlinear device to the total IM2. The result is accurate because there is no presumption in computation.

About the Compression Distortion Summary

Consider the circuit in Figure 1-9. The system has a time-dependent operating point $V_0(t)$ at frequency $\omega\text{-}_{LO}$ (for an LNA or PA, the operating point is time-independent, so $\text{-}\omega_{LO} = 0$).

Figure 1-9 A Simple Circuit Representation
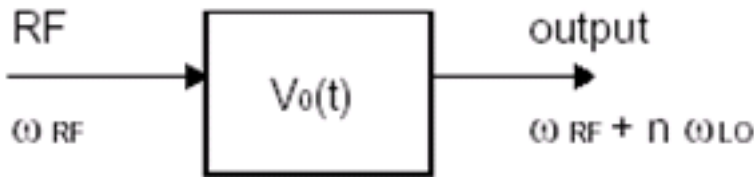
As shown in Equation 1-17, when the RF input signal is applied, the output signal at $\text{-}\omega_{RF} + n\text{-}\omega_{LO}$ is a function of input magnitude V_{RF}.

(1-17)
$$V_{out}(\omega_{RF} + n\omega_{LO}) = c_1 V_{RF} + c_3 V_{RF}^3 + \dots$$

The first term is the linear response to the RF input. Coefficient c_1 can be obtained by small signal analysis. Other terms are distortion due to third and higher order nonlinear response. (For the Compression Distortion Summary analysis in Spectre RF, the highest nonlinear term concerned is the third order.) The total distortion, as represented in Equation 1-18, is measured in dB.

(1-18)
$$dis = 10 \cdot \log\left(\frac{V_{out}}{c_1 V_{RF}}\right)$$

While you want to know the contribution of each individual device to the total distortion, mathematically it is impossible to make such separation for third or higher order nonlinearity. To quantify the effect of nonlinear response at a particular device on the output signal, the Compression Distortion Summary assumes that the rest of the circuit responds to the RF signal linearly and computes the distortion in the output caused solely by the nonlinear

response at this device. With this approach, although the circuit is hypothetical, the result provides useful information about how sensitive the device is to distortion. Effects of the nonlinear response of the device and the transfer function from the device nodes to output nodes are reflected in the calculation. What is missing is the interaction between the devices.

It is important to point out that since the operating point is time-dependent, all the other devices are still nonlinear in the calculations. Only their response to the RF signal is linearized. In particular, for mixer cases, transistors in the LO block still function nonlinearly to generate the clock signal. However, their contribution to output distortion is expected to be very small in the distortion summary because the RF signal usually does not disturb the LO part of the circuit. Both linear and nonlinear responses in the LO block are almost zero.

About the Perturbation Method

SpectreRF supplies four specialized PAC analyses to solve weakly nonlinear circuits based on the perturbation approach.

■ Rapid IP3

■ Rapid IP2

■ IM2 Distortion Summary

■ Compression Distortion Summary

In previous versions of SpectreRF, you could use either QPSS-based or QPAC-based methods to calculate IP3 for systems containing a mixer and a LO. In the QPSS-based method, three-tone QPSS analysis with LO, RF1 and RF2 tones at the frequencies ω_{LO}, ω_{RF1} and ω_{RF2} respectively, is run at a given RF power level. IM3 of harmonic $2\omega_{RF1} - \omega_{RF2} - \omega_{LO}$ is obtained from the solution. Assuming RF power is low enough and IM3 is dominated by leading order V_{RF}^3 terms, $log(V_{IM3})$ is expected to be a linear function of $log(V_{RF})$ with a slope of 3. IP3 is then extrapolated from V_{IM3}. Here V_{IM3} and V_{RF} are amplitudes of the IM3 and RF signals, respectively. The QPSS-based method requires very high accuracy to accommodate large dynamic range between RF and LO signals because they are mixed in the same solution vector. For large circuits, the QPSS-based method also relies on speed and convergence of the multi-tone QPSS analysis.

In the QPAC-based method, a two-tone QPSS analysis at frequencies- ω_{RF1} and ω_{-LO} is run first. Then RF2 input is included as a small signal by the QPAC analysis to calculate IM3 at $2\omega_{-RF1} -- \omega_{RF2} - \omega_{LO}$. As in the QPSS-based method, the QPAC-based method also has to cover dynamic range between RF1 and LO and depends on convergence of the two-tone QPSS analysis.

Compared to the QPSS-based approach, the QPAC-based approach reduces computation from a three-tone QPSS analysis to a two-tone QPSS analysis plus a QPAC analysis by

applying first order perturbation to the RF2 signal. You can further reduce the amount of computation by treating both RF signals as perturbation to the steady-state operating point at the LO frequency with zero RF input. In this way, leading order intermodulation between the RF1 and RF2 signals in IM3 can be computed directly from third-order perturbation. Since no multi-tone large signal simulation is needed, this can dramatically improve efficiency and avoid the convergence problem of multi-tone simulation.

Now SpectreRF can use a perturbation-based approach to solve weakly nonlinear circuits based on Born approximation. The perturbation-based approach does not require explicit high order derivatives from the device model. All equations are formulated in the form of RF harmonics that can be implemented in both the time and frequency domains.

For a nonlinear system, you can express the circuit equation as in Equation 1-19.

(1-19) $L \cdot v + F_{NL} v = \varepsilon \cdot s$

in this equation

■ The first term is the linear part

■ The second term is the nonlinear part

■ s is the RF input source

■ ε is introduced to track the order of the perturbation expansion

Under weakly nonlinear conditions, the nonlinear part of Equation 1-19 is small compared to the linear part. This allows you to solve the equation using Born approximation iteratively as shown in Equation 1-20.

(1-20) $u^{(n)} = v^{(1)} - L^{-1} \cdot F_{NL}(u^{(n-1)})$

In Equation 1-20, $u^{(n)}$ is the approximation of v and it is accurate to the order of $O(\varepsilon^n)$.

Since the evaluation of F_{NL} takes full nonlinear device evaluation of F and its first derivative, no higher order derivatives are needed. This allows you to carry out higher order perturbation without modifying your current device models. Also, the dynamic range of perturbation calculations covers only RF signals. This gives the perturbation method advantages in terms of accuracy.

Frequently Asked Questions and Answers

How do I select the input power for perturbation-based measurements?

The perturbation method works best under weakly nonlinear conditions. High input power which induces most nonlinearity will produce an unreasonable result. Input power that is too low will blend with the numerical noise floor. A recommended input power range for general circuits is -50 dBm to -20 dBm.

Why is dB the unit used in the Compression Distortion Summary while Volts is the unit used in the IM2 Distortion Summary?

Compression distortion evaluates distortion as the ratio of total output (linear and nonlinear) to the linear output. IM2 Distortion Summary computes the contribution of each nonlinear device to the total IM2.

How do I measure the IP2 of a double balanced mixer?

It is well known that the IP2 of an ideal double-balanced mixer is infinite. In practice, IP2 is a function of the matching between two branches. To measure IP2, add a small amount of mismatch, such as 2%, to the transistors and resistors between the two branches. Then run a worst case simulation.

Why should I set the RF input source to DC?

The perturbation method is a nonlinear small signal analysis. The analysis treats the RF signal as the small signal which is defined by `pacmag` (or `pacdbm`). If you set up the RF signal to be sinusoidal (or some other type of large signal), the PSS and small signal results will be affected.

Why might the Rapid IP2 result seem inaccurate, inconsistent, or just plain wrong?

Rapid IP2 is recommended to measure IP2 for direct-conversion mixers. For ordinary LNAs or mixers, the IM2 frequency is out of the work frequency band and the Rapid IP2 measurement produces a less meaningful result.

Why do the shooting and flexible balance PSS simulation engines produce different results?

Usually, the shooting and flexible balance engines produce results that match well. You can ensure the results will match by setting enough harmonics when you use the flexible balance engine.

Why do I see the error "More than one frequency matches are found…?"

For perturbation-based measurement, incommensurate frequencies should be used for all tones. If multiple combinations of tone frequencies match either *Frequency of Linear Output Signal* or *Frequency of IM Output Signal*, you will see this error message because perturbation-based measurement can't figure out which frequency you want to use as IM1, IM2 or IM3. In other words, each IM1, IM2 or IM3 frequency should be determined by a unique set of

k1 * flo + k2 * frf1 + k3 * frf2

respectively. To avoid this problem, adjust the values of *frf1* and *frf2* slightly.

References

[1]   Behzad Razavi. RF Microelectronics, Prentice Hall, c1998. Series: Prentice-Hall Communications Engineering & Emerging Technologies Series.

[2]   Thomas H. Lee. The Design of CMOS Radio-Frequency Integrated Circuits. Cambridge University Press, 1998.

[3]   Ken Kundert, "Accurate and Rapid Measurement of IP2 and IP3", `www.designers-guide.org`

Large Signal S-Parameters

Characterizing RF circuits with small-signal S-parameters is a well established practice. However, small-signal S-parameters are not sufficient to describe both strongly nonlinear circuits and circuits that exhibit frequency translation. This is especially true for designs that use power amplifiers and mixers.

As a natural extension of small-signal S-parameters, large-signal S-parameters (LSSPs) are defined as the ratio of reflected (or transmitted) waves to incident waves. Since small-signal S-parameters are based on the simulation of a linearized circuit, they are independent of input power. This description illustrates how to measure LSSPs using Spectre and SpectreRF in the Analog Design Environment (ADE).

LSSPs are based on large-signal steady state simulation techniques such as SpectreRF's PSS analysis with both its time domain shooting Newton (TD) engine and it's harmonic balance (HB) engine. LSSPs are sensitive to input power levels.

For multi-port circuits, you can represent

■ The reflected wave, B_i at port i

■ The incident wave, A_i at port i

■ The frequency $k\omega$ (harmonic k)

by

■ The port voltage, V_i

■ The current, I_i

■ The termination impedance, R_i

The multi-port circuit representation is described as follows.

The reflected wave, B_i, at port i, and the frequency $k\omega$ (harmonic k) are characterized by Equation 1-21.

(1-21) $$B_i(k\omega) = \frac{V_i(k\omega) + R_i I_i(k\omega)}{2\sqrt{R_i}}$$

The incident wave, A_i, at port i, and the frequency $k\omega$ (harmonic k) are characterized by Equation 1-22.

(1-22) $$A_i(k\omega) = \frac{V_i(k\omega) - R_i I_i(k\omega)}{2\sqrt{R_i}}$$

The LSSPs between port i with harmonic m and port j with harmonic n, are defined as in Equation 1-23.

(1-23) $$S_{ij}(m\omega, n\omega) = \frac{B_i(m\omega)}{A_j(n\omega)}$$

Where $A_i(k\omega) = 0$, for all ports other than j and for harmonics other than n.

Large-Signal S-Parameters for a Two-Port Circuit

Consider calculating LSSPs for a two-port circuit. When you apply a source at port 1 and terminate port 2, as shown in Figure 1-10, you can measure S_{11} and S_{12} at a given input power P_1.

Figure 1-10 Test Setup to Measure the Large-Signal S-Parameters S11 and S21
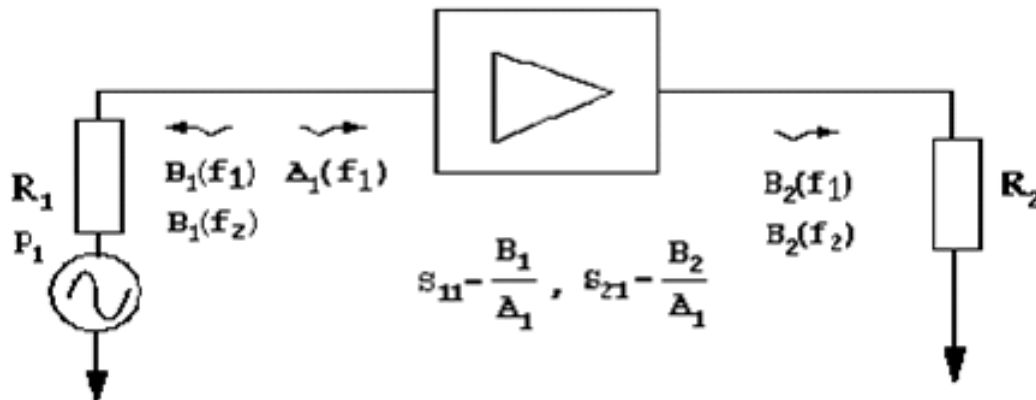
Similarly, when you apply a source at port 2 and terminate port 1, as shown in the Figure 1-11, you can measure two more LSSPs, S_{12} and S_{22}.

Figure 1-11 Test Setup to Measure the Large-Signal S-Parameters S12 and S22
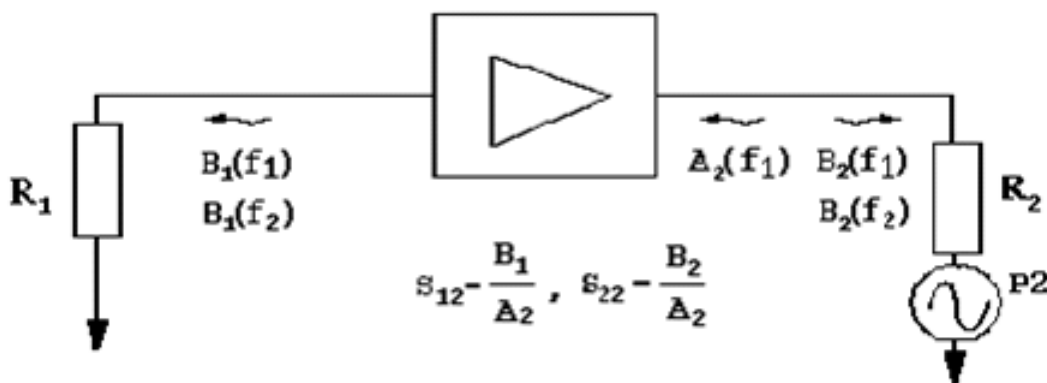
To be consistent with the input power level measured for S_{11} and S_{21}, you can compute the power applied to port 2 in this stage using the relationship in Equation 1-24.

(1-24) $$P_2 = |S_{21}|^2 P_1$$

References

[1]   V. Rizzoli, A. Lipparini, F. Mastri, "Computation of Large-Signal S-parameters by Harmonic-Balance Techniques," in *Electronics Letters*, Vol. 24, Issue 6, pp. 329-330, 17 Mar 1988.

[2]   Jan Verspecht, Frans Verbeyst, Marc Vanden Bossche, "Network Analysis Beyond S-parameters: Characterizing and Modeling Component Behavior under Modulated Large-Signal Operating Conditions," in *56th ARFTG Conference Proceedings*, Broomfield, Colorado, USA, December 2000.

Using the PSTB and STB Analyses with Linear Periodic Time-Varying Circuits

The Spectre stability analysis (STB) rapidly evaluates the stability of feedback circuits. As is true for all small signal analyses, the circuit under analysis must first be linearized about a DC operating point before the STB small signal analysis is performed. After the circuit is linearized about a DC operating point, the STB analysis then calculates the loop gain, gain margin and phase margin for the circuit using a subset of Nyquist criteria.

it is also important to evaluate the stability of periodic steady state regimes of nonlinear circuits such as power amplifiers, injection locked oscillators and dividers. The STB analysis fails to predict the behavior of periodic steady state regimes of nonlinear circuits due to the nonlinear effects these circuits produce.

Periodic stability analysis (PSTB) performs stability analysis for circuits with a periodically time-varying operating point, which must first be obtained using a PSS analysis. The small signal PSTB analysis calculates the loop gain, gain margin and phase margin for circuits with a periodically time-varying operating point.

A Linear Periodic Time-Varying (LPTV) Feedback Circuit

Figure 1-12 shows the topology of a general linear time-invariant (LTI) feedback circuit around a specific feedback loop.

Figure 1-12 Topology for a Linear Feedback Circuit
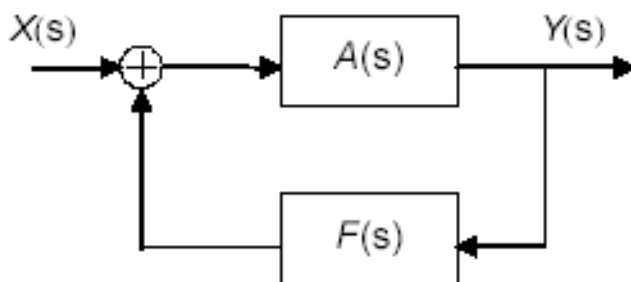
Equation 1-25 defines the closed-loop transfer function

(1-25) $\quad H(s) = \dfrac{Y(s)}{X(s)} = \dfrac{A(s)}{1 + A(s)F(s)}$

where X(s) and Y(s) represent input and output respectively.

Equation 1-26 defines loop gain.

(1-26) $\quad \mu(s) = -G(s)H(s)$

For the linear periodic time-variant (LPTV) feedback circuit shown in Figure 1-13

■ The fundamental frequency is ω_o

■ The input is $\mathbf{X}(s) = [..., X_m, ...]$

  Where $X_m = X(s + m \ast \omega_o)$ and $m = 0, \pm 1, \pm 2, ...$, m and n represent different sidebands.

■ The output is $\mathbf{Y}(s) = [..., Y_n, ...]$

  Where $Y_n = Y(s + n \ast \omega_o)$ and $n = 0, \pm 1, \pm 2, ...$, m and n represent different sidebands.

Figure 1-13 Topology for a Linear Periodic Time-Variant Feedback Circuit
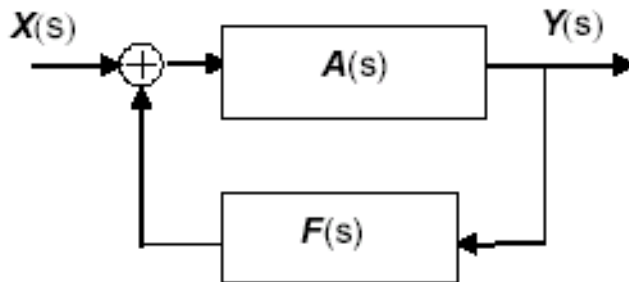
Equation 1-27 expresses the closed-loop harmonic transfer matrix (**HTM**).

(1-27) $\quad \mathbf{H}(s) = (\mathbf{I} + \mathbf{A}(s)\mathbf{F}(s))^{-1}\mathbf{A}(s)$

where **A**(s) and **F**(s) are matrixes and **I** is the identity matrix.

Evaluating the Stability of a LPTV Circuit Using PSTB Analysis

If you treat the feedback loop determined by Equation 1-27 as a multi-input multi-output (MIMO) system, the poles of any transfer function of the system are the same [1]. If you express a kind of transfer function of the zero sideband in the feedback loop as in Equation 1-28, you can obtain an exact open-loop transfer function $-\mu(s)$ by applying the loop-based algorithm [2] to the zero sideband.

(1-28) $\quad H_{00}(s) = \dfrac{A_0(s)}{1 + A_0(s)F_0(s)} = \dfrac{A_0(s)}{1 - \mu(s)}$

To evaluate stability, apply the Nyquist criteria to $\mu(s)$ (loop gain). This requires three steps you can perform using PSTB analysis.

1. Calculate $\mu(0 + j\omega)$ by sweeping ω

2. Plot $\mu(0 + j\omega)$ in a Bode plot or a Nyquist plot

3. Judge stability in one of the following two ways

   ❑ By the magnitude and phase of the loop gain

   or

   ❑ By the number of clockwise encirclements of the (1, 0) point on the loop gain curve

This stability check is restricted to local stability in the sense that the results are only valid under small perturbations of the steady state (that is, by small signal analysis). A large

perturbation might induce the circuit to permanently jump to a different steady state. To account for the large perturbation requires a global stability analysis in the parameter space.

(1-29) $H_b(\Omega, A) = 0$

(1-30) $\mu(0 + j\omega, \Omega, A) = 0$

Equation 1-29 is a PSS analysis equation. The parameter *A* can dramatically affect the circuit's steady state solution, such as source power or amplitude. The bifurcation curves are the solutions in the (Ω, A) space that meet Equation 1-30. This means that to determine global stability, you can use a two-dimensional (Ω, A) parameter sweep combining the PSS and PSTB analyses. This process might be time-consuming.

Example 1: Comparing the STB and PSTB Analyses

Figure 1-14, the `oscHartley` schematic, is an inherently nonlinear VCO. You can use the STB and PSTB analyses on the Hartley VCO to compare the nonlinearity effect on circuit stability.

Figure 1-14 Hartley Oscillator (oscHartley)
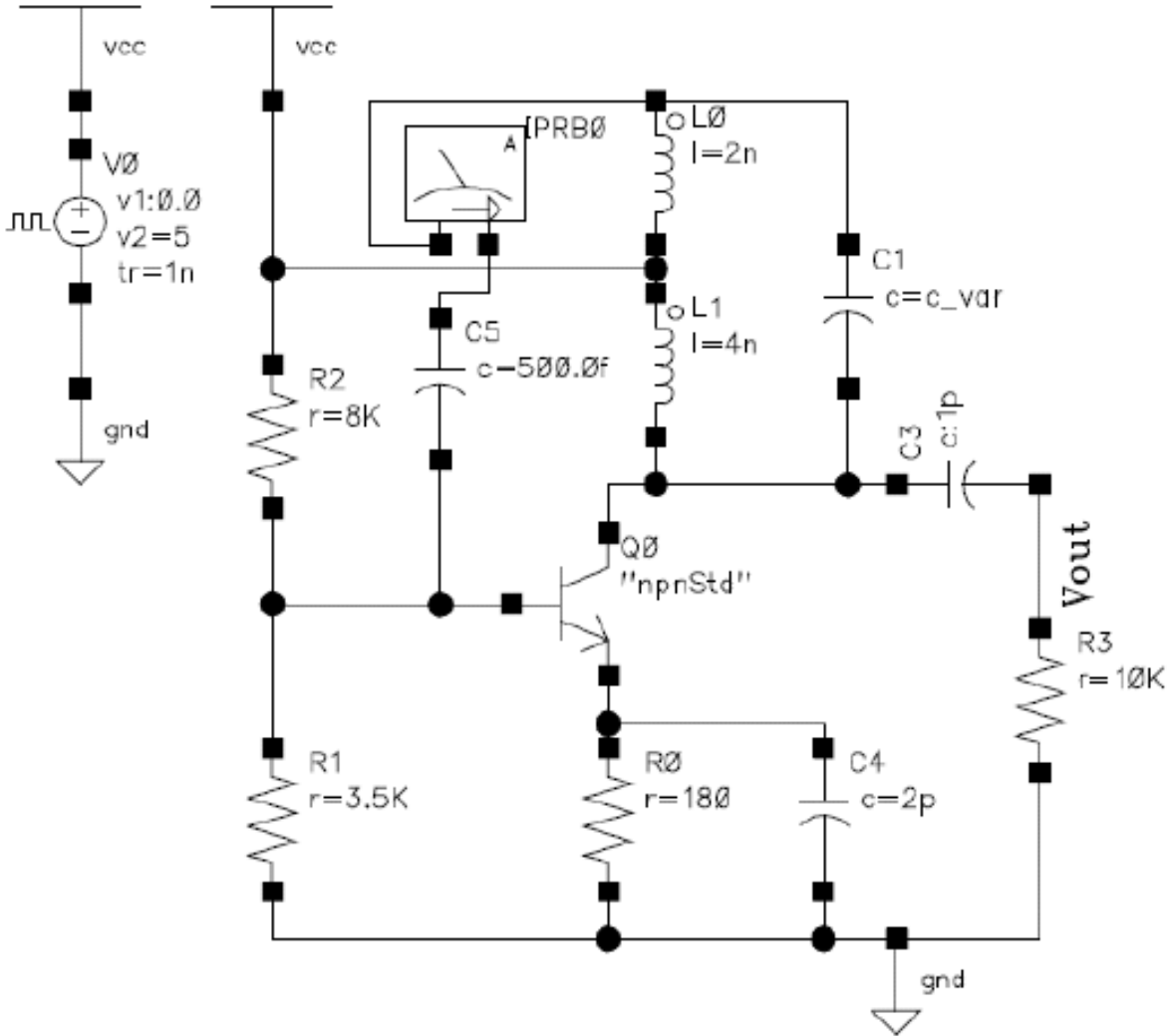
This VCO has the basic Hartley oscillator topology and is tunable between 720 MHz and 1.15 GHz. The oscillation frequency (f_0) is determined by the resonant circuit made up of inductors *L0*, *L1* and capacitor *C1*. In this particular VCO, the values of *L1* and *L2* are fixed whereas *C1* is a varactor diode. As a result, `Cvar`, the varactor diode s-junction capacitance, is a function of the applied voltage, as expressed in Equation 1-31.

(1-31) $$Cvar = \left|\frac{C_{j0}}{\left(1 + \frac{V}{\phi}\right)}\right|\gamma$$

In Equation 1-31

■ *V* is the applied junction voltage (*V*)

■ C_{j0} is the junction capacitance (*F*) for *V = 0V*

■ Φ is the barrier potential (*V*)

■ γ is the junction gradient coefficient

These variables have the following values

C_{j0}	8 pF
Φ	0.75 V
γ	0.4

Because C_{var} is inversely proportional to *V* and f_o is inversely proportional to C_{var}, the oscillation frequency is proportional to *V*. In other words, as you increase *V*, C_{var} decreases and f_o increases.

Set up for the STB and PSTB Analyses

Open the `oscHartley` schematic from your writable copy of the *rfExamples* library.

Set the Initial Conditions for the STB Analysis

1. Set up V_0 as a dc voltage source (*Source type = dc*, *dc* = 5).

2. Set the voltage on the C_{var} to 6 V (V_cntl = 6). The oscillator will oscillate at about 1.1 GHz.

3. Set `reltol` to 1e^{-4} (reltol = 1e^{-4})

4. Insert a current probe in the feedback loop (IPRB0).

Set Up the STB Analysis

1. *Sweep range* (*Start* = 1000M, Stop = 1200M)

2. *Sweep type=linear*

3. *Number of steps=400*

4. *probe* = IPRB0

Run the simulation

In WaveScan, plot the dB20 (loop gain) and Phase (loop gain) of the STB results. (See Figure 1-15.)

Set Up the PSS and PSTB Analyses

Set the Initial Conditions

1. Change V_0 to a *pulse* source (*val1*=0, *val2*=5, *rise*=1n) to start the oscillator.

2. Other parameters are the same as for the STB analysis.

Set up a PSS Analysis

1. *Beat Frequency* =1.1G

2. *Number of Harmonics* =30

3. *errpreset = moderate*

4. *tstab* =100n

5. Enable the oscillator button

6. Set *Oscillator node* = /Vout

7. Set *Reference node* = /gnd!

8. Set *Engine=shooting*

Set up a PSTB Analysis

1. *Sweep range* (*Start*=1000M, *Stop*=1.2G)

2. *Sweep type=linear*

3. *Number of steps*=400

4. *probe* = IPRB0

Run the Simulations

In WaveScan, plot the dB20 (loop gain) and Phase (loop gain). Append a Bode Plot of the PSTB results to the same window as the STB results. The plot in Figure 1-15 is displayed.

For an ideal oscillator, the loop gain at oscillating frequency, f_0, should be 1, so the dB20 (loopgain) and the Phase (loopgain) should both be zero at f_0 and the phase should change abruptly at f_0. In practice, both dB20 (loopgain) and the Phase (loopgain) should be close to zero.

For this autonomous circuit, the PSS analysis gives f_0 = 1.1035 G. From Figure 1-15, the PSTB analysis gives dB20 (loopgain) = 0 and Phase (loopgain) = 0. From Figure 1-15, the STB analysis gives dB20 (loopgain) and Phase (loopgain) far from zero. The results show that a PSTB analysis gives more accurate stability information for nonlinear circuits than does a STB analysis.

Figure 1-15 VCO Loop Gain
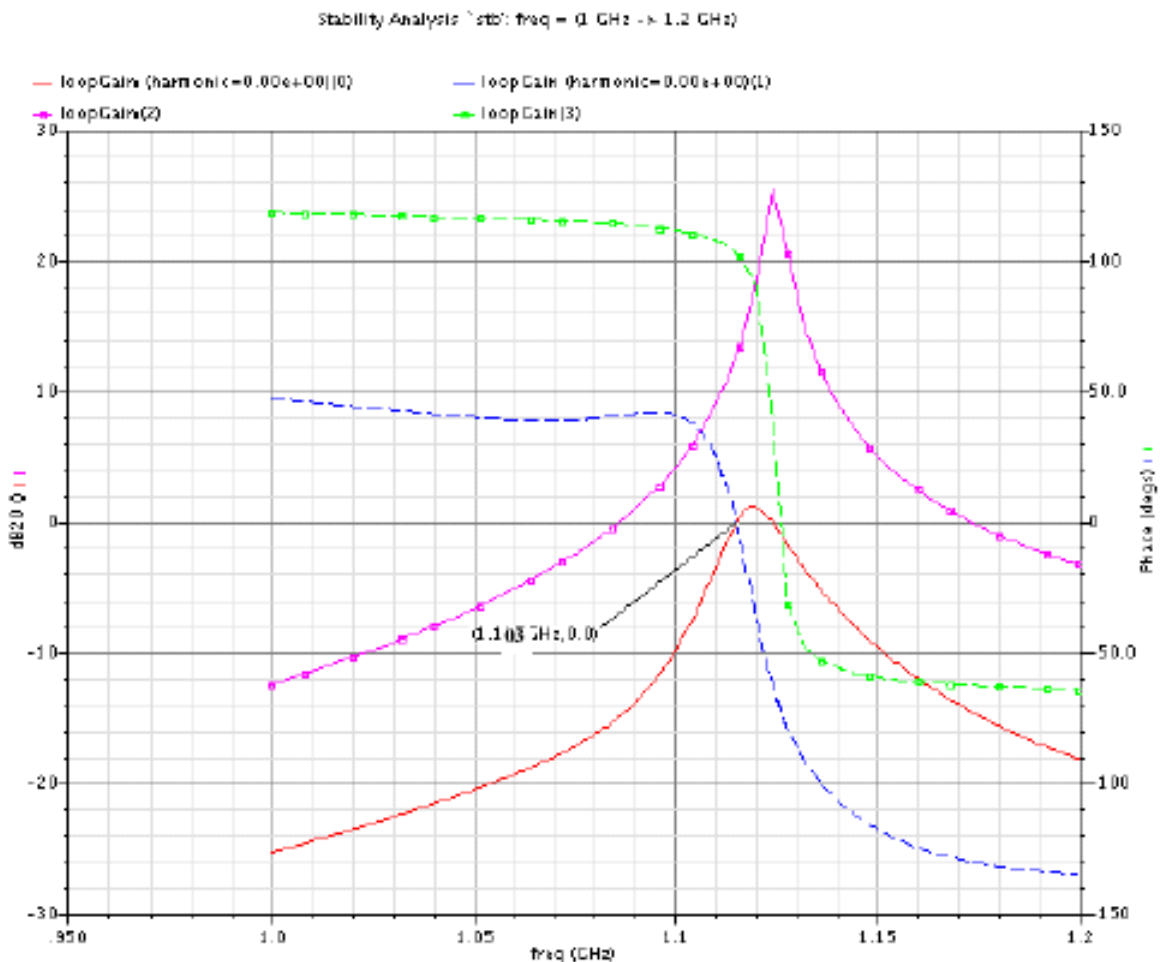
In Figure 1-15 lines with boxes are from the STB analysis, lines without boxes are from the PSTB analysis, solid lines represent magnitude and dotted lines represent phase.

Example 2: Local Stability of an Injection-Locked Oscillator

Figure 1-16 shows an injection-locked oscillator based on the VCO in Figure 1-14 on page 62. The injection-locked oscillator also includes the following for injection: current source I0, resistor R4 and capacitor C7. The frequency and amplitude of I0 are the parameters `inj_freq` and `inamp`.

Figure 1-16 Schematic of the Injection-Locked Oscillator
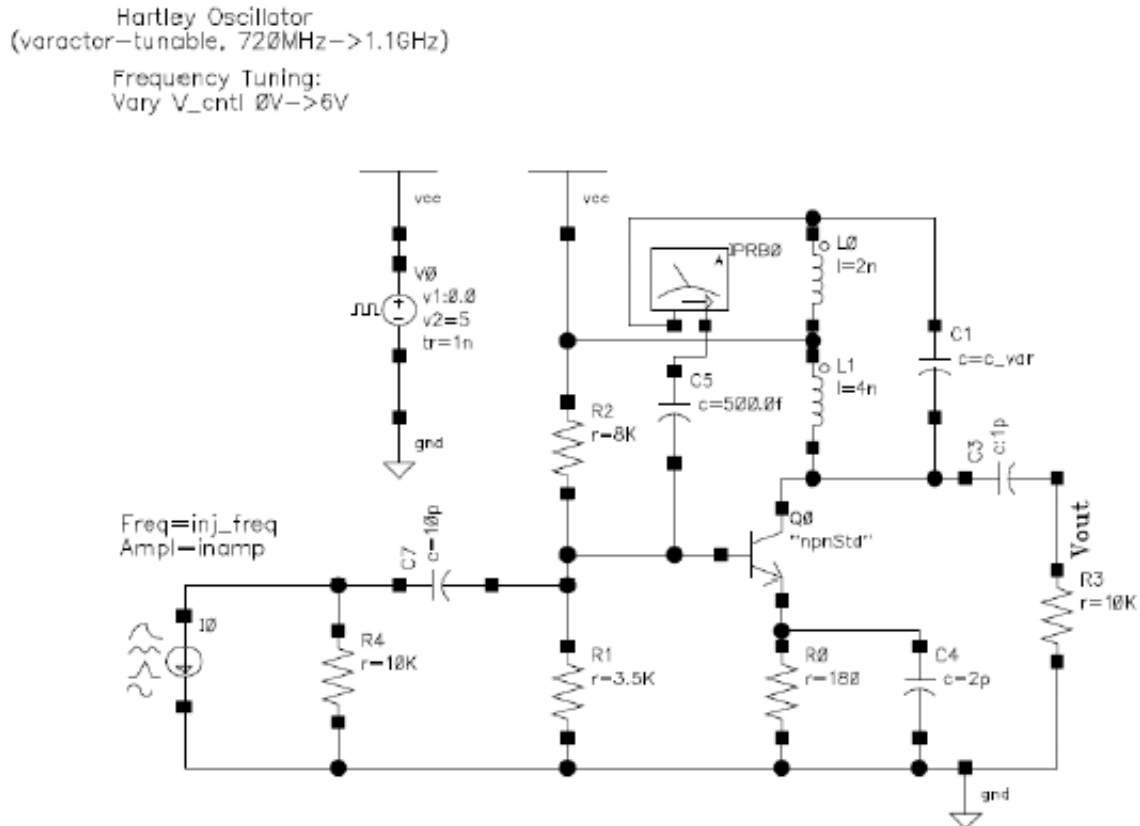
Set the Initial Conditions for the PSS and PSTB Analysis

1. Set `inj_freq` = 1000M.

2. Set `inamp` = 13mA

3. Other parameters are the same as the PSTB analysis in section 3 a.

Set Up the PSS Analysis

1. *Beat Frequency* =1G

2. *Number of Harmonics* =30

3. *errpreset = moderate*

4. *tstab* =100n

5. *Engine=shooting*

Set Up a PSTB Analysis

1. *Sweep range* (*Start*=1000M, *Stop*=1.2G)

2. *Sweep type=linear*

3. *Number of steps*=400

4. *probe* = IPRB0

Run the Simulation

In WaveScan, plot the real vs imag (loopgain) of the PSTB analysis.

Run the same simulation again but change `inamp` to 13.85mA.

Run the same simulation again but change `inamp` to 15mA.

Plot the results of the three simulations in the same window. The plot in Figure 1-17 is displayed.

You can see from Figure 1-17 when `inamp` = 13.85mA, the loopgain = 1. (From the Bode plot, the corresponding small signal frequency is 1.114GHz). It implies that there is a potential for a free-running oscillation at 1.114G.

Figure 1-17 Nyquist Plot of the Loopgain of the Injection-Locked Oscillator
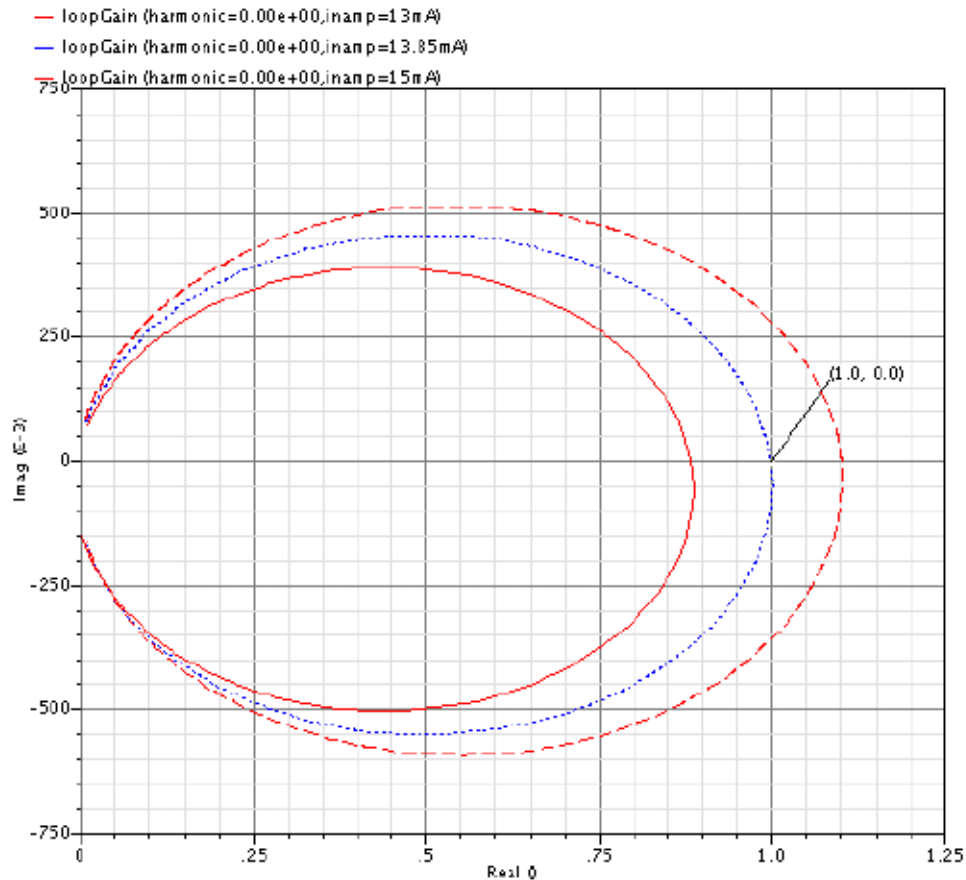
Figure 5 Nyquist plot of the loopgain of the injection-locked oscillator
(Dash line: *inamp=13mA*, Dot line: *inamp=13.85mA*, Solid line: inamp=15mA)

In Figure 1-17, the dashed line represents `inamp` =13mA. The dotted line represents
`inamp` = 13.85mA, The solid line represents `inamp` =15mA)

When `inamp` is greater than 13.85mA, the loop gain curve does not encircle the (1, 0) point,
which implies that the circuit is stable. When `inamp` is less than 13.85mA, the loop gain curve
encircles the (1, 0) point clockwise, which implies that the circuit is unstable. In fact, when you
run a PSS analysis with `inamp=13mA` and increase `tstab`, such as `tstab=40` µs, the PSS
analysis does not converge. If you run a long transient analysis, such as when *Stop*=350 µs,
the resulting waveform is not periodic as you can see in Figure 1-18.

Figure 1-18 The Output Waveform of the Injection-Locked Oscillator With Injection Current inamp = 13 mA
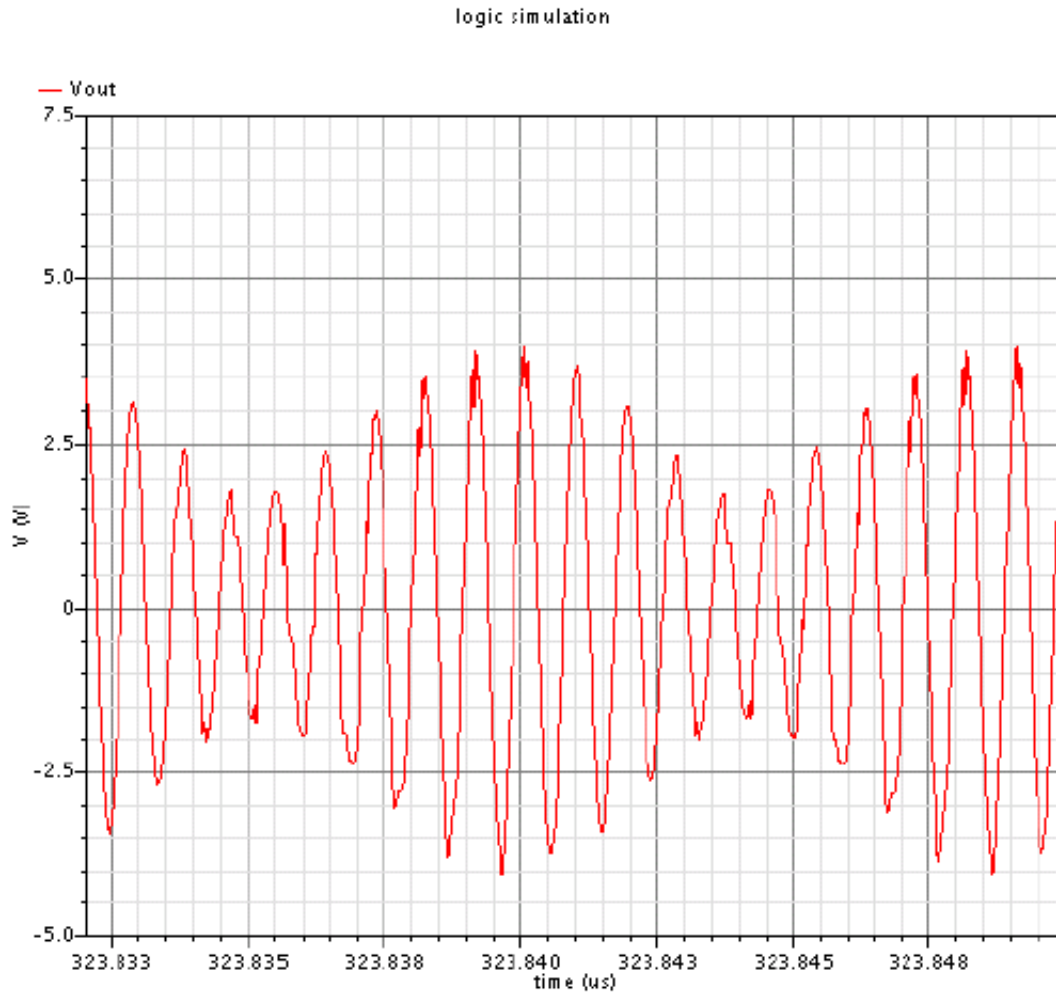
Example 3: Global Stability of Injection-Locked Oscillators

Equation 1-32 represents the locked oscillating condition.

(1-32) $$loopgain(injfreq, inamp1) = 1$$

Equation 1-33 represents the free-running oscillator condition.

(1-33) $loopgain(freq, inamp2) = 1$

In Equations 1-32 and 1-33

■ *inj_freq* represents the injection frequency

■ *freq* represent the arbitrary small signal frequency

■ *inamp1* and *inamp2* are the amplitudes of the injection current

You can use the two curves determined by Equations 1-32 and 1-33 to evaluate the global stability of the circuit. Produce the curves using a PTSB analysis.

Set the Initial Conditions for the PSS and PSTB Analyses

1. Set `inj_freq` = 1075M.

2. Set `inamp` = 16mA

3. Other parameters are the same as used for the previous PSTB analysis.

Set Up the PSS Analysis

1. `Beat Frequency` =1075M

2. `Number of Harmonics` =30

3. `errpreset = moderate`

4. `tstab` =100n

5. `Engine=shooting`

Set Up the PSTB Analysis

1. `Sweep range` (`Start=1050M`, `Stop=1.2G`)

2. `Sweep type=linear`

3. `Number of steps=400`

4. `probe` = IPRB0

Set Up a Parametric Analysis to Evaluate Global Stability

1. `Variable name = inamp`

2. `Range Type = From/To`

3. `From=16mA`

4. `To=20mA`

5. `Step Control = Linear steps`

6. `Step Size =0.5mA`

Run the simulation

In WaveScan, plot the loop gain as a Bode Plot similar to Figure <u>1-19</u>.

You can see that when `inamp` = 16.5 mA and `loopgain` = 1 at frequency 1.114 GHz, the circuit is a potential free-running oscillation. When `inamp` is less than 16.5 mA, the circuit is unstable (you can find this in a corresponding Nyquist Plot). When `inamp` increases, free-running oscillation disappears and the circuit tends to be locally stable. However, when `inamp` increases to 19.5 mA, another oscillation occurs at 1075 M, which implies that the circuit is locked to oscillate at the injection frequency. When `inamp` is greater than 19.5 mA, the circuit again tends to stabilize. From Figure <u>1-32</u>, you can clearly observe the injecting-locking process as you increase the injection current.

Not all injection frequencies can be locked. Only those injection frequencies near a free-running frequency can be locked. A complete global stability evaluation requires a two dimensional sweep of (`inj_freq`, `inamp`). However, it is not easy to obtain two curves accurately around the locked area for the strong competition between locked mode and free running mode. This unstable property will lead to PSS analyses that cannot converge and PSTB analyses that cannot run.

Figure 1-19 Bode Plot of Loop Gain with inj_freq = 1075M
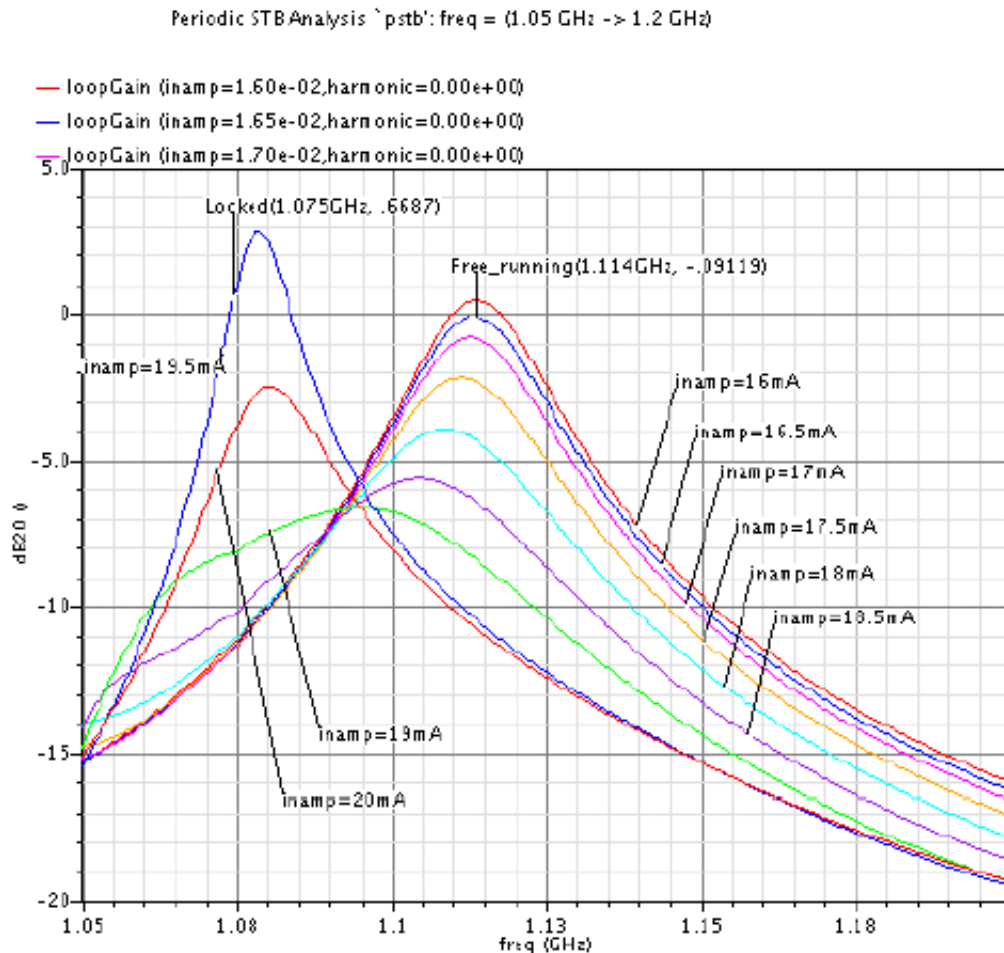
References

[1]   Ogata K, Modern control engineering, Printice-Hall, 1998.

[2]   Michael Tian, V. Visvanathan, Jeffrey Hantgan, and Kenneth Kundert, "striving for small-signal stability", IEEE Circuit & Devices, Jan 2001(31).

The Spectre/RF MATLAB Toolbox

The MATLAB® Toolbox provides an interface between both the Spectre and UltraSim® circuit simulation technologies and the MATLAB data manipulation environment. This section describes how to use the toolbox to read Spectre simulation results into MATLAB and to use MATLAB to perform some standard RF measurements.

The toolbox provides all Spectre and SpectreRF users with an alternative method for post-processing simulation data and making standard RF measurements in MATLAB. The toolbox allows experienced SpectreRF users to use Simulink® and MATLAB to make customized measurements on information extracted from Spectre simulation results. Experienced users can also perform high-level design tasks in Simulink and MATLAB.

MATLAB, a powerful mathematical and graphic tool, provides rich data processing and display functionality. Many users want to customize their measurements and displays. The toolbox supports these use models.

The MATLAB Toolbox includes a number of functions you can use to

■ Read Spectre simulation results in PSF or SST2 format into MATLAB

■ Perform basic data filtering and plotting tasks

■ Support typical RF measurements such as IP3 and compression point

Install the Toolbox Package

The MATLAB Toolbox is a standalone package shipped with MMSIM version 6.1 and higher. The home directory for the toolbox is

`<instdir>/tools/spectre/matlab`

where `<instdir>` is the installation directory for the MMSIM simulation software. In the MATLAB Toolbox there are 3 `MEX`-files and 12 `M`-files.

Tip

> Use the `` `cds_root spectre` `` command to determine the installation directory of your MMSIM simulation software.

Configure the Toolbox Package

The Spectre/RF Toolbox in MATLAB depends on the Spectre simulation run environment. It uses shared libraries located in the Spectre installation path.

1. Make sure you are running Spectre 6.0 or a higher version.

2. Verify the dynamic library path by checking the `LD_LIBRARY_PATH` environment variable. Make sure that both `<instdir>/tools/dfII/lib` and `<instdir>/tools/lib` are in the path. For C shell users, use the following command

```
setenv LD_LIBRARY_PATH `cds_root spectre`/tools/dfII/lib:`cds_root spectre`/tools/lib:${LD_LIBRARY_PATH}
```

3. Solaris users also need to add the toolbox installation path to `LD_LIBRARY_PATH`. For C shell users, use the following command

```
setenv LD_LIBRARY_PATH `cds_root spectre`/tools/spectre/matlab:${LD_LIBRARY_PATH}
```

4. The MATLAB script sets the `MATLABPATH` environment variable to include the MATLAB toolbox directories and the user created directories. You need to add the toolbox installation path to the `MATLABPATH` environment variable. For C shell users, use the following command

```
setenv MATLABPATH `cds_root spectre`/tools/spectre/matlab:${MATLABPATH}
```

The Basic Toolbox Functions

Each toolbox function command has an associated help page which you can display by typing `help <command_name>` in MATLAB. This section introduces the basic toolbox functions, describes how to use each function and describes how to write measurement functions.

cds_srr

Lists the datasets in the result directory, lists signals in a dataset or reads a signal into MATLAB.

1. To list the datasets in the result directory, use the command

```
datalist = cds_srr('result_directory')
```

   All `dataset` names in the result directory are returned with `datalist` as a string vector.

2. To list the signal names in a dataset, use the command

```
signals = cds_srr('result_directory', 'dataset_name')
```

   All `signal` names and `property` names in the `dataset` are returned with `signals` as a string vector.

3. To read a signal into MATLAB, give both a `dataset` name and a `signal` name and use the command

```
signal = cds_srr('result_directory', 'dataset_name', 'signal_name')
```

The value of the `signal` is returned with *signal*. The `signal` can be a single value or a structure containing a matrix.

Normally, when you use **cds_srr** to read a signal into MATLAB, **cds_srr** returns a structure containing fields. The first field is `info`, a string vector of `name`, `unit` pairs. In each pair, the first value is the `name` of the field, the second value is the `unit` of the field. The first `name`, `unit` pair in the `info` field describes the final value, the other pairs describe the sweep information if the signal was swept. An inner sweep is always listed before an outer sweep in `name`, `unit` pairs in the `info` field. The innermost sweep can be a matrix with variable sizes in each column, the other sweeps have to be a vector of fixed size.

In MATLAB,

■   If a semi-colon (`;`) is at the end of a command, the system will not display the return results.

■   Two single quotes ('') mean empty data.

Tip

The **cds_srr** command normally returns informational messages. To turn off these messages, use a zero (0) as the fourth parameter. For example, the following command runs silently.

```
datalist = cds_srr('results_directory', '', '', 0);
```

The **cds_srr** command supports both PSF and SST2 formats. **cds_srr** is an external function with **cds_innersrr** running in the background. The **cds_srr** and **cds_innersrr** commands have the same interface, but **cds_srr** adds some post processing to make the resulting matrix easier to read.

cds_evalsig

Filters the data from swept analyses. The command is

```
v = cds_evalsig(signal, expression)
```

The first parameter *signal* is the result of the **cds_srr** command. The second parameter *expression* is a string expression. You can use both relational operators (`<`, `<=`, `==`, `>=`, `>>`) and logic operators (`&`, `|`).

If the *signal* has the swept fields `prf` and `time`, you can write an expression like the following,

```
prf < -20 | prf == -10 & time >= 2e-8
```

The logic operator (`|`) is only effective between expressions with the same field name. The following expression is meaningful.

```
prf==-10 | prf >= -20
```

When the logic operator (`|`) is used between two different field names, it is equivalent to the logic operator (`&`). The following two expressions are equivalent.

```
prf==-10 | time <= 2e-8
```

and

```
prf==-10 & time <= 2e-8
```

cds_plotsig

Plots swept signals. The command is

```
cds_plotsig(signal, expression, sweep_name, type_id)
```

The first two parameters *signal* and *expression* have the same meaning as described for the **cds_evalsig** command. The third parameter *sweep_name* is the name of a sweep field. You can define a special x-axis with *sweep_name* when the result has multiple sweeps. The default value for *sweep_name* is the name of the innermost sweep. The fourth parameter *type_id* is a string type id that can be any of the following: `mag`, `phase`, `real`, `imag`, `both`, `db10`, `db20` or `dbm`.

cds_harmonic

This command is specifically designed for RF results. It helps you to select harmonics and sidebands. In the toolbox data structure created by the **cds_srr** command, all sweep field names are listed in the `info` field. The harmonic information is not an independent sweep, it always combines frequencies and tones and it is not listed in the `info` field. The `harmonic` and `harmUnit` fields contain the harmonic information. The **cds_harmonic** command can select the interesting harmonics with these pieces of information. The command is

```
hsig = cds_harmonic(signal, harms)
```

For the PSS analysis, `harms` is a single integer. For the QPSS analysis `harms` is a vector. For example, use the following command for a QPSS analysis.

```
ord3 = cds_harmonic(rfout, [2 -1])
```

cds_interpsig

The intervals between time points in Spectre results are not equal. We provide an interpolation command to distribute the time points evenly. The command is

```
v = cds_interpsig(signal, sweep_name, num, method)
```

Where the first parameter `signal` results from other commands such as **cds_srr**. The second parameter `sweep_name` is the name of the inner sweep—normally it is `time`. The third parameter `num` is the vector length after interpolation. The fourth parameter `method` specifies a method for interpolation—it is a string value which can be `linear`, `cubic`, `nearest` or `spline`. The default interpolation method is `linear`.

Only the first parameter is required.

The Measurement Commands

cds_fft

The FFT (Fast Discrete Fourier Transform) is a typical RF measurement. MATLAB provides an internal `fft` function. The command **cds_fft** calls the internal MATLAB `fft` function. It prepares the input signal, and calls `fft` for each outer sweep. The command is

```
fsig = cds_fft(tsig)
```

The input parameter `tsign` is a time-domain signal. The output parameter `fsig` contains frequency domain data.

Tip

> The **cds_fft** command is a MATLAB script. It is located in the file `cds_fft.m` in the installation directory. If you want to write your own measurement, this script offers a good example.

In `cds_fft.m`, the script checks the input value at the beginning. The `tsig` should exist and not be empty. As a time sweep result of **cds_srr**, it must have the fields `info` and `time`. Because the intervals between time points in `tsig` vary, the script calls **cds_interpsig** to distribute the time points evenly. From the vector of field times, the script gets the frequency vector. `tsig` can also be a multi-level sweep, so it does an fft for each outer sweep. After the fft, it copies additional sweep information from the `tsig` to `fsig`, and fills the field `time` with the field `freq`.

cds_compression

Returns the n-dB input referred compression point for the wave supplied. The command is

```
comp = cds_compression(vport, iport, harm, rport, gcomp, curve)
```

Where the first parameter *vport* is the voltage of the port. Normally it is the return value of **cds_srr**. The second parameter *iport* is the current of the port. It can be calculated from the voltage and impedance. The third parameter *harm* is the 1st order harmonic, the default value is 1. The parameter *rport* is the impedance of the port, the default value is 50. The parameter *gcomp* is gain compression in dB, the default value is 1. The parameter *curve* is the control flag of curve display. The default value of curve is on. It can be off.

cds_ipn

Returns the Intercept Point for the wave supplied. The command is:

```
[ipn_in, ipn_out] = cds_ipn(vport, harmspur, harmref, epoint, rport, ordspur,
iport, epref, ordref, curve)
```

The parameters of **cds_ipn** are similar to the parameters of **cds_compression**. The first parameter *vport* is the voltage of the port. The second parameter *harmspur* is the order harmonic value. The third parameter *harmref* is the reference order (1st order) harmonic value. The parameter *epoint* is the input power extrapolation point. The parameter *rport* is impedance of the port. The default value is 50.0. The parameter *ordspur* is order number. The default value is 3. The parameter *iport* is the port current. The default value is *vport*/*rport*. The parameter *epref* is the input power reference point. It will use *epoint* when this parameter is not specified. The parameter *ordref* is the reference order number. The default value is 1. The parameter *curve* is the control flag of curve display. The default value is on. It can be on or off.

The return parameter *ipn_in* is the input reference IPn value and *ipn_out* is the output reference IPn value.

Example

This section shows how to use the toolbox to make RF measurements in MATLAB. The example circuit is an LNA design from the RF workshop which is available at

```
<instdir>/tools/spectre/examples/SpectreRFworkshop
```

Please refer to *Lab3: Gain Compression and Total harmonic Distortion (Swept PSS)* in the workshop *LNA Design Using SpectreRF*.

After finishing the LNA Lab3, you get the results in

```
simulation/Diff_LNA_test/spectre/schematic/psf
```

To start MATLAB from the system prompt, type the following command

```
matlab
```

In MATLAB, get time domain data with the command

```
>> rdir = 'simulation/Diff_LNA_test/spectre/schematic/psf';
>> tdout = cds_srr(rdir, 'sweeppss_pss_td-sweep', 'RFout');
```

Figure 1-20 Gain Compression (1dB Compression)
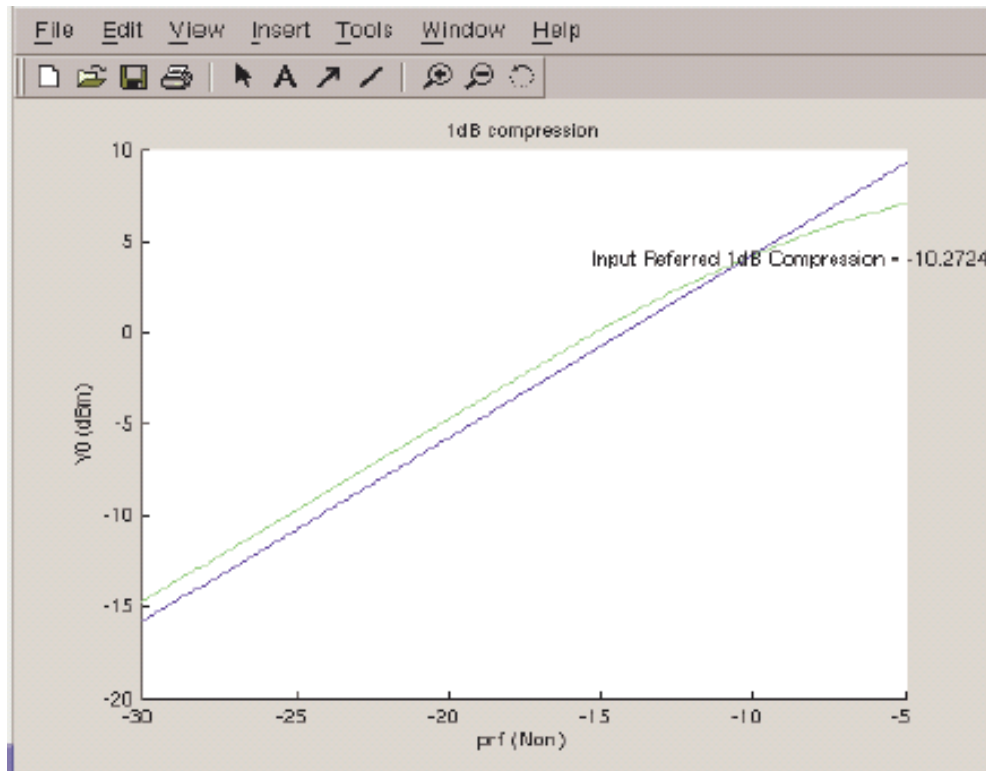
To get 1 dB compression point, use the following command

```
>> cds_compression(fdout)
```

A window with the result will pop up, as Figure 1-20 shows.

tdout is a power sweep result. For different prf, the time point numbers are different, and for each prf, the intervals between time points vary. The field *time* is a 161x11 matrix as the field V.

```
Execute command:
    >> td161=cds_interpsig(tdout);
    >> td100=cds_interpsig(tdout, 'time', 100);
    >> hold on
    >> cds_plotsig(tdout)
    >> cds_plotsig(td161)
    >> cds_plotsig(td100)
```

The waveforms shape of td161 and td100 are the same as $tdout$. But in td161 and td100, the time points are the same for each different prf and the intervals of time points are the same for each prf. So, the field time is stored and handled as a vector.

To get the frequency domain data, execute command:

```
>> fdout = cds_srr(rdir, 'sweeppss_pss_fd-sweep', 'RFout');
```

$fdout$ contains 11 harmonics, the field `harmonic` lists the harmonic number. To get the 1st order harmonic signal, use the following command

```
>> fd1 = cds_harmonic(fdout, 1)
```

Figure 1-21 Total Harmonic Distortion
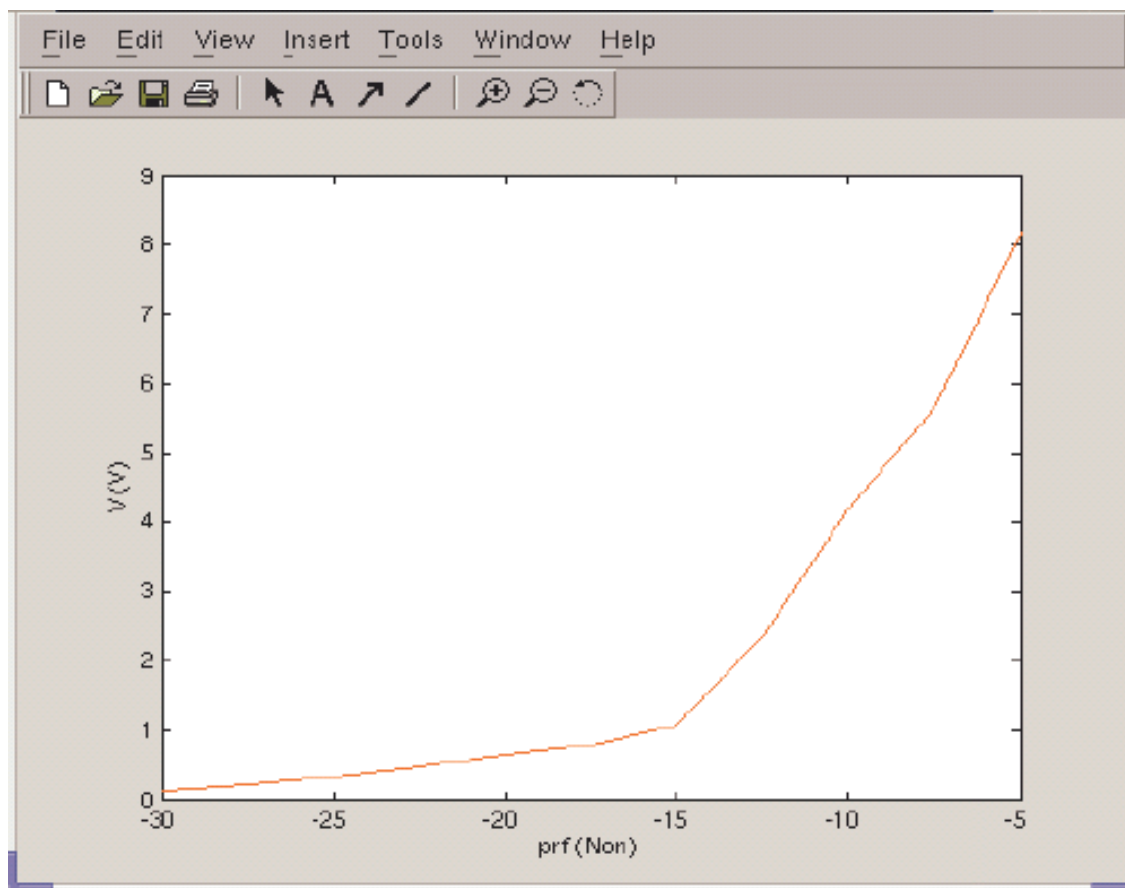
To get total harmonic distortion in percent, use the following command

```
>> thd = cds_thd(fdout);
>> cds_plotsig(thd);
```

The result is shown in Figure 1-21.

To compare the frequency data with time domain data.

```
>> fd1 = cds_evalsig(fdout, 'prf = -30');
>> td2fd = cds_fft(tdout);
>> fd2 = cds_evalsig(td2fd, 'prf = -30 & freq<=2.5e10')
```

The result is shown in Figure 1-22.

Figure 1-22 Compare FD data before FFT and after
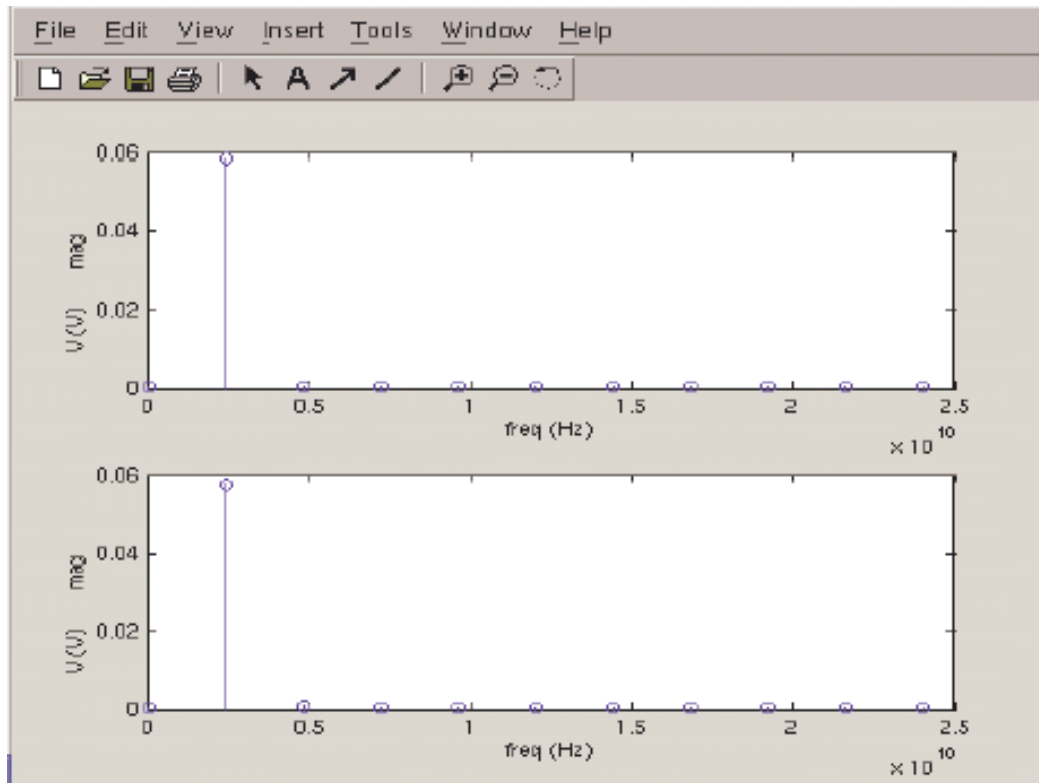
Compatibility with Aptivia MATLAB Functions

This topic is of interest to those users who are using acv measures in MATLAB. The document for these measures is in `<cdsinst>/doc/matlabmeasug/` Chapters 3 and 4.

The Spectre/RF MATLAB Toolbox is better able to handle both swept analyses and RF-related analyses such as Monte Carlo and PSS/PAC than acv measures. While acv measures provide more measurements for the transient and ac analyses, the MATLAB Toolbox can reuse acv measure functions with the command **cds2vsde** and the global variable `CDS2ACV`.

Set the global variable CDS2ACV to 1 in MATLAB with the following command.

```
>> global CDS2ACV; CDS2ACV = 1
```

When CDS2ACV=1, **cds_srr** prints vsde compatibility information after loading in the dataset. **cds_srr** will translate the signal to acv data structure for acv measures automatically.

The **cds2vsde** command is designed for translating signals to acv data structure. The command is

```
cds2vsde(sig, raw_file, variable, option)
```

The parameters *raw_file* and *variable* are string values to identify the signal. The parameter *option* can be add or replace as other acv functions.

Note: The acv data structure cannot handle multi-level sweeps, and for each *raw_file* the sweep value should be the same. So **cds2vsde** will use different variables or different *raw_files* for multi-level sweep signals.

If you want to convert variable *td2fd* to acv format, use the following command

```
>> cds2vsde(td2fd, 'sweeppss_pss_fd-sweep', 'td2fd');
```

It will print out:

```
vsde compatibility information
raw_file: sweeppss_pss_fd-sweep
variable: td2fd(prf=-30)
variable: td2fd(prf=-27.5)
variable: td2fd(prf=-25)
variable: td2fd(prf=-22.5)
variable: td2fd(prf=-20)
variable: td2fd(prf=-17.5)
variable: td2fd(prf=-15)
variable: td2fd(prf=-12.5)
variable: td2fd(prf=-10)
variable: td2fd(prf=-7.5)
variable: td2fd(prf=-5)
```

Then use command **meas_plot** for plotting:

```
>> meas_plot('sweeppss_pss_fd-sweep', 'td2fd(prf=-20)', 'stem');
```

Reference

[1] *Simulation Results Reader User Guide*, Product Version 5.0

[2] *MATLAB External Interfaces Reference* available at

   `http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html`

[3] SpectreRF Workshop--LNA Design Using SpectreRF, MMSIM6.0USR2

MATLAB Support Matrix

Table 1-1 MATLAB Toolbox

MMSIM	MATLAB Release	Supported Platform		
		Linux 32	*Linux 64*	*Solaris 64*
	R2007a	Supported	Supported	Supported
	R2007b	Supported	Supported	Supported
	R2008a	Supported	Supported	Supported
MMSIM13.1	R2008b	Supported	Supported	Supported
	R2009a	Supported	Supported	Supported
	R2009b	Supported	Supported	Supported
	R2010a	Supported	Supported	Not Supported [1]
	R2010b	Supported	Supported	Not Supported [1]
	R2011a	Supported	Supported	Not Supported [1]
	R2011b	Supported	Supported	Not Supported [1]
	R2012a	Supported	Supported	Not Supported [1]
	R2012b	Not Supported[2]	Supported	Not Supported [1]
	R2013a	Not Supported[2]	Supported	Not Supported [1]
	R2013b	Not Supported[2]	Supported[3]	Not Supported [1]
	R2014a	Not Supported[2]	Supported[3]	Not Supported [1]

1.   From MATLAB versions 2010a onwards, Solaris Platform is no longer supported.
2.   From MATLAB versions 2012b onwards, Linux 32-bit platform is no longer supported

3. From MATLAB vesions 2012b onwards, Red Hat Enterprise Linux 5 is no longer supported

Table 1-2 MATLAB Cosimulation

MMSIM	MATLAB Release	Supported Platform		
		Linux 32	*Linux 64*	*Solaris 64*
	R2007a	Supported	Supported	Supported
	R2007b	Supported	Supported	Supported
	R2008a	Supported	Supported	Supported
MMSIM13.1	R2008b	Supported	Supported	Supported
	R2009a	Supported	Supported	Supported
	R2009b	Supported	Supported	Supported
	R2010a	Supported	Supported	Not Supported [1]
	R2010b	Supported	Supported	Not Supported [1]
	R2011a	Supported	Supported	Not Supported [1]
	R2011b	Supported	Supported	Not Supported [1]
	R2012a	Supported	Supported	Not Supported [1]
	R2012b	Not Supported[2]	Supported	Not Supported [1]
	R2013a	Not Supported[2]	Supported	Not Supported [1]
	R2013b	Not Supported[2]	Supported[3]	Not Supported [1]
	R2014a	Not Supported[2]	Supported[3]	Not Supported [1]

1. From MATLAB versions 2010a onwards, Solaris Platform is no longer supported.
2. From MATLAB versions 2012b onwards, Linux 32-bit platform is no longer supported
3. From MATLAB vesions 2012b onwards, Red Hat Enterprise Linux 5 is no longer supported

Note: Starting with the MMSIM 14.1release, the MATLAB toolbox supports only the R2012a and above versions.

Noise Separation in Pnoise and Qpnoise Analysis

This section describes how to analyze RF circuits using the noise separation features in the Pnoise and Qpnoise analyses. SpectreRF users in the Analog Design Environment (ADE) will find noise separation information to be useful.

Principles of Noise Separation in RF Circuits

For the Pnoise and Qpnoise analyses, input noise can be either *stationary* or *cyclostationary*. A simple instance of stationary noise is the white noise in a resistor with constant resistance. Cyclostationary noise is generally due to the fact that in most RF circuits the operating point of the nonlinear devices, primarily transistors, is periodic and time-varying.

To better illustrate the difference between stationary noise and cyclostationary noise, consider the white thermal noise generated by either the nonlinear drain-to-source or channel resistor in a MOS transistor. The formula for *white thermal noise* is given as

(1-34) $\quad \sqrt{4KTR(V)}$

In Equation 1-34, $R(V)$ is the bias-dependent small signal drain-to-source resistance. Assuming the transistor is driven by a periodic large signal excitation, for example the clock in a switched capacitor filter, $R(V)$ will be time-varying and you can no longer model its associated thermal noise as a simple stationary noise source. You must treat the noise source as a cyclostationary random process.

When input noise passes through an RF circuit, the *aliasing* or *noise folding* effect is introduced by the frequency translation intentionally performed by the linear periodic time-varying (LPTV) characteristics of the RF circuit. You can summarize the noise transfer process as follows.

■ First, for noise that is bias dependent, the time-varying operating point modulates the noise sources

■ Second, the transfer function from the noise source to output modulates the noise source contribution to the output

■ The final result is the sum of the noise contributions both up-converted and down-converted to the desired output frequency. The noise transfer process is shown in Figure 1-23, where a mixer is used to demonstrate.

Figure 1-23 How a Mixer Moves Noise Around
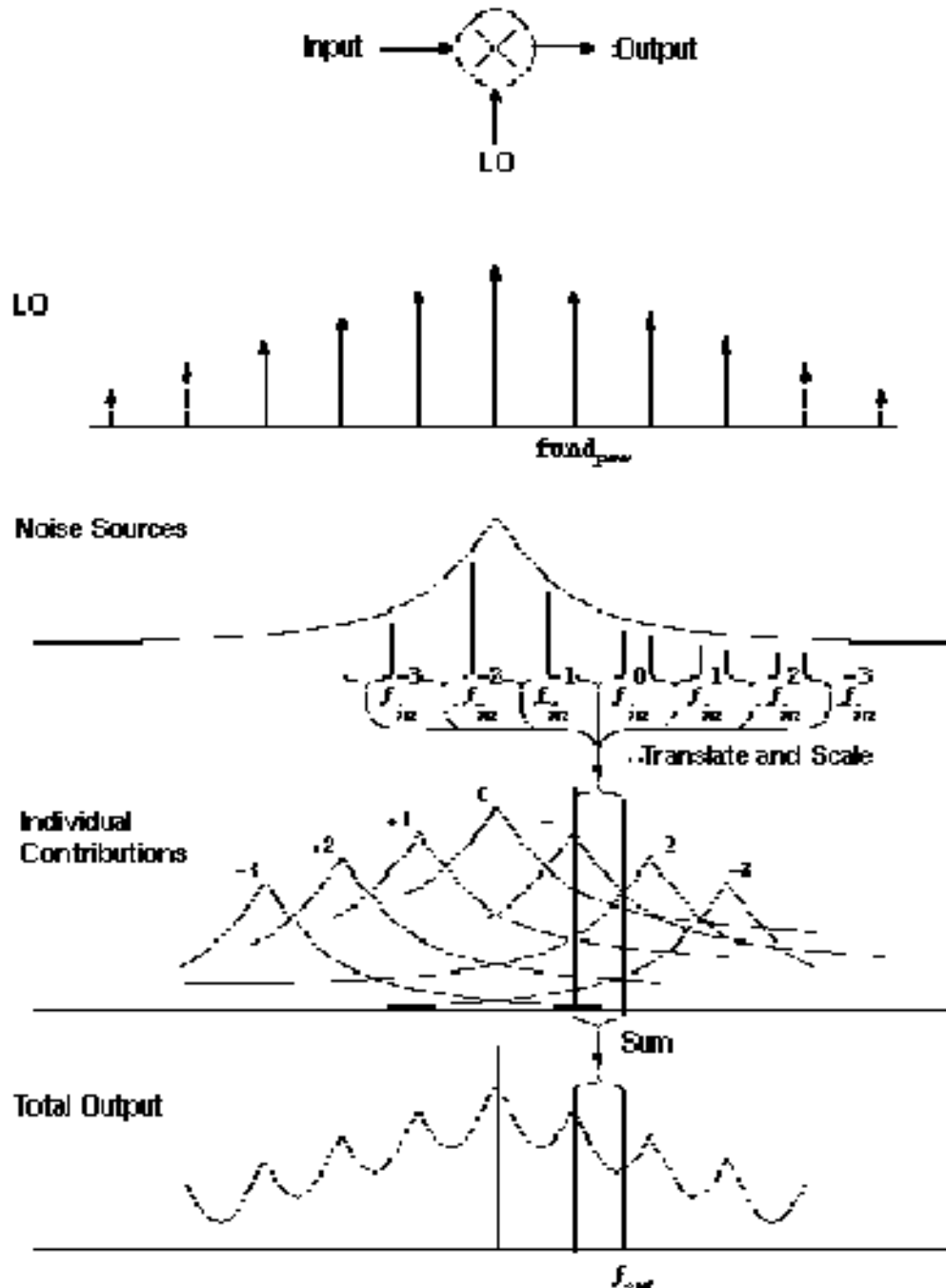
The Pnoise and Qpnoise analyses calculate the output noise transferred from the noise sources by transfer functions. Using the Direct Plot form, you can plot *output noise*, *input*

noise, *noise figure*, *transfer function*, and so on. By using the Noise Summary Print form and selecting sort and truncation methods, you can print *spot noise* or *integrated noise*.

The noise separation feature provides more information about how much of the noise is due to the noise sources and how much of the noise is due to the corresponding transfer function. The main purpose of giving the extra information is to provide useful information about how noise sources contribute noise to the output.

You can select which one of the two approaches to use to decrease the output noise

■   By decreasing the noise sources using different device dimensions

■   By decreasing the transfer functions with other circuit architectures

For some projects, you might want to separate the noise due to sources from the noise due to transfer functions in order to satisfy noise specifications. When you include the noise separation specifications in the design process you are using global optimization to help produce better designs.

When you take *K* sidebands and one noise source into account, Equation 1-35 illustrates the power spectrum density (PSD) of the output noise due to this single noise source.

(1-35) $$Output_{SingleSource} = \sum_{i \in K} \left[NoiseSource_i \cdot TransferFunction_i^2 \right]$$

Where *NoiseSource$_i$* represents the PSD of *Source$_i$* sampled at the frequency shift of *i $*$ fundamental*.

When the noise is cyclostationary, Equation 1-36 represents the output PSD due to this source.

$$Output_{SingleSource} = [NoiseSource_{i1}, NoiseSource_{i2}, \ldots NoiseSource_{iK}] \cdot$$

(1-36) $$[FreqConversionMatrixOfBias] \cdot$$

$$\left[NoiseSource_{j1}^2, NoiseSource_{j2}^2, \ldots NoiseSource_{jK}^2 \right]^T$$

By introducing the *noise gain* as a parameter, the formulas in Equations 1-35 and 1-36 can be unified as they are in Equation 1-37.

(1-37) $$Output_{SingleSource} = \sum_{i \in K} \left[NoiseSource_i \cdot NoiseGain_i^2 \right]$$

Therefore, you know that

■ When the noise source is stationary, the noise gain equals the transfer function

■ When the noise source is cyclostationary, the noise gain is a normalized combination of frequency conversion due to bias modulation and transfer functions.

The Noise Separation GUI

This section describes how to use noise separation in the analog design environment (ADE).

Setting Up a Pnoise Analysis for Noise Separation

After you use the ADE GUI to set up both a PSS analysis and a Pnoise analyses, use the *Noise Type* section at the bottom of the Pnoise Choosing Analysis form to set up the noise separation analysis. When you set *Noise Type* as *sources*, the *Noise Separation* field is visible and you can select *yes* or *no*, as shown in Figure 1-35.

Notice the message below Noise Separation,
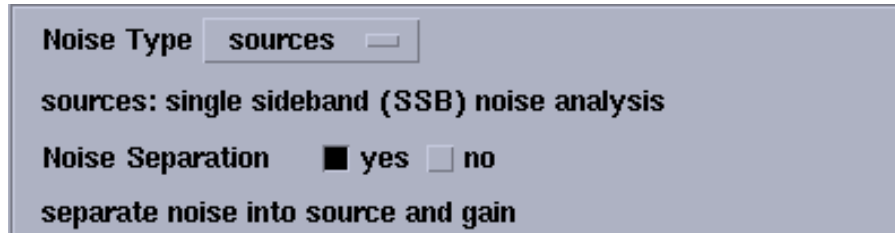
`separate noise into source and gain`

This message further describes noise separation in the current context.

Note: Please notice the following information as it applies to this context.

❑ There are two simulation engines: *Shooting* (TD) and *Flexible Balance (FB)*. Both simulation engines support noise separation. You select the simulation engine in the PSS analysis you set up to run before this Pnoise analysis.

❑ The *Noise Separation* field is visible and usable only after you set *Noise Type* to *sources*.

❑ After you set *Noise Separation* to *yes*, the *saveallsidebands* parameter in the Pnoise Options form is not required. The noise separation results include all the expected sidebands.

Figure 1-24 Setting up for Noise Separation in the Pnoise Choosing Analysis Form

When you run the PSS and Pnoise analyses, noise separation is measured during the simulation and the corresponding results are saved.

Setting Up A QPnoise Analysis for Noise Separation

After you use the ADE GUI to set up both a QPSS analysis and a QPnoise analysis, set up the noise separation analysis by selecting *yes* for *Noise Separation* as shown in Figure 1-36.

Notice the message below *Noise Separation,*
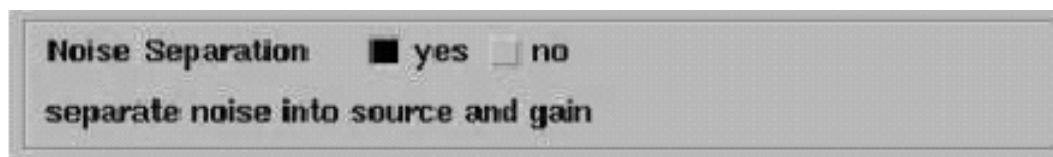
separate noise into source and gain

This message further describes noise separation in the current context.

Note: Please notice the following information as it applies to this context.

■   There are two simulation engines: *Shooting* (TD) and *Flexible Balance (FB)*. Both simulation engines support noise separation. (You select the simulation engine in the QPSS analysis you set up to run before this QPnoise analysis.)

■   After you set *Noise Separation* to *yes*, the *saveallsidebands* parameter in the QPnoise Options form is not required. The noise separation results include all the expected sidebands.

Figure 1-25 Setting up for Noise Separation in the QPnoise Choosing Analysis Form

When you run the QPSS and QPnoise analyses, noise separation is measured during the simulation and the corresponding results are saved.

Plotting Noise Sources, Gains, and Outputs

1. After running the noise separation analyses from ADE, select *Results — Direct Plot — Main Form* in the ADE Simulation window.

   Aside from *pss* and *pnoise* (or *qpss* and *qpnoise*), another choice for Analysis, *pnoise separation* (or *qpnoise separation*), displays in the Direct Plot form.

2. Select *pnoise separation* (or *qpnoise separation*) to create a noise separation plot.

   Six *Function* items are displayed. *Sideband Output*, *Instance Output*, *Source Output*, *Instance Source*, *Primary Source* and *Src. Noise Gain*.

3. Select one of the Functions.

Sideband Output	Plots the noise contribution made by multiple selected sidebands to the output noise
Instance Output	Plots the noise contribution of some instances (for example, MOS, or BJT) to the output at one selected sideband
Source Output	Plots the noise contribution made by one selected sideband to the output noise
Instance Source	Plots the noise sources of some instances at one selected sideband
Primary Source	Plots the primary noise sources (for example, re or rb in a BJT) at one selected sideband
Src. Noise Gain	Plots the noise gains of primary noise sources (for example, re or rb in a BJT) from source to output at one selected sideband.

   Notice the message below the *Function* area in the Direct Plot Form. For example, after selecting the function, *Sideband Output*, you see the following message.

   ```
   Noise contribution of sideband to output
   ```

   This message further describes the noise separation plot in the current context.

Selecting Other Information in the Direct Plot Form

You further define what RF simulation measurements to plot by entering and selecting items in the Direct Plot form; for example, functions, signal levels, modifiers, plots and so on.

1. While making selections in the Direct Plot form, follow new messages as they display.

Additional instructions and prompts might appear within the body of the Direct Plot form or at the bottom of the form. For example, after selecting *Sideband Output* for *Function* and *V**2 / Hz* for *Signal Level*, you can then select either *Power* or *dB20* for *Modifier*. You might also see a message such as the following.

Currently, only sideband data is available

This message accompanies the *Output Sidebands* list box.

2. After selecting *Power* for *Modifier* and all of the sidebands in the list box, you would be left with the following message.

Press plot button on this form...

The completed Direct Plot form displays.

3. As the last step in the Direct Plot form, click *Plot* to plot the noise contributed by each individual sideband.

The Plot displays as shown in Figure 1-26.

Figure 1-26 Output Sidebands Plot
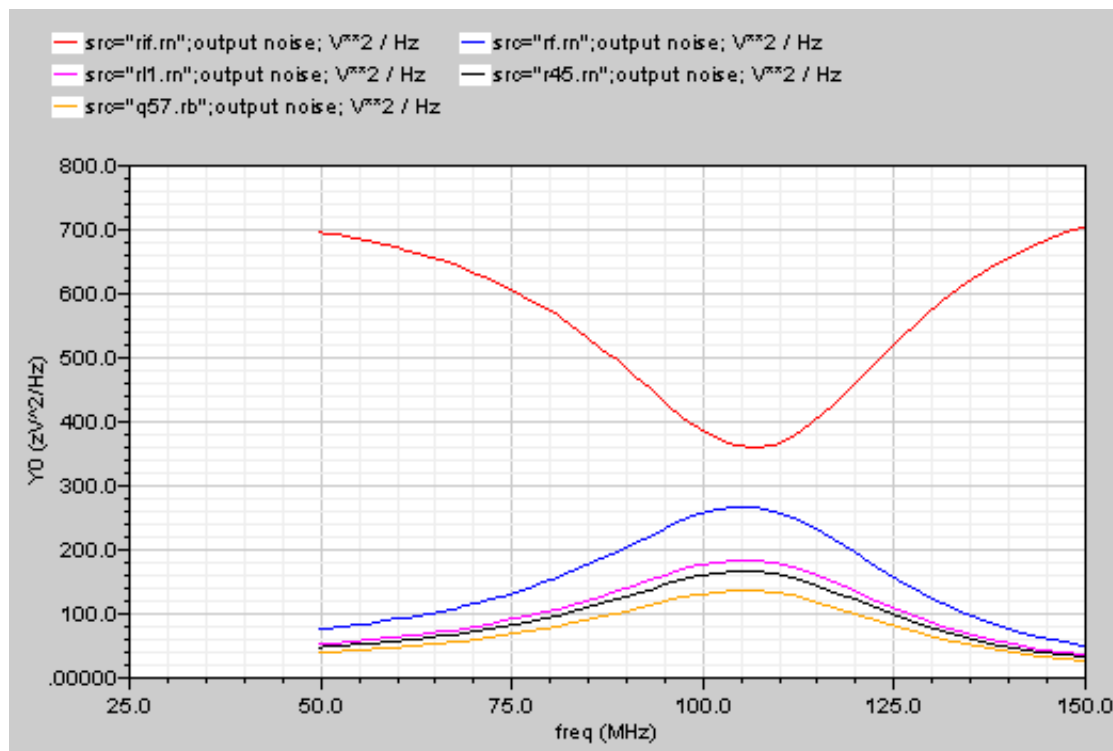
When you press the plot button (or perform another specified action, such as *Select Net in schematic*), a simulation plot appears, by default, in the current Waveform Window. If the

Waveform window or Schematic window is not open, selecting a direct plot function automatically opens both windows.

Brief instructions for all six noise separation plots follows.

Sideband Output

Plots the noise contribution of selected sidebands to the output.

Signal Level	Possible choices of measure units
Modifier	Possible measure scale according to the *Signal Level*
Output Sideband	A list box of all possible sidebands. It is multi-selection.

Instance Output

Plots the noise contribution of some instances; for example, MOS, BJT, and so on, to the output at one selected sideband.

Signal Level	Same as above
Modifier	Same as above
Output Sideband	Same as above, except single-selection
Filter	Frame
	Include All Types —All device types in right box selected
	Include None — Clear all selections in right box
	Include Inst — Only include these instances. Note the device types of these instances must be selected.
	Exclude Inst — Exclude these instances from the selected device types.
Truncate	Frame — Truncate the top number of *Instance Output*. The default is 3.

Instance Source

Plots the noise sources of some instances at one selected sideband.

Signal Level	Same as above
Modifier	Same as above
Output Sideband	Same as above
Filter	Same as above
Truncate	Same as above

Source Output

Plots the noise contribution of the primary noise source (for example, re and rb in a BJT) to the output at one selected sideband.

Signal Level	Same as above
Modifier	Same as above
Output Sideband	Same as above
Filter	Same as above
Truncate	Frame — Truncate the top number of *Source Output*. The default is 3.

Primary Source

Plots the primary noise sources (for example, re and rb in a BJT) at one selected sideband.

Signal Level	Same as above
Modifier	Same as above
Output Sideband	Same as above
Filter	Frame — Same as above

Truncate	Frame — Same as above

Source Noise Gain (Src. Noise Gain)

Plots the noise gains of primary noise sources (for example, re and rb in a BJT) from source to output at one selected sideband.

Signal Level	Same as above
Modifier	Same as above
Output Sideband	Same as above
Filter	Frame — Same as above
Truncate	Frame — Same as above

Printing the Noise Source Summary Results

You can also view all noise source information in the Noise Summary Print form.

➤ Select *Results — Print — Noise Summary* in the ADE Simulation window.

In the Noise Summary form, aside from the `pnoise` and `qpnoise` items, another item, `pnoise_src` (or `qpnoise_src`) is displayed. The field structure of the form, as well as the `fill`, `filter` and `truncate` methods, are the same as those in the `pnoise summary` (or `qpnoise summary`) Plot form.

In print form, the Noise Source Summary provides an overview of noise source distribution in the circuit under analysis. The organizational structure of the Noise Source Summary is similar to the Output Noise Summary, but the latter provides the contributions to the output of these noise sources.

Beyond providing an overview of the source distribution, the Noise Source Summary is also useful in the following cases. The Noise Source Summary helps you to

■ Easily locate obviously abnormal sources. For example, such as a particular noise source with unreasonable PSD according to your experience or knowledge.

■ Quickly verify some circuit design specifications. For example, in a low noise amplifier (LNA) design, you need the noise figure (NF) in order to evaluate input matching.

Suppose the inner resistance of a signal is Rs, and the equivalent input resistance of the LNA is Rp, then NF simply equals

$$1 + Rs/Rp$$

Since you can express Rp as a simple function of rb, Rs for a Bipolar LNA, with the Noise Source Summary at hand, you can easily estimate the NF by hand.

Displaying the Noise Ssource Information in WaveScan

You can also view all noise source information in WaveScan.

➤ Select *Tools — Results Browser* in the ADE Simulation window.

Several data results are listed in In the Results Browser. Among them

❑ pnoise-sources pnoise or qpnoise-sources qpnoise — includes noise source information,

❑ pnoise-xfersrcs.pnoise or qpnoise-xfersrcs.qpnois— includes noise source, gain and output noise information.

You can plot the selected results in the waveform window, post processed in calculator and displayed in table.

The Noise Separation Flow

The hierarchical function layout, filter and truncation algorithms in the Direct Plot form, provide a good way to quickly locate the noise sources which cause the most noise in the output noise. The hierarchical layout is based on the hierarchy collection of output noise, which is simply described as follows.

■ The total output noise equals to the sum of output noise at each sideband

■ The output noise at one sideband equals to the sum of instance output noise at this sideband. An instance can be a MOS, a BJT or a resistor etc.

■ The output of one instance at one sideband equals to the sum of primary source output in this instance at this sideband. That is, a BJT includes re, rb, ic et al primary sources.

The Basic Noise Separation Flow

Direct Plot Form -> Pnoise Separation (or Qpnoise Separation) ->

Sideband Output[1] -> Instance Output[2] ->

Source Output[3] -> Instance Source/Primary Source/Src. Noise gain[4]

Note:

[1] Decide which sidebands contribute the most output noise

[2] Decide which instances contribute the most output noise to the selected sideband

[3] Decide which primary noise sources contribute the most output noise to the filtered and truncated instances

[4] Locate which primary noise sources or gains have the greatest effect on output noise

Noise Separation Example

This example shows how to locate the noise sources that contribute the most Output Noise. It uses the NE600 mixer is in the *rfExamples* library. The schematic is shown in Figure 1-27. Suppose the RF input signal is frf=900 MHz, LO signal flo=1 GHz, the expected IF should be 100 MHz.

Figure 1-27 NE600 Mixer Schematic
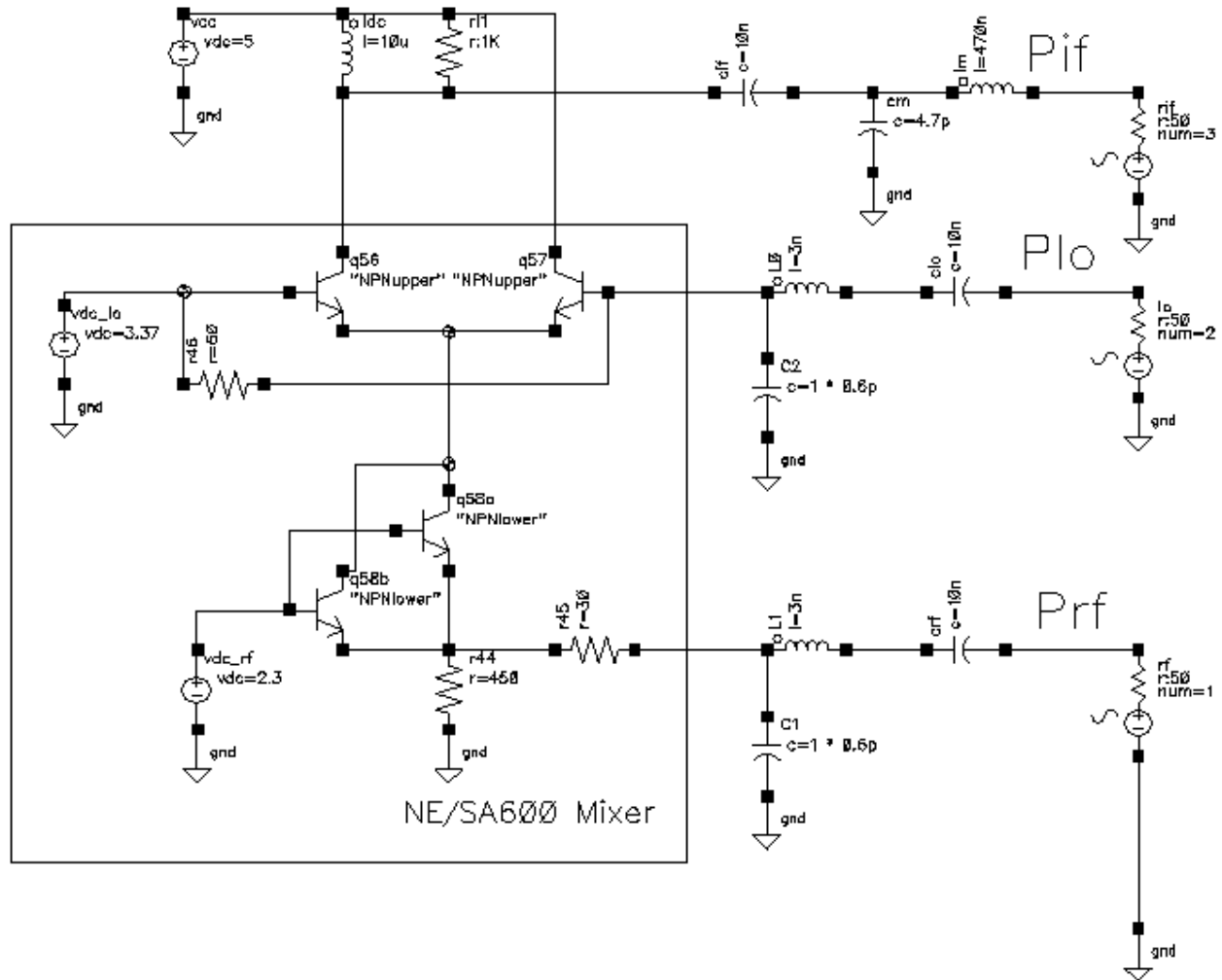
Setting Up and Running a Pnoise Separation Analysis

Using the ADE GUI

Stimulus

Set up the following stimuli:

A large sinusoidal signal at the LO port (*PORT2)*

Set the RF port (*PORT1*) source `type=dc`

Parameters

Set up the following parameters:

```
flo=1G

prf=-30dBm
```

Simulation/Analyses

1. Set up the following PSS analysis.

Beat frequency: *flo*

Output harmonics: *Number of harmonics 10*.

2. Set up the following Pnoise analysis.

Sweeptype: *default*

Output Frequency Sweep Range (Hz):

Start *50M*

Stop *150M*

Sweep Type: *Logarithmic*

Points Per Decade: *100*

Sidebands: *Maximum Sideband 5*

Output: *probe*

Set up as the Output Probe Instance: */rif*

Input Source: *port*

Set up as the Input Source Port */rf*

Reference side-band: *Enter in field -1*

Noise Type: *sources*

Noise Separation: *yes*

3. Run the PSS and Pnoise simulations.

Using the Flow to Locate Noise Sources

1. Plot the Sideband Output.

Drag the mouse to select all sidebands (from -5 to 5). In this case, the result is shown in Figure 1-28.

Figure 1-28 Sideband Output of the ne600 with Pnoise Separation
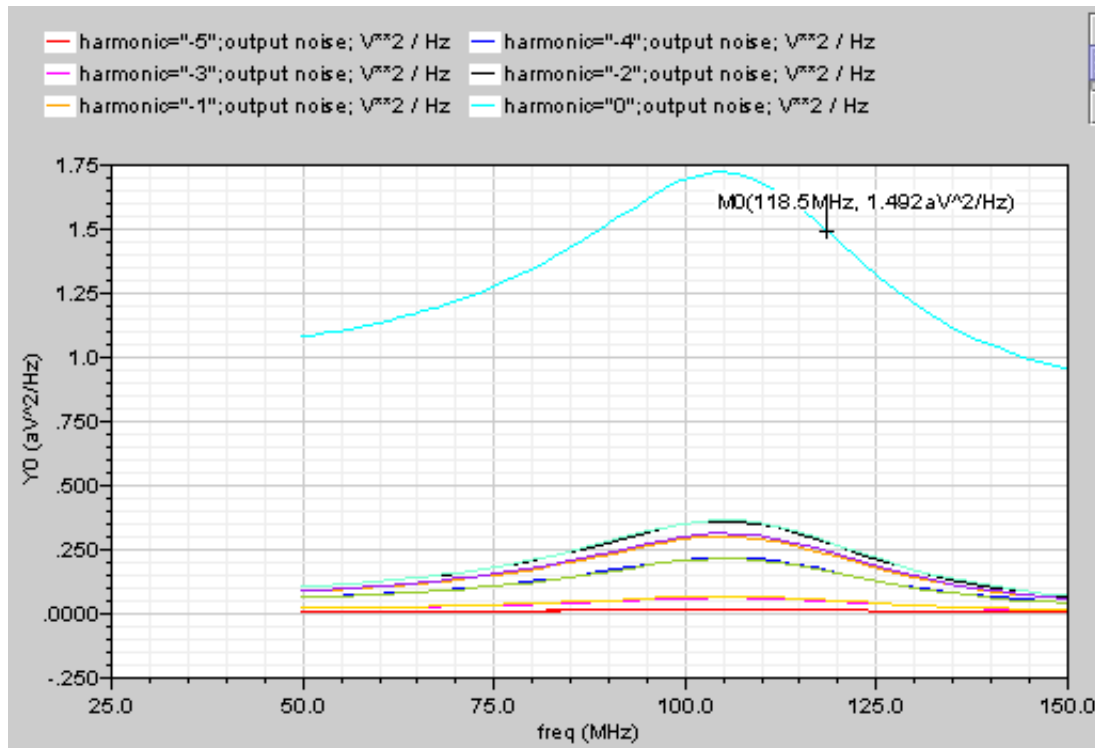
From Figure 1-28, you can see that Sideband 0 contributes more output noise than any other sideband. Therefore, the next step is to check the instances in sideband 0.

2. Plot the *Instance Output.*

❏ Set output sideband to 0

❏ Include all types

❏ Top number is 5

The result is shown in Figure 1-29.

Figure 1-29 Instance Output of ne600 with Pnoise Separation
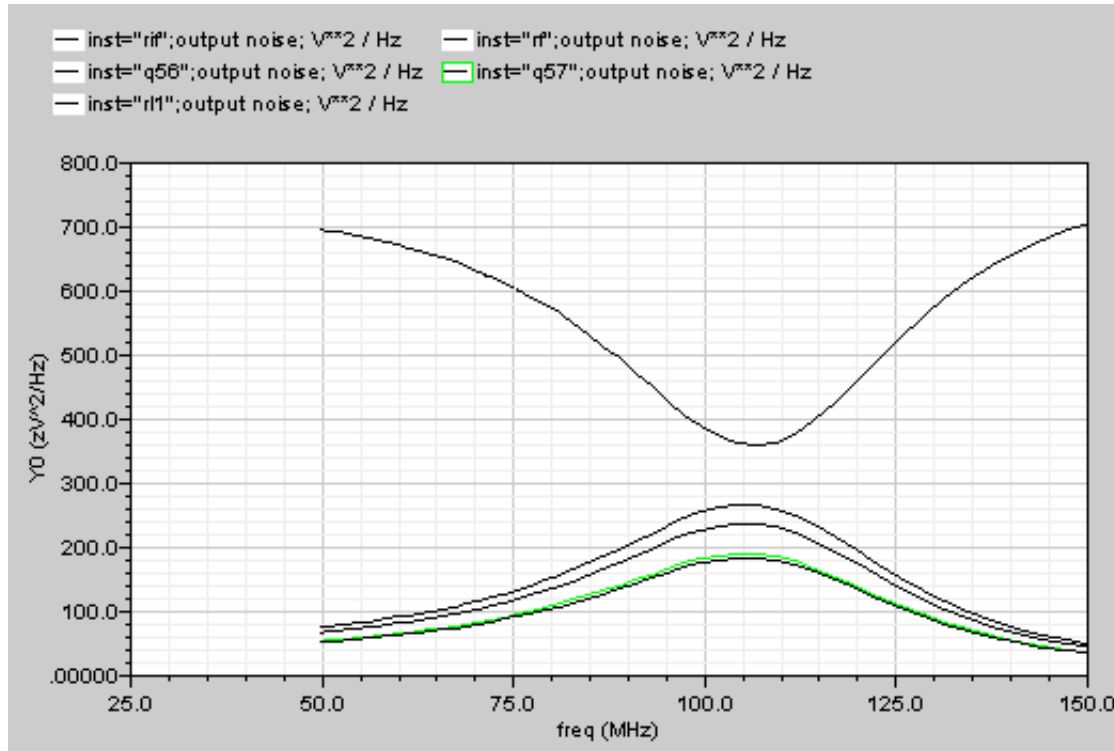
From Figure 1-29 you can see that rif, rf, q56, q57 and rl1 (especially the rif) contribute more output noise than other instances.

3. Plot the *Source Output.*

❏   Set **output sideband**: 0, **include all types**, **top number**: 5,

The result is shown in Figure 1-30.

Figure 1-30 Source Output of ne600 with Pnoise Separation
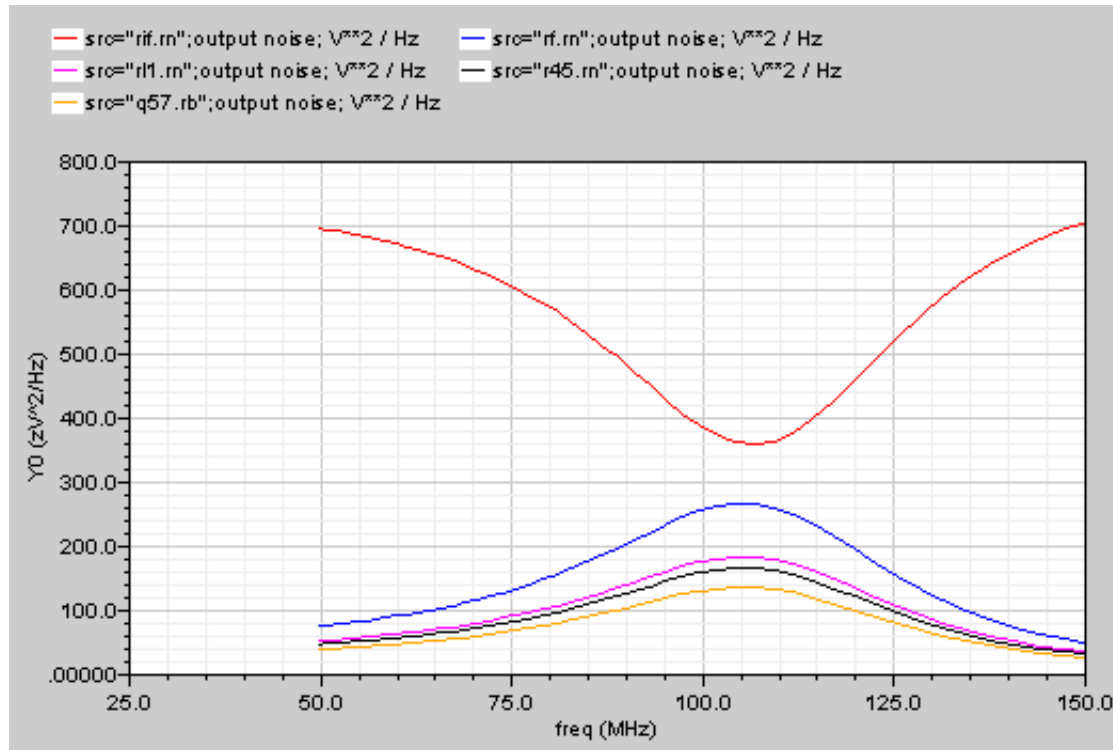
In Figure 1-30, you can see that rif.rn, rf.rn, rl1.rn, r45.rn and q57.rb (especially rif.rn) contribute more output noise than other noise sources.

Note: The list order of instances in this plot *Source Output* (in Figure 1-30) is different from the list order of *Instance Output* in Figure 1-29.

Up to now it is obvious that rif.rn is the main contributor to the output noise in this circuit. Since rif only includes one noise source rif.rn, The *Instance Source* and the *Primary Source* in Figure 1-31 should give the same plot. You can further check the *Primary Source* and *Src. Noise Gain* of rif.rn.

4. Plot Primary Source and Src.Noise Gain

set **output sideband**: 0, **include all types**, **top number**: 1, the result is shown in Figures 1-31 and 1-32.

Figure 1-31 Primary Source of ne600 with Pnoise Separation
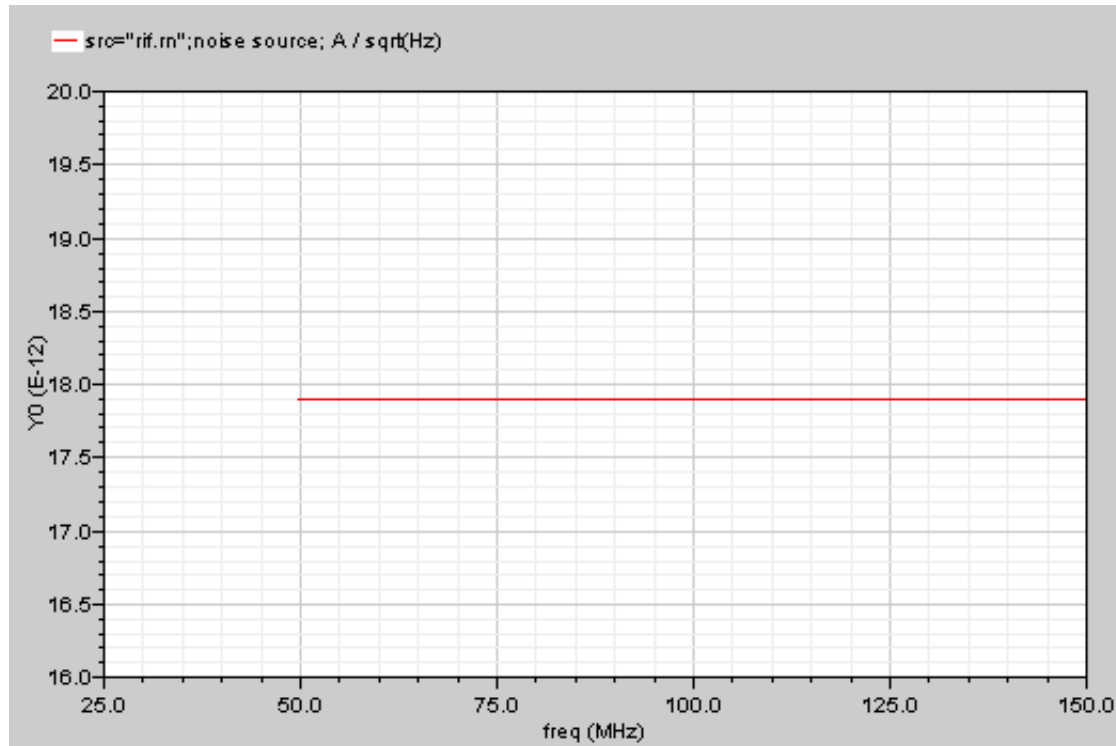
Figure 1-32 Source Noise Gain of ne600 with Pnoise Separation
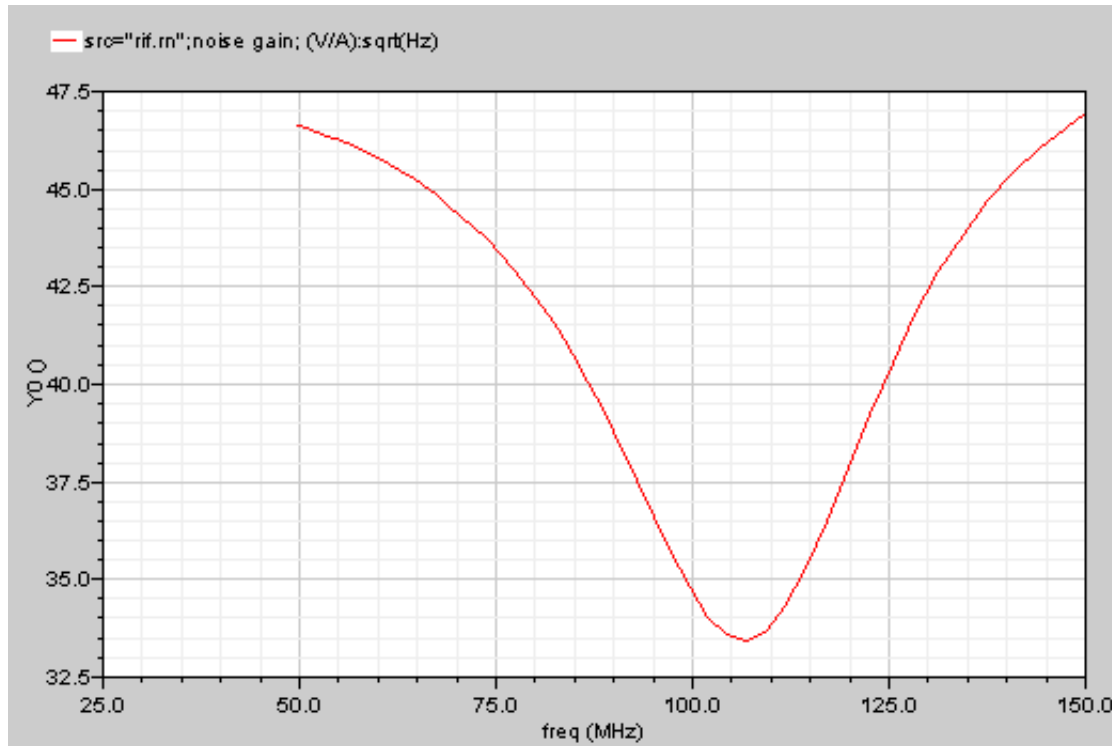
To improve the noise performance of this circuit, decreasing the output noise of rif.rn is an effective solution.

There are two approaches

❏ One is decreasing the magnitude of noise source rif.rn by adjusting the device geometric size

❏ The other is decreasing the transfer function of rif.rn by adjusting the circuit architecture.

Other Notes

■ The **Instance Output** and **Instance Source** use the same truncation methods — truncation is by the top values of instance output noise. The **Source Output**, **Primary Source** and **Src. Noise Gain** use another truncating method — truncation is by the top number of the primary noise source output. Hence the list order of the former two plots may be different from the later three. If you want to check one instance in **Instance Output/Instance Source** and its primary sources in **Source Output/Primary Source/ Src. Noise Gain** at the same time, you may be required to set different truncation values.

■ An instance of an active device such as MOS, BJT includes several primary sources, the orders of magnitude. Some instances such as a thermal resistor generally include one primary source, whose PSD is *4KTG* where *G* is the conductance.

■ When adjusting the circuit architecture by decreasing noise gain or the a transfer function of one noise source, might enlarge the transfer functions of other noise sources. Thus, you should consider using more global optimization.

■ The Noise summary print form gives an overview of noise source distribution. It is useful for finding out which noise sources are abnormal. You can use WaveScan to check the result noise data. WaveScan is also useful for locating the faults in binomial `pnoise` (or `qpnoise)` analysis results.

Simulating Noise and Jitter

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 modeled as periodically time-varying. Noise that has periodically time-varying properties is said to be cyclostationary.

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 1-33.

Figure 1-33 Very Simple Mixer Schematic
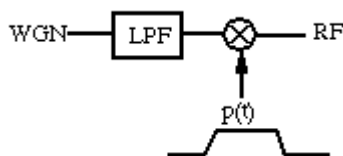
The amplitude of the noise measured at the RF output shown in Figure 1-33 periodically varies depending on the magnitude of the modulating signal *p(t)*, as shown by the sample points in Figure 1-34.

Figure 1-34 Time-Varying Noise Process Analyzed at ξ_1 and ξ_2
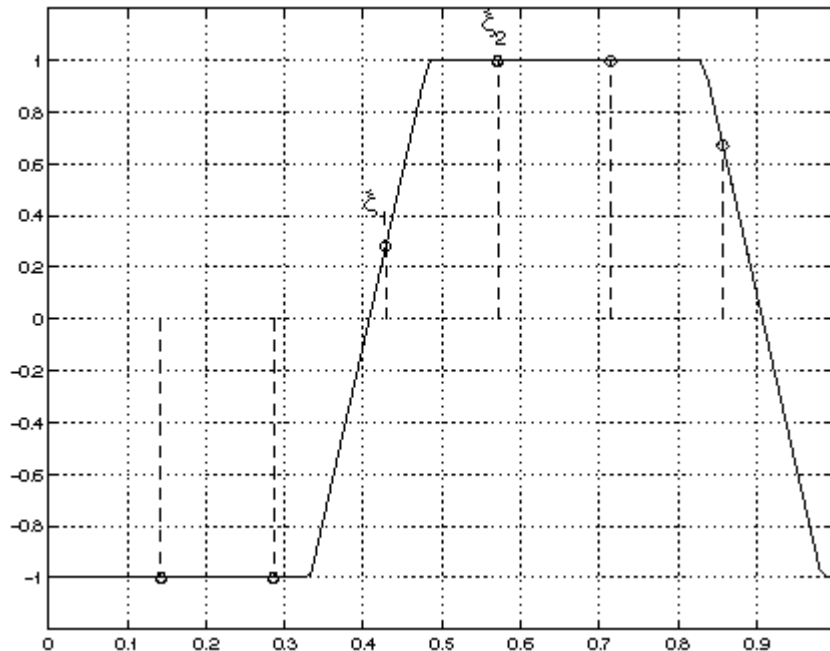
In Figure 1-34

- 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 1-33 on page 105. 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 1-34 on page 106.

- Figure 1-34 is a representation of periodically-modulated noise. It shows noise processes for two different phases in the periodic interval. Each process is stationary.

Spectre$^{®}$ circuit simulator RF analysis (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 1-35 and 1-36 show two such noise processes for two different phases in the periodic interval. Each process is stationary. Figure 1-35 shows the noise process for the phase marked ξ_1 in Figure 1-34 on page 106.

Figure 1-35 Noise Process for Phase ξ_1
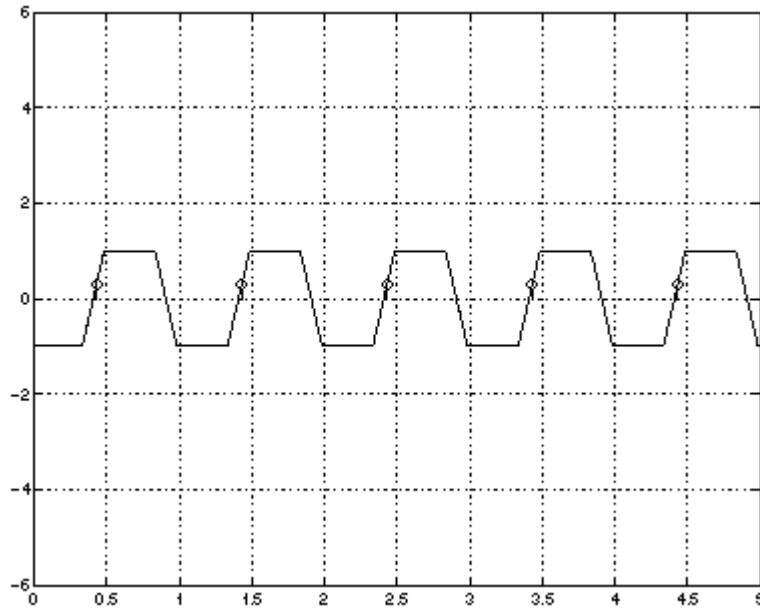
Figure 1-36 shows the noise process for the phase marked ξ_2 in Figure 1-34 on page 106.

Figure 1-36 Noise Process for Phase ξ_2
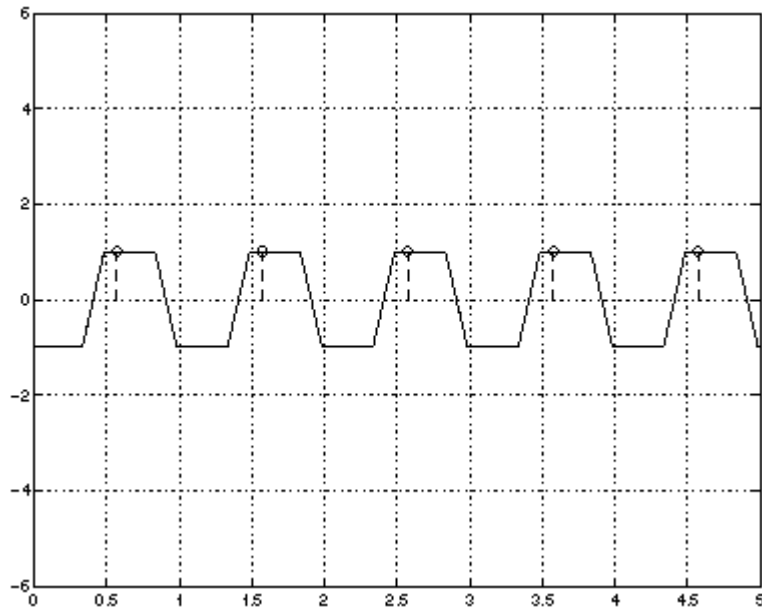
See the "Reference Information on Time-Varying Noise" on page 120 for a more detailed introduction to noise in periodically time-varying systems.

Calculating Time Domain Noise

The following steps tell you how to calculate time-domain noise using SpectreRF.

1. In a terminal window, type `icms` (in IC5141) and virtuoso (in IC 6.1.4) to start the environment.

2. In the ADE 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.*

5. The Choosing Analyses form changes to let you specify information for a Pnoise analysis.

6. 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.

e. 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.

7. In the ADE window, choose *Simulation – Netlist and Run.*

8. The simulation runs.

9. In the ADE window, choose *Results – Direct Plot – PSS.*

The PSS Results form appears.

10. To calculate time-varying noise power, perform the following steps in the PSS Results form:

a. Click *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.

11. 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.

12. To calculate the autocorrelation function for one of the sampled processes, perform the following steps:

    a. Display the spectrum using instructions from <u>step 11</u>.

    b. In the ADE window, choose *Tools – Calculator*.

    c. The calculator appears.

    d. Click *wave* in the calculator and the select the appropriate frequency-domain spectrum.

    One of the sample waveforms is brought into the calculator

    e. 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.

    f. Choose an appropriate window (e.g., Cosine2) and number of samples (around the number of frequency points in the interval [0,1/T]),

    g. Apply the *DFT* and plot the results.

    Harmonic q of the DFT results gives the value of the discrete autocorrelation for this sample phase, $R(q)$.

    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

$$S_{XY}^{\alpha}(\omega)$$

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.

After 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 1-37, first you measure the noise voltage appearing at the excitation ports of the circuit on the left in Figure 1-37. 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 1-37 Calculating Noise Correlations and Equivalent Noise Parameters
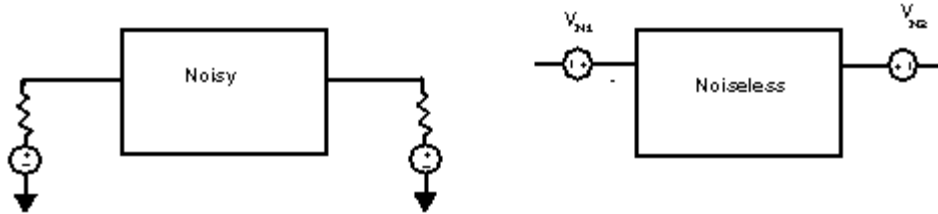
When you know the noise at each port and its correlation, you can obtain any of various sets of equivalent noise parameters. For example, you can express noise in an impedance representation as the equivalent correlated noise voltage sources shown in Figure 1-37, as equivalent noise resistances and the correlation parameters, and as F_{min}, R_N, G_{opt}, and B_{opt}.

Cyclostationary Noise Example

As an example which illustrates the various aspects of cyclostationary noise, consider the simple mixer circuit shown in Figure 1-38.

Figure 1-38 Simple Mixer Circuit
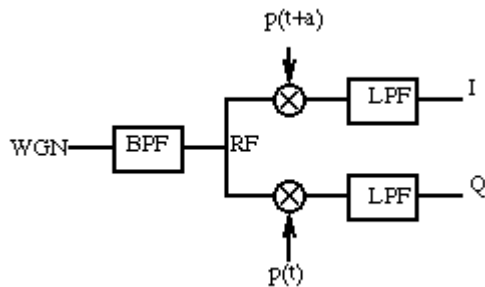
In this simple mixer circuit, white Gaussian noise passes through a high-order band-pass filter with center frequency ω_0. Then it is multiplied by two square-waves which have a phase shift a with respect to each other. Finally the output of the ideal multipliers is put through a one-pole low-pass filter 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 1-39 on page 113. 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 1-39 Time-Varying Noise Power Before Low-Pass Filter
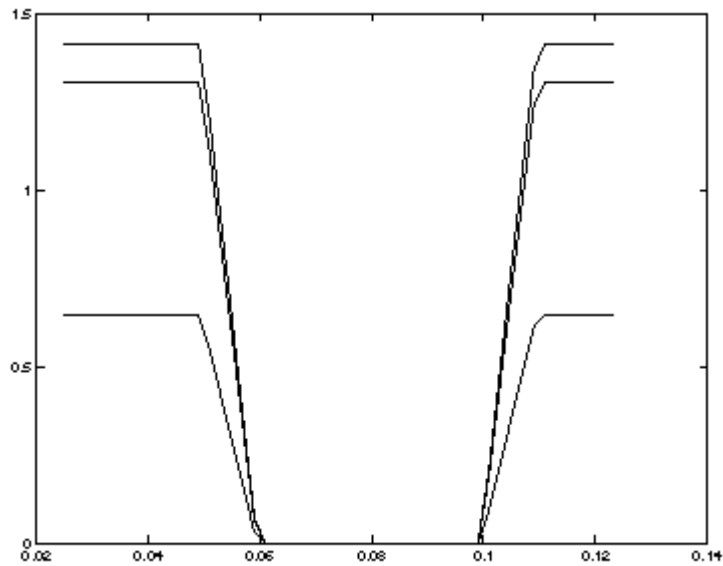
Figure 1-40 shows the spectrum of a sampled noise process. Note the periodically replicated spectrum.

Figure 1-40 Spectrum of a Sampled Noise Process
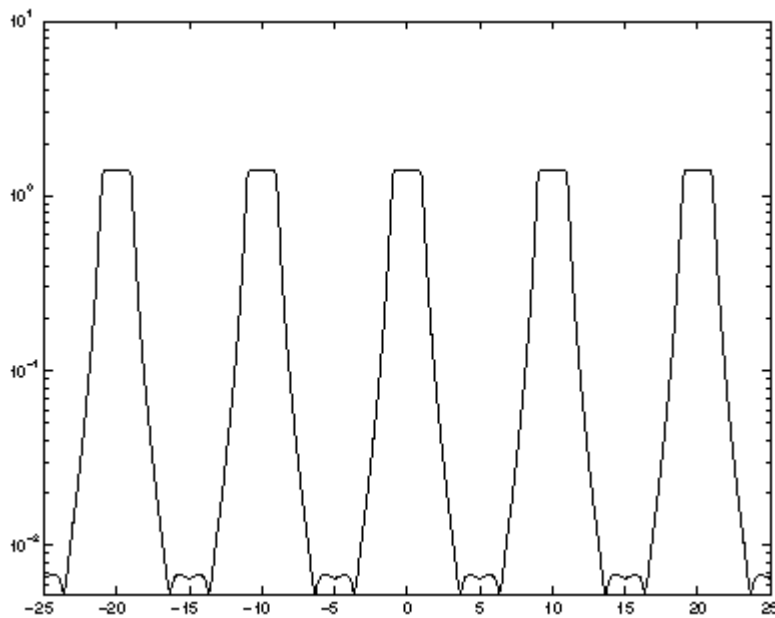
The noise behavior at the output ports is examined next. The output spectra at the I and Q outputs are shown in Figures 1-41 and 1-42. The noise density at I is concentrated around

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 1-41 Power Spectra With LO Tones 90^{deg} Out of Phase
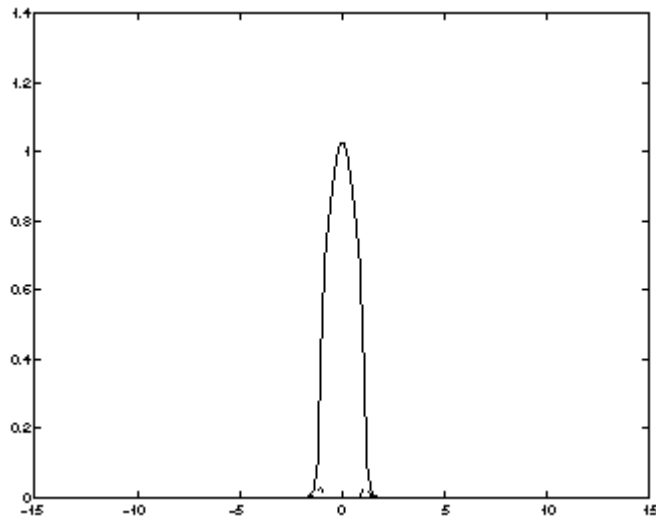
More interesting is the cross-correlation spectrum of the I and Q outputs, shown as the dashed line in Figures 1-41 and 1-42. When the signals applied to the mixers are 90 degrees out of phase (as in Figures 1-41), the cross-power spectral density of the noise at the separate I and Q outputs is small, indicating little noise correlation. If the tones are not quite out of phase (as in Figures 1-42), the correlation is much more pronounced, though in neither case is it completely zero.

In Figures 1-41 and 1-42, 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 1-42 Power Spectra With LO Tones 72^{deg} Out of Phase
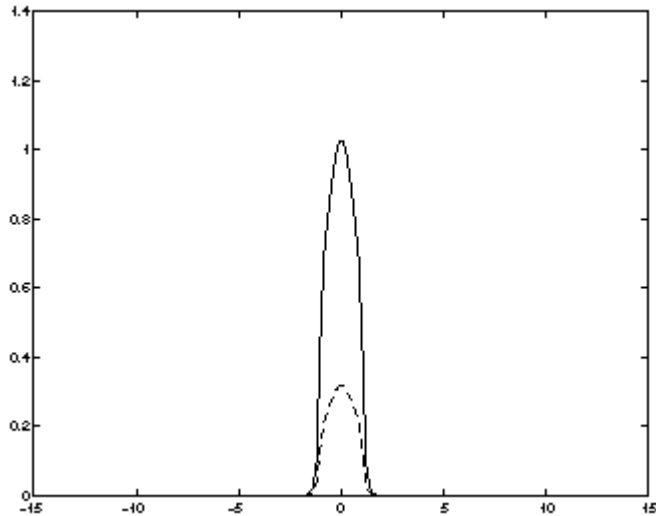
A more interesting example comes from examining the correlation between the noise at the *I* output and the noise at the RF input. The density function as given by

$$S_{IR}^{(1)}(\omega)$$

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 1-43 Noise Spectrum at the RF Input
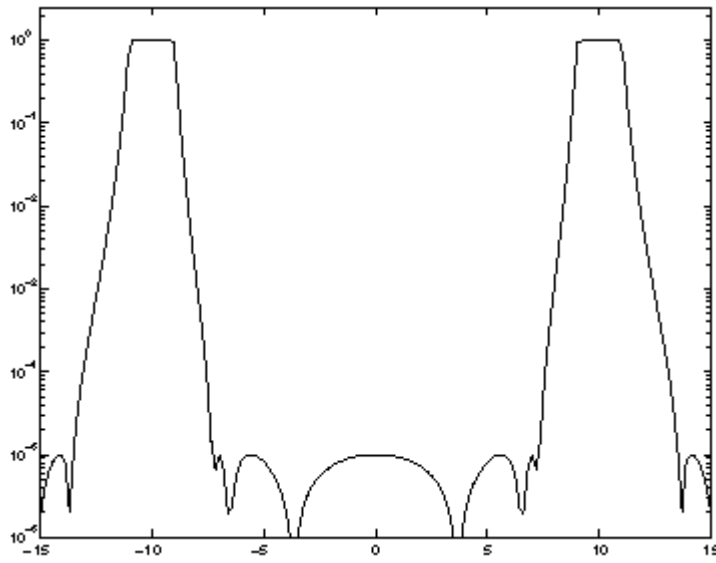
In Figure 1-43, the noise spectrum at the RF input is given by the following function

$$S_{RR}^{0}(\omega)$$

Figure 1-44 Noise Spectrum at Various Power Densities
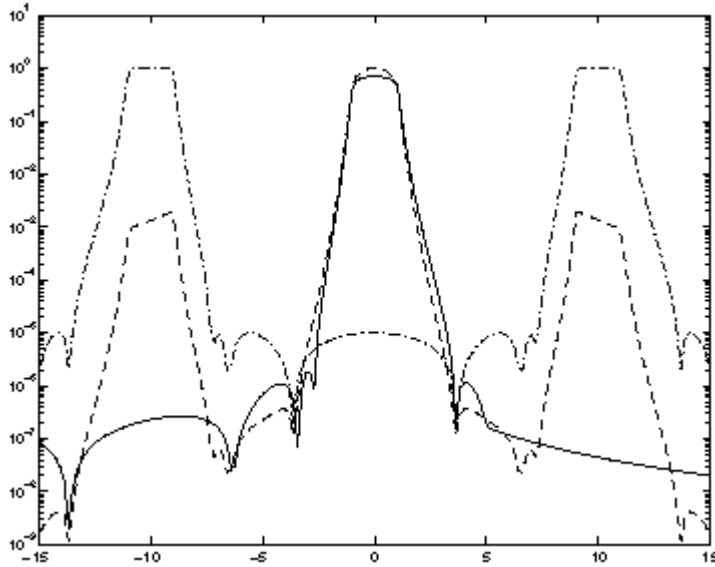
In Figure 1-44, the solid, dashed, and dashed-dot lines represent the following

■   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

$$S_{IR}^{(1)}(\omega)$$

■   The dashed line represents the noise spectrum at the I output with the following function

$$S_{II}^{(0)}(\omega)$$

■   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 1-45.

Figure 1-45 Power Spectral Densities of an Image-Reject Receiver
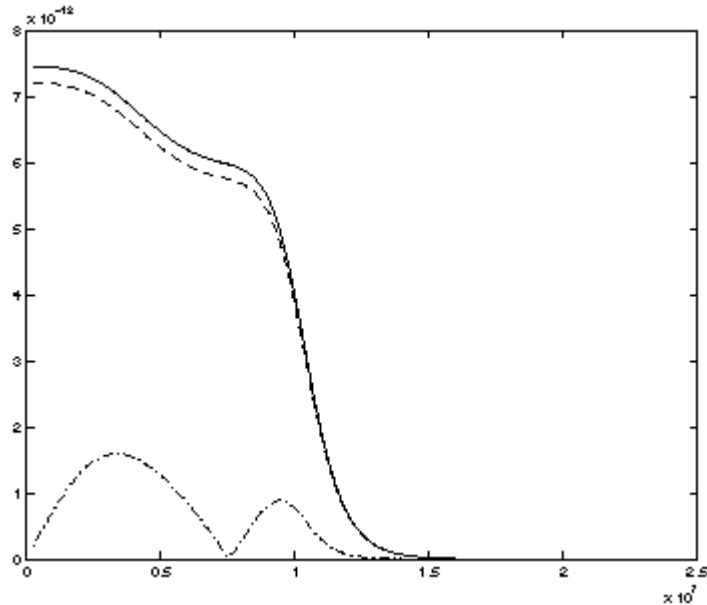
In the image-reject receiver example shown in Figure 1-45, the power spectral densities are represented as follows

■ *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 1-46.

Figure 1-46 Correlation Coefficient for Output Noise of Image-Reject Receiver
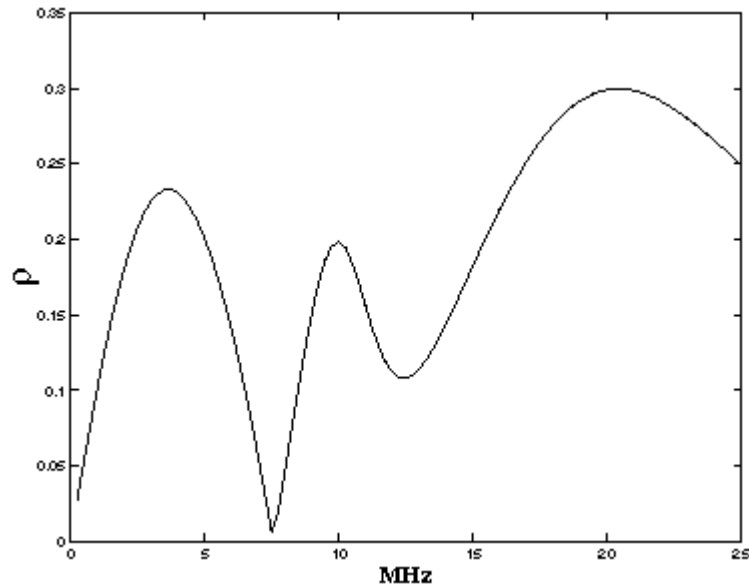
Reference Information on Time-Varying Noise

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 1-47.

Figure 1-47 Deterministic Current Source Driving a Noisy Linear Resistor
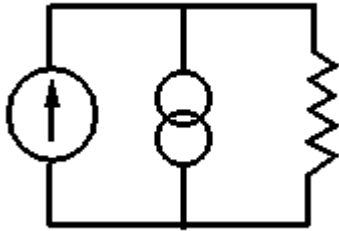
The voltage that appears across the resistor is

$$v(t) = i(t)R + n(t)$$

The desired signal $i(t)R$, shown in Figure 1-48, 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 1-48 The Desired Signal i(t)R
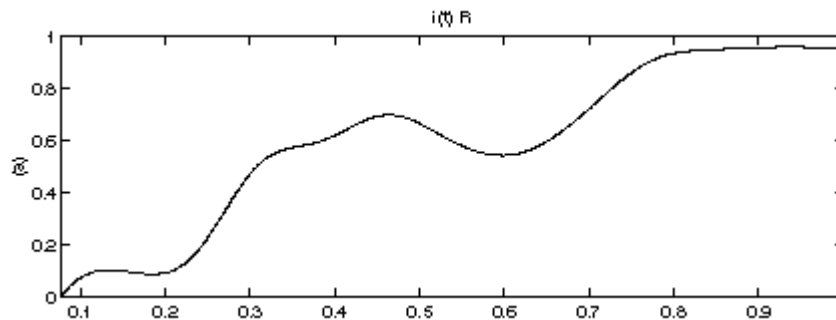
The total measured voltage is shown in Figure <u>1-49</u>.

Figure 1-49 The Total Measured Voltage
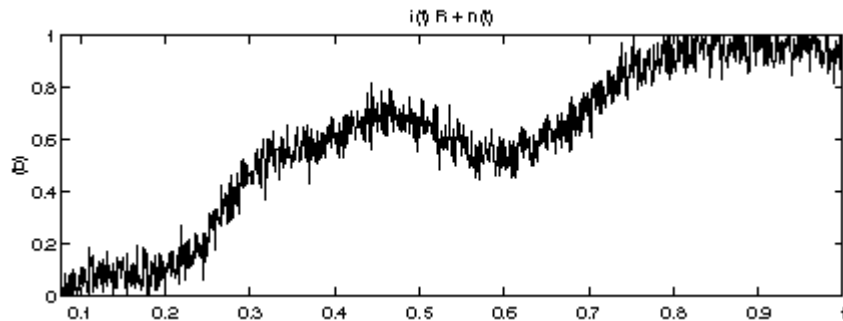
The added noise process alone, *n(t)*, is a random process and so it must be characterized in ways that are different than for deterministic signals. That is, at a time t_0 the voltage produced by the driven current can be exactly specified—it is $i_0 sin t_0 R$. Just by inspecting Figure 1-48 we can 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 is different and Figure 1-49 changes. Due to its innate randomness, we must use a statistical means to characterize *n(t)*.

Now consider the circuit in Figure 1-50, where we restrict attention to the noise source/ resistor pair alone.

Figure 1-50 Resistor Modeled as a Noiseless Resistance with an Equivalent Noise Current Source
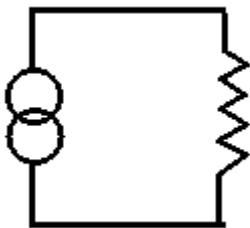
A typical measured noise current/voltage is shown in Figure 1-51.

Figure 1-51 Typical Measured Noise Current/Voltage
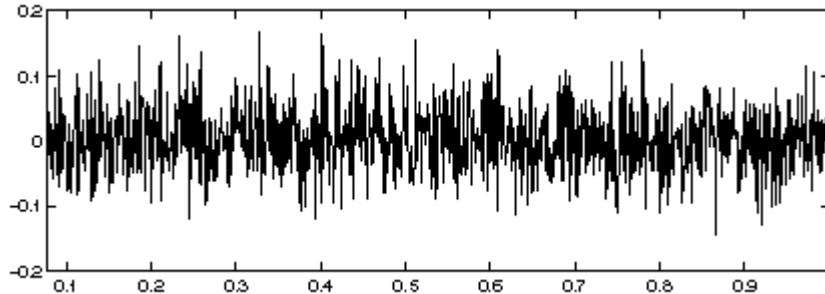
Because we cannot predict the specific value of $n(t)$ at any point, we might instead try to predict what its value would be on average, or what we might *expect* the noise to be. For example, if we measure many noise voltage curves in the time domain, $n(t)$, and average over many different curves, we obtain an approximation to the expected value of $n(t)$ which we denote by $E\{n(t)\}$. For thermal noise, we 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 1-52. 250 measurements were needed to compute this curve.

Figure 1-52 Expected Noise Power
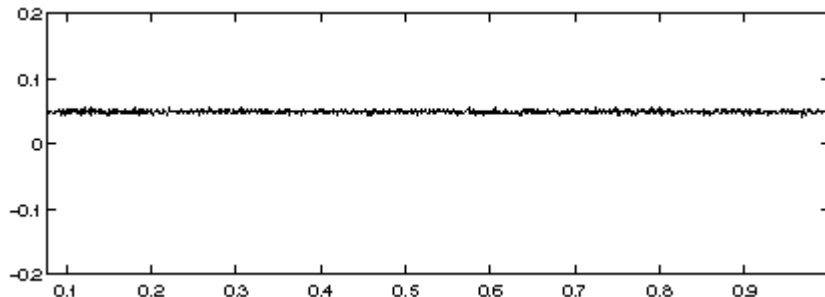
Now suppose that we wish to tap the circuit at multiple points. Each point has its own noise characteristics, but they are not necessarily independent. Consider the circuit shown in Figure 1-53.

Figure 1-53 Circuit Illustrating Correlated Noise
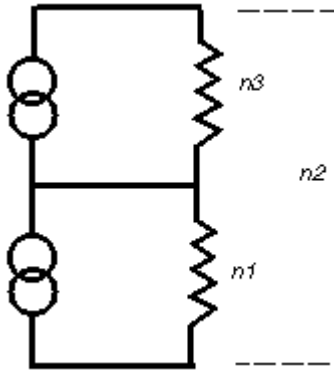
The signals $n_1(t)$ and $n_2(t)$ are obtained by measuring the voltage across a single resistor ($n_1(t)$), and across both resistors ($n_2(t)$), respectively. Just measuring $E\{n_1(t)^2\}$ and $E\{n_2(t)^2\}$ is not enough to predict the behavior of this system, because $n_1(t)$ and $n_2(t)$ are not independent.

To see $n_1(t)$ and $n_2(t)$ are not independent, consider Figures 1-54 and 1-55. Samples of each of the processes are taken and plotted on an X-Y graph.

Figure 1-54 Samples of n$_1$(t) Plotted Versus n$_2$(t)
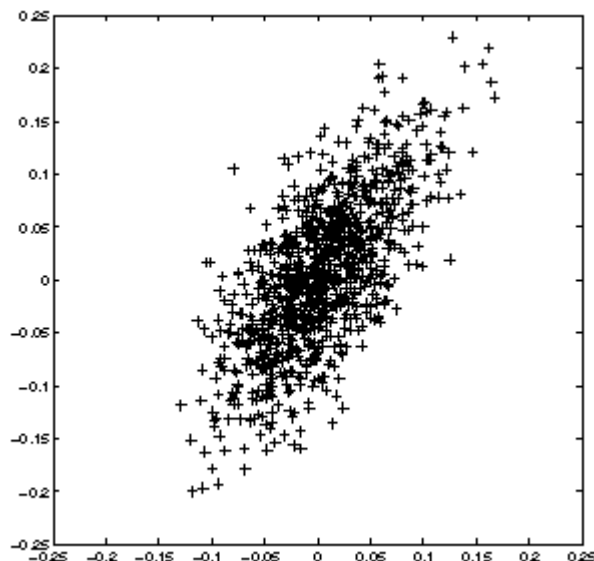
Because $n_1(t)$ composes part of $n_2(t)$, $n_1(t)$ and $n_2(t)$ are correlated so in Figure 1-54, the X-Y plot has a characteristic skew along the $X=Y$ line, relative to the $n_1(t)$, $n_3(t)$ plot in Figure 1-55,

Figure 1-55 Samples of $n_1(t)$ Plotted Versus $n_3(t)$
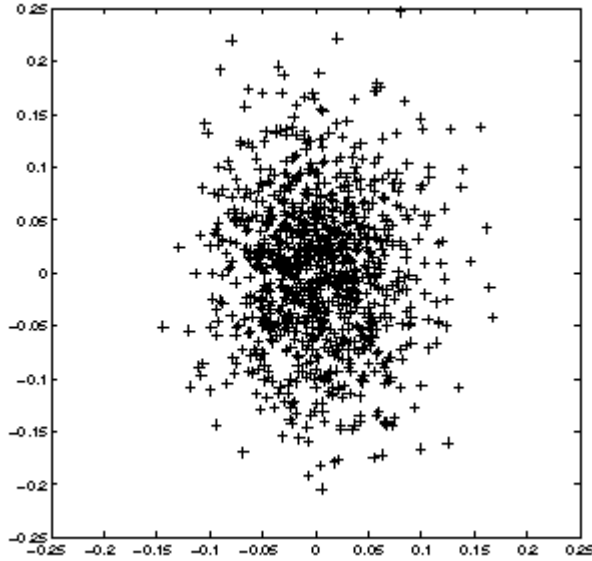
The signals $n_1(t)$ and $n_3(t)$ are uncorrelated because they represent thermal noise from different sources. The additional measurement needed to describe the random processes is the measurement of the correlation between the two processes, $E\{n_1(t)n_2(t)\}$. We can also define a time-varying correlation 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 a first example, consider adding a capacitor in parallel to the simple resistor, as shown in Figure 1-56.

Figure 1-56 A Simple RC Circuit
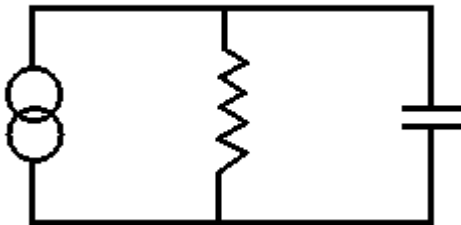
A sample of the noise process in shown in Figure 1-57.

Figure 1-57 Noise Process for a Simple RC Circuit
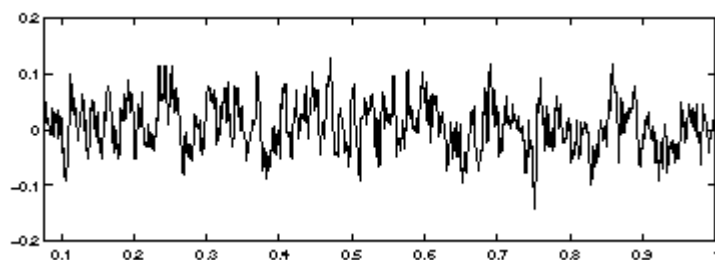
The noise looks different than the noise of the resistor alone, because the low-pass filter action of the RC circuit eliminates very high frequencies in the noise. However, we cannot see this effect simply by measuring $E\{n(t)^2\}$ as shown in Figure 1-58.

Figure 1-58 Expected Noise Power for an RC Circuit
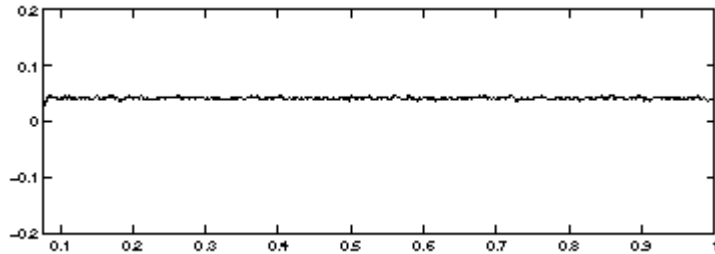
The measurement of $E\{n(t)^2\}$ is independent of time for an RC circuit, just as it was for the for the resistor circuit.

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 1-58 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*.

Power Spectral Density for Resistor Thermal Noise
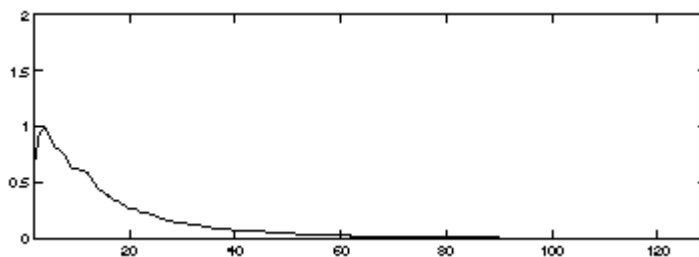
Figure 1-59 shows the spectrum of the noise process after filtering by the resistor-capacitor system.

Figure 1-59 Resistor-Capacitor Filtered Spectral Noise Process
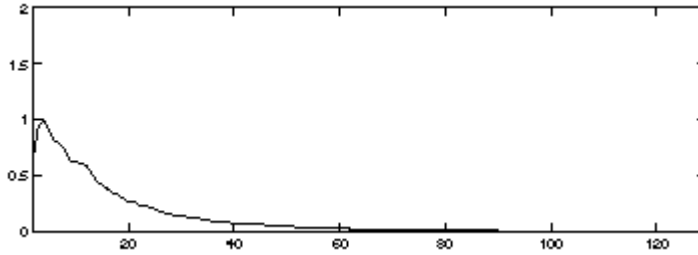
It is easy to rigorously account for the effect of the RC-filter on the power spectrum of the noise signal. Suppose a random signal x is passed through a time-invariant linear filter with frequency-domain transfer function $h(\omega)$. Then the output is $y(\omega)=h(\omega)x(\omega)$.

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 $Z(\omega)$,

$$h(\omega) = Z(\omega) = \frac{(1/C)}{j\omega + \dfrac{1}{RC}}$$

and so the noise voltage power density is

$$S_n(\omega) = \frac{\dfrac{4k_B T}{RC}}{\omega^2 + \left(\dfrac{1}{RC}\right)^2}$$

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 is much smaller than the output total noise power density if the amplifier has small gain.

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 1-60.

Figure 1-60 SimpleTime-Varying Circuit with Switch
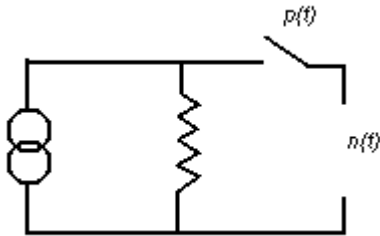
Further suppose that the switch is periodically opened and closed. When the switch is open, there is no noise measured. When the switch is closed, the thermal noise is seen at the voltage output. A typical noise waveform is shown on the bottom left in Figure 1-61.

Figure 1-61 Typical Waveforms for Noise, Noise Power and White Noise
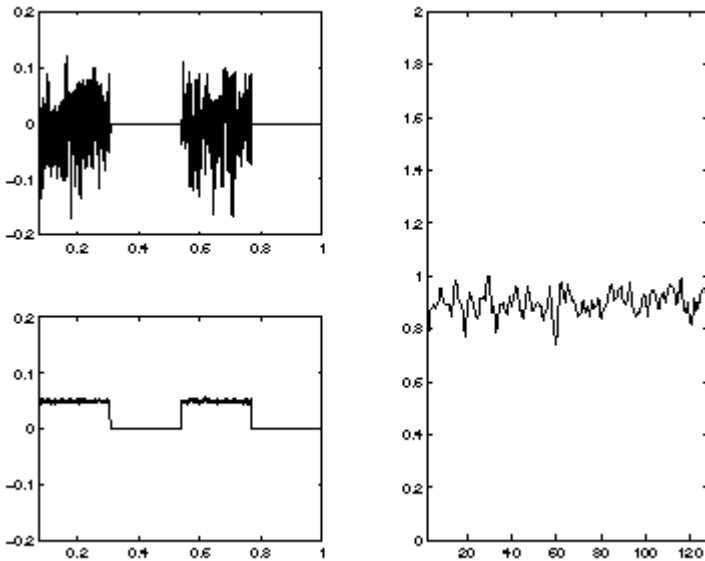
The time-varying noise power $E\{n(t)^2\}$ can be computed and is shown in Figure 1-61 on the top left, above the time-varying noise waveform. The expected power periodically switches between zero and the value expected from the resistor noise. This is different than the resistor-only and resistor-capacitor 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 1-61, 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 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 because $n(t)$ 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 $R_x(t, t+\tau)$ because 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 because 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

$$E\{c_k c^*_L\} = |c_k|^2 \varsigma_{kl}$$

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

$$\omega_0 = \frac{2\pi}{T}$$

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.

Because $h(t)$ 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

$$S_v(\omega) = E\{\langle v(\omega)v(\omega)^* \rangle\} = \sum_k \sum_l h_k h_l^* E\{n(\omega - k\omega_0)n(\omega - l\omega_0)^*\}$$

Because the process n is stationary, this reduces to

$$S_v(\omega) = \sum_k |h_k|^2 S_n(\omega - k\omega_0)$$

Because $S_n(\omega)$ 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 <u>Figure 1-61</u> on page 131, 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 1-62. 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 1-62 Time-Variation Introduces Frequency Correlation
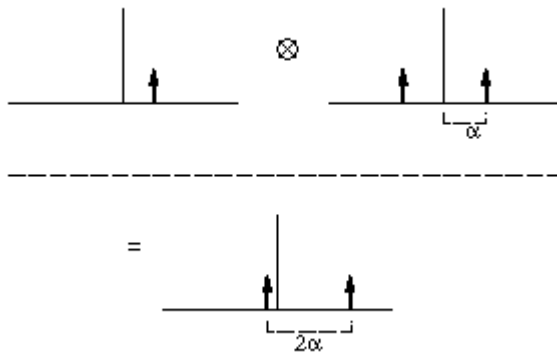
Physically, what happens is that to form a waveform with a defined *shape* in time, the different frequency components of the signal must add in a coherent, or correlated fashion. In a process like thermal noise, the Fourier coefficients at different frequencies have phase that is randomly distributed 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, that is

$$R_n^{\xi}(p, p + q) = R_n^{\xi}(q)$$

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 shows 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 in general is 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.

Oscillator Noise Analysis

In RF systems, local oscillator phase noise can limit the final system performance. Spectre$^{®}$ circuit simulator RF analysis (SpectreRF) 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 simulator.

"Phase Noise Primer" on page 141 discusses how phase noise occurs and provides a simple illustrative example.

"Models for Phase Noise" on page 145 contains mathematical details about how the SpectreRF 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 SpectreRF 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.

<u>"Calculating Phase Noise"</u> on page 155 provides some suggestions for successful and efficient analysis of oscillators and discusses the limitations of the simulator.

<u>"Troubleshooting Phase Noise Calculations"</u> on page 157 explains troubleshooting methods for difficult simulations.

<u>"Frequently Asked Questions"</u> on page 162 answers some commonly asked questions about phase noise and the SpectreRF simulator.

<u>"Further Reading"</u> on page 167 and <u>"References"</u> on page 168 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, see *Spectre Circuit Simulator RF Analysis Theory.*

Phase Noise Primer

Consider the simple resonant circuit with a feedback amplifier shown in Figure 1-63, 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 1-63 A Simple Resonant Oscillator
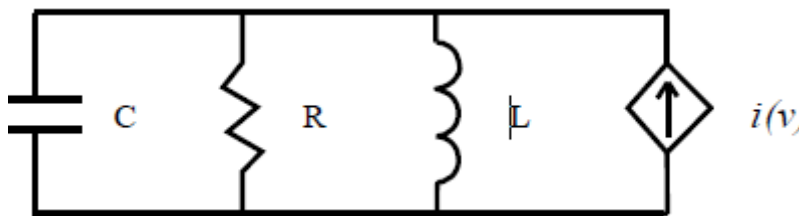
A simple model for the nonlinear transconductance is a cubic polynomial. We hypothesize a nonlinear resistor with a current-voltage relation given by

$$i(v) = -\left(\frac{v}{R}\right)(1 - \alpha v^2)$$

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.

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 1-64 shows the total time-averaged noise in the voltage variable.

Figure 1-64 Noise in a Simple Van der Pol System
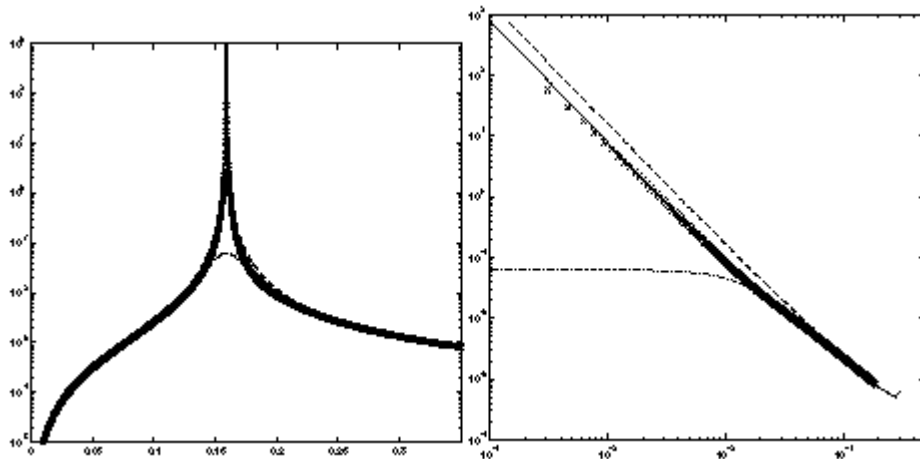
By plotting *Power Spectral Density* against *Normalized Noise Frequency Offset* for a Q = 5 system, Figure 1-64 shows noise in a simple Van der Pol system.

The left half of Figure 1-64 shows noise as a function of absolute frequency.

The right half of Figure 1-64 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 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 <u>Models for Phase Noise</u>. If you are not interested in these mathematical details, you might skip ahead to <u>"Calculating Phase Noise"</u> on page 155.

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

$$S(\omega) = \frac{4k_B T}{R}$$

Where k_B is Boltzman's constant. 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

$$H(\omega) = \frac{v_0(\omega)}{i_s(\omega)}$$

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 $\omega' \ll \omega_0$.

$$S_v(\omega') = \frac{4k_B TR}{1 + 4(\omega'/\omega_0)^2 Q^2}$$

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

$$S_v(\omega') = \frac{k_B TR}{(\omega'/\omega_0)^2 Q^2}$$

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 151). 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 is [3]

$$S_v(\omega) = \sum_k \left|H_k(\omega)\right|^2 S_\xi(\omega + k\omega_0)$$

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 $|H_k(\omega)|^2$. 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 Periodic Time Varying Linear (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 ω - 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_-

$$
\begin{bmatrix} 1-(1+\omega')^2+\left(\dfrac{i}{Q}\right)(1+\omega') & -\left(\dfrac{i}{Q}\right)(-1+\omega') \\[2ex] -\left(\dfrac{i}{Q}\right)(1+\omega') & 1-(-1+\omega')^2+\left(\dfrac{i}{Q}\right)(-1+\omega') \end{bmatrix} \begin{bmatrix} i_+ \\[1ex] i_- \end{bmatrix} = \begin{bmatrix} c_+ \\[1ex] 0 \end{bmatrix}
$$

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',

$$
H_0^2 \cong H_{-2}^2 \cong \frac{R^2}{16Q^2\left(\dfrac{\omega'}{\omega_0}\right)^2}
$$

For a general Van der Pol circuit with a parallel resistor R that generates white current noise, $\xi(t)$, with $S_\xi(\omega) = 4\,k_B\,T/R$,

$$
S_v(\omega') = \frac{k_B TR}{2\left(\dfrac{\omega'}{\omega}\right)^2 Q^2}
$$

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 simulator computes the total noise generated by the circuit; see "Details of the SpectreRF Calculation" on page 152).

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 SpectreRF 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 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 simulator computes noise in oscillators. Understanding the material in this section can help you troubleshoot and understand difficult oscillator problems.

The analysis the SpectreRF simulator performs is similar to the simple analysis in the section "Linear Time-Varying (LTV) Models" on page 146. During analysis, the SpectreRF simulator

1. First finds the periodic steady state of the oscillator using the PSS analysis.

2. Then linearizes around this trajectory.

3. 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 SpectreRF 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]. After 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 $vs(t)$ 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 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 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. After 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 1-64 on page 143, 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 158 for a discussion.

Table 1-3 recommends simulator options for various classes of circuits.

Table 1-3 Recommended SpectreRF Parameter Values

Circuit	errpreset	reltol	vabstol	iabstol
Easy	moderate	$1.0e^{-4}$	default	default

Table 1-3 Recommended SpectreRF Parameter Values

Circuit	errpreset	reltol	vabstol	iabstol
Hard-I	conservative	$1.0e^{-5}$	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. This is the default setting when you set `errpreset=moderate`.

- *Hard-I* circuits are most high-Q 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 and it is the default when you set `errpreset=conservative`.

- *Hard-II* and *Hard-III* circuits include a few particularly difficult circuits.

  Usually these options are used only in a convergence study 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 158). 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 158).

The parameters in Table A-1 are used in error control at two places.

At the local truncation error (LTE) at each transients integration. The LTE control formula is

$$v_n(t) - v_{n, pred}(t) < lteratio \times [reltol \times v_{n, max} + vabstol]$$

- To control the periodicity error over the period. The periodicity error control formula is

$$v_n(0) - v_n(T) < lteratio \times steadyratio \times [reltol \times v_{n, max} + vabstol]$$

The default value for `vabstol` is $1.0e^{-6}$, The default value for `iabstol` is $1.0e^{-12}$ which is accurate enough for most cases. Tighten `iabstol` when necessary. Refer to section 4.3 in [17] for detailed discussion about transient integration.

To speed up transients integration during the `tstab` stage, the default values at the `tstab` stage are `reltol=1.0e-3`, `lteratio=3.5`, `relref=sigglobal`, `maxstep=T/25`. `method=traponly`. After the `tstab` stage, those parameters are set back according to `errpreset`.

For circuits, such as extremely high Q oscillators, that need very high accuracy, you can gain further accuracy by turning on the highorder refinement `highorder=yes` and `errpreset=conservative`. This runs multi-interval Chebyshev PSS refinement after the Shooting phase of the PSS analysis.

An effective and fast method to start the oscillator is to run the new autonomous envelope analysis and to save the simulation results. Use the results of the autonomous envelope analysis as the initial condition for the PSS analysis.

A longer `tstab` stage helps with PSS convergence. However a longer `tstab` stage can slow the simulation. Using autonomous envelope analysis to establish `tstab` is considerably faster than using the transient analysis for `tstab`. See *Spectre Circuit Simulator RF Analysis Theory.* for information on the autonomous envelope analysis.

For releases before MMSIM60 USR1, an effective method to start the oscillation is to add a kicker to the circuit. The kicker can be either a voltage or current source. To effectively start the oscillation, the kicker has to be placed at the most sensitive place which usually is close to the oscillating transistor. The kicker can be either a PWL source or a damped sinusoidal source with frequency set to the oscillation frequency. The kicker must die down and remain stable after oscillation is established to avoid affecting the PSS analysis.

Troubleshooting Phase Noise Calculations

The SpectreRF 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 152 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 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 sub-harmonic responses and cannot be properly analyzed with the SpectreRF simulator. As another example, Colpitts oscillators, properly constructed, can be made to exhibit chaotic as well as sub-harmonic 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 bias circuitry can create these effects.

The SpectreRF simulator cannot simulate these circuits because simulation of an autonomous circuit with sub-harmonic 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 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 TΩ resistor. Changing to a 2 TΩ 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 <u>"Frequently Asked Questions"</u> on page 162 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 SpectreRF 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 simulator prints a warning (see "Message III" on page 161). 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.

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 SpectreRF simulator correctly identifies this situation, it prints a warning (see "Message III" on page 161). The solution is usually to simulate using the next higher accuracy step (see Table 1-3 on page 155). 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 SpectreRF simulator is not designed to deal with them; see "Amplitude Noise and Phase Noise in the Linear Model" on page 151 and "Frequently Asked Questions" on page 162.

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 152.

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 1-3 on page 155) 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 1-3 on page 155). 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 1-3 on page 155) 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 is very difficult for PSS to converge. Similarly, the results of the small-signal analyses are probably not 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 158).

Frequently Asked Questions

The following questions are similar to those commonly asked about oscillator noise analysis with the SpectreRF 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 152 and the detailed discussion of the Van der Pol oscillator "Linear Time-Varying (LTV) Models" on page 146.

I have a circuit that contains an oscillator. Can I simulate the oscillator separately and use the phase noise SpectreRF 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 ΔS_v in the noise is proportional to [1]S / [1]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 `tstab` values (See "The tstab Parameter" on page 161). 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. (Incommensurate frequencies are those for which there is no period that is an integer multiple of the period of each frequency.) 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, because $t = -\infty$, 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 145, and, in particular, "Amplitude Noise and Phase Noise in the Linear Model" on page 151. Because the normal NOISE analysis is satisfactory for these circuits and also much faster, it is unlikely that Pnoise is 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 151, 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.

[17] K. Kundert, "The Designer's Guide to Spice & Spectre," *Kluwer Academic Publishers*, 1995.

Measuring AM, PM and FM Conversion

Derivation

Consider a sinusoid that is simultaneously both amplitude and phase modulated as in Equation 1-38.

(1-38) $v_m(t) = A_c(1 + \alpha(t))\cos(\omega_c t + \phi_c + \phi(t))$

In Equation 1-38, 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 1-39. See [ziemer76].

(1-39) $v_m(t) = A_c(1 + \alpha(t))[\cos(\omega_c t + \phi_c) - \phi(t)\sin(\omega_c t + \phi_c)]$

Converting to complex exponentials gives Equation 1-40.

(1-40) $v_m(t) = \dfrac{A_c}{2}(1 + \alpha(t))$
$$\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + j\phi(t)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)} \right) \right]$$

Letting both the amplitude and phase modulation be complex exponentials with the same frequency, ω_m, gives Equations 1-41, 1-42, 1-43, 1-44 and 1-45.

(1-41) $\quad \alpha(t) = A e^{j\omega_m t}$

(1-42) $\quad \phi(t) = \Phi \dfrac{e^{j\omega_m t} + e^{-j\omega_m t}}{2}$

Where

(1-43) $\quad A = A_{A} e^{j\phi_A t}$

(1-44) $\quad \Phi = A_{\Phi} e^{j\phi_\Phi t}$

(1-45)
$$v_m(t) = \frac{A_c}{2}\left(1 + A e^{j\omega_m t}\right)$$
$$\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + \right.$$
$$\left. j\frac{1}{2}\Phi\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)\left(e^{j(\omega_c t + \phi_c)} - e^{-j(\omega_c t + \phi_c)}\right)\right]$$

Assuming that both A and Φ are small and neglecting cross modulation terms gives Equation 1-46.

(1-46)

$$v_m(t) = \frac{A_c}{2}\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)} + \right.$$

$$Ae^{j\omega_m t}e^{j(\omega_c t + \phi_c)} + Ae^{j\omega_m t}e^{-j(\omega_c t + \phi_c)} +$$

$$j\frac{\Phi}{2}\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)e^{j(\omega_c t + \phi_c)} -$$

$$\left. j\frac{\Phi}{2}\left(e^{j\omega_m t} + e^{-j\omega_m t}\right)e^{-j(\omega_c t + \phi_c)}\right]$$

Simplifying gives Equation 1-47.

(1-47)

$$v_m(t) = \frac{A_c}{2}\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)}\right.$$

$$+ Ae^{j((\omega_m + \omega_c)t + \phi_c)} + Ae^{j((\omega_m - \omega_c)t - \phi_c)}$$

$$+ \frac{1}{2}j\Phi\left[e^{j((\omega_m + \omega_c)t + \phi_c)} - e^{j((\omega_m - \omega_c)t - \phi_c)}\right.$$

$$\left.\left. + e^{j((-\omega_m + \omega_c)t + \phi_c)} - e^{j((-\omega_m - \omega_c)t - \phi_c)}\right]\right]$$

In Equation 1-47,

■ 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

$$\frac{1}{2}\left[j\Phi e^{j((\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((\omega_m - \omega_c)\,t - \phi_c)}\right.$$

$$\left. + j\Phi e^{j((-\omega_m + \omega_c)\,t + \phi_c)} - j\Phi e^{j((-\omega_m - \omega_c)\,t - \phi_c)}\right]$$

Because the left-side term $v_m(t)$ represents a real-time signal, only the real parts of the complex terms are of interest. Dropping the imaginary parts and rearranging Equation 1-47 produces Equation 1-48,

(1-48)
$$v_m(t) = \frac{A_c}{2}\left[e^{j(\omega_c t + \phi_c)} + e^{-j(\omega_c t + \phi_c)}\right.$$

$$+ A e^{j(\omega_m - \omega_c)\,t} e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)\,t} e^{-j\phi_c}$$

$$\left. + A e^{j(\omega_m + \omega_c)\,t} e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)\,t} e^{j\phi_c}\right]$$

When you ignore the negative ω_m term in Equation 1-48, you get

■ The LSB (lower sidebands) terms are

$$A e^{j(\omega_m - \omega_c)\,t} e^{-j\phi_c} - j\Phi e^{j(\omega_m - \omega_c)\,t} e^{-j\phi_c}$$

■ The USB (upper sidebands) terms are

$$A e^{j(\omega_m + \omega_c)\,t} e^{j\phi_c} + j\Phi e^{j(\omega_m + \omega_c)\,t} e^{j\phi_c}$$

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 1-49 and 1-50.

(1-49) $$l(t) = L e^{j(\omega_m - \omega_c)\,t}$$

(1-50) $\quad u(t) = Ue^{j(\omega_m + \omega_c)\,t}$

Where

$$L = A_L e^{j\phi_L}$$

$$U = A_U e^{j\phi_U}$$

Matching common frequency terms between Equations 1-48, 1-49, and 1-50 gives Equations 1-51, 1-52, 1-53 and 1-54.

(1-51) $\quad L = \dfrac{A_c}{2}\left(Ae^{-j\phi_c} - j\Phi e^{-j\phi_c}\right)$

(1-52) $\quad U = \dfrac{A_c}{2}\left(Ae^{j\phi_c} + j\Phi e^{j\phi_c}\right)$

(1-53) $\quad \dfrac{2}{A_c}Le^{j\phi_c} = A - j\Phi$

(1-54) $\quad \dfrac{2}{A_c}Ue^{-j\phi_c} = A + j\Phi$

Solving for the modulation coefficients gives Equations 1-55 and 1-56.

(1-55) $\quad A = \dfrac{1}{A_c}\left(Le^{j\phi_c} + Ue^{-j\phi_c}\right)$

(1-56) $\quad \Phi = \dfrac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)$

Thus, Equation 1-55 gives the transfer function for amplitude modulation and Equation 1-56 gives the transfer function for phase modulation.

Positive Frequencies

Notice that L is defined in Equation 1-49 on page 172 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 the Spectre$^{\circledR}$ circuit simulator RF analysis (SpectreRF) 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 1-57,

(1-57) $\quad \tilde{l}(t) = \tilde{L}e^{j(\omega_c - \omega_m)t}$

Because the signals are real, L is a complex conjugate of

$\tilde{L}$

And the reverse is also true, as in Equation 1-58,

(1-58) $\quad L = \tilde{L}^{*}$

Equations 1-59 and 1-60 are produced by rewriting Equation 1-55 on page 173 and Equation 1-56 on page 174 in terms of

$\tilde{L}$

(1-59) $\quad A = \dfrac{1}{A_c}\left(\tilde{L}^* e^{j\phi_c} + Ue^{-j\phi_c}\right)$

(1-60) $\quad \Phi = \dfrac{j}{A_c}\left(\tilde{L}^* e^{j\phi_c} - Ue^{-j\phi_c}\right)$

FM Modulation

For FM modulation, the phase modulation $\phi(t)$ becomes the integral of the FM modulation signal, $\omega(t)$ as shown in Equation 1-61.

(1-61) $\quad v_m(t) = A_c \cos(\omega_c t + \phi(t))$

Where

(1-62) $\quad \phi(t) = \int \omega(t)dt$

Recall from Equation 1-42 on page 170 and Equation 1-56 on page 174 that

(1-63) $\quad \phi(t) = \dfrac{j}{A_c}\left(Le^{j\phi_c} - Ue^{-j\phi_c}\right)e^{j\omega_m t}$

Combining Equation 1-62 and Equation 1-63 and the differentiating both sides results in Equations 1-64, 1-65, and 1-66.

(1-64) $\quad \omega(t) = \dfrac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)e^{j\omega_m t}$

(1-65) $\quad \Omega = A_\Omega e^{j\phi_\Omega} = \dfrac{\omega_m}{A_c}\left(Ue^{-j\phi_c} - Le^{j\phi_c}\right)$

Or

(1-66) $\quad \Omega = j\omega_m \Phi$

Simulation

The test circuit, represented by the two netlists shown in Example on page 176 and Example on page 177, 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 1-59 on page 175 and Equation 1-60 on page 175.

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 1-66, has the filename
`modulators.va`.

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 1-4 shows results for the output of the AM modulator with $v_{LO} = cos(\omega_c t)$.

Table 1-5 shows results for the output of the PM modulator with $v_{LO} = cos(\omega_c t)$.

Table 1-4 Results for the AM Modulator Output

pacphase	L	U	A	Φ
0	1/2	1/2	1	0
45	$\dfrac{1+j}{2\sqrt{2}}$	$\dfrac{1+j}{2\sqrt{2}}$	$\dfrac{1+j}{2}$	0
90	$j/2$	$j/2$	j	0
180	−1/2	−1/2	−1	0

Table 1-5 Results for the PM Modulator Output

pacphase	L	U	A	Φ
0	−1/2	1/2	0	1
45	$\dfrac{1-j}{2\sqrt{2}}$	$\dfrac{j-1}{2\sqrt{2}}$	0	$\dfrac{1+j}{\sqrt{2}}$
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 1-6 shows results for the output of the AM modulator with $v_{LO} = sin(\omega_c t)$.

■   Table <u>1-7</u> shows results for the output of the PM modulator with $v_{LO} = sin(\omega_c t)$.

Table 1-6 Results for the AM Modulator Output

pacphase	L	U	A	Φ
0	$j/2$	$-1/2$	1	0
45	$\dfrac{j-1}{2\sqrt{2}}$	$\dfrac{1-j}{2\sqrt{2}}$	$\dfrac{1+j}{\sqrt{2}}$	0
90	$-1/2$	$1/2$	j	0
180	$-j/2$	$j/2$	-1	0

Table 1-7 Results for the PM Modulator Output

pacphase	L	U	A	Φ
0	$1/2$	$1/2$	0	1
45	$\dfrac{1+j}{2\sqrt{2}}$	$\dfrac{1+j}{2\sqrt{2}}$	0	$\dfrac{1+j}{\sqrt{2}}$
90	$j/2$	$j/2$	0	j
180	$-1/2$	$-1/2$	0	-1

Finally, Table <u>1-8</u> 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 1-8 Results for the FM Modulator Output

pacphase	L	U	Ω
0	$-1/2$	$1/2$	1
45	$-\dfrac{1+j}{2\sqrt{2}}$	$\dfrac{1+j}{2\sqrt{2}}$	$\dfrac{1+j}{\sqrt{2}}$
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.

[1]   J. Roychowdhury, D. Long and P Feldmann. *Cyclostationary Noise Analysis of Large RF Circuits with Multitone Excitations. JSSC Vol. 33 No.3* 1998

2

The Spectre RF Analyses

Spectre® circuit simulator RF analysis (Spectre RF) 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 Spectre RF analyses described in this section are

■ PSS (large-signal analysis). See "Periodic Steady-State Analysis (PSS)" on page 183.

■ PAC (small-signal analysis). See "Periodic AC Analysis (PAC)" on page 198.

■ PSP (small-signal analysis). See "Periodic S-Parameter Analysis (PSP)" on page 205.

■ PXF (small-signal analysis). See "Periodic Transfer Function Analysis (PXF)" on page 210.

■ Pnoise (small-signal analysis). See "Periodic Noise Analysis (Pnoise)" on page 216.

■ PSTB (periodic stability analysis). See "Periodic Stability Analysis (PSTB)" on page 226.

■ QPSS (large-signal analysis). See "Quasi-Periodic Steady-State Analysis (QPSS)" on page 229.

■ QPnoise (small-signal analysis). See "Quasi-Periodic Noise Analysis (QPnoise)" on page 236.

■ QPAC (small-signal analysis). See "Quasi-Periodic AC Analysis (QPAC)" on page 242.

■ QPSP (small-signal analysis). See "Quasi-Periodic S-Parameter Analysis (QPSP)" on page 245.

■ QPXF (small-signal analysis). See "Quasi-Periodic Transfer Function Analysis (QPXF)" on page 253.

■ ENVLP (envelope analysis). See "Envelope Analysis (ENVLP)" on page 276.

■ HB (large-signal analysis). See "Harmonic Balance Steady State Analysis (HB)" on page 257.

- HBAC (small-signal analysis). See "Harmonic Balance AC Analysis (HBAC)" on page 264.

- HBnoise (small-signal analysis). See "Harmonic Balance Noise Analysis (HBnoise)" on page 267.

- HB S-Parameter Analysis (hbsp). See "HB S-Parameter Analysis (hbsp)" on page 272

- Envelope Analysis (ENVLP). See "Envelope Analysis (ENVLP)" on page 276

Periodic Steady-State Analysis (PSS)

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 the required starting point for the periodic time-varying small-signal analyses: PAC, PSP, PXF, and Pnoise.

The Shooting Method

Spectre RF simulation traditionally uses a technique called the shooting method to implement PSS analysis. This method is an iterative, time-domain method which starts with a guess or estimate of the initial condition and ultimately finds an initial condition that directly results in a steady-state solution.

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.

Cadence's Fourier integral method, a new approach to Fourier analysis, makes PSS analysis using the shooting engine 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`, Spectre RF automatically performs a high-order refinement after the shooting method. Spectre RF uses the Multi-Interval Chebyshev polynomial spectral algorithm (MIC) to refine the simulation results. When you set `highorder` to `no`, MIC is turned off. However, when you set `highorder` to `yes`, Spectre RF tries harder to converge. In a case where MIC fails to converge, Spectre RF falls back to the original PSS solution. For more information, see "The High-Order and Finite Difference Refinement Parameters" on page 190 and "The errpreset Parameter in PSS Analysis" on page 191.

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 190.

Parameters for PSS Analysis

For more information on PSS analysis parameters, refer to the *Periodic Steady-State Analysis (pss)* section in the Spectre Circuit Simulation Reference manual.

The PSS Algorithm for Driven and Autonomous Circuits

The PSS analysis works with both autonomous and driven circuits.

■ Driven circuits require some time-varying stimulus to generate a time-varying response.

   Some common driven circuits include amplifiers, filters, mixers, and so on.

■ 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.

See "Autonomous PSS Analysis" on page 187 for additional information on the algorithm for PSS analysis of autonomous circuits.

> *Important*
>
> The `errpreset` parameter works differently for autonomous and driven circuits. For detailed information, see "The errpreset Parameter in PSS Analysis" on page 191.

Driven PSS Analysis

A PSS analysis using the shooting method 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.

The Initial Transient Phase of PSS Analysis

The initial transient analysis provides a flexible mechanism to direct the circuit to a particular steady-state solution you are interested in and to avoid undesired solutions. Another use of the initial transient simulation is to help convergence by eliminating large but fast decaying

modes that are present in many circuits. For example, in the case of driven circuits, consider the reset signal in Figure 2-1 on page 185.

PSS starts by performing a loose transient analysis for the interval from t_{start} to t_{stop} where t_{stop} is $t_{onset} + t_{stab} + period$. This initial loose transient analysis disregards your `errpreset` setting and performs a transient analyses with `errpreset=liberal`. If the initial transient results are relevant, you can output them by setting `saveinit` to `yes`. The steady-state results are always computed for the specified period, from t_{init} to t_{stop}. By default, t_{start} and t_{stab} are set to zero, while t_{init}, t_{onset} and t_{stop} are always automatically generated and your `errpreset` settings are used.

Figure 2-1 Initial Transient Analysis and Timing Relationships for PSS Analysis
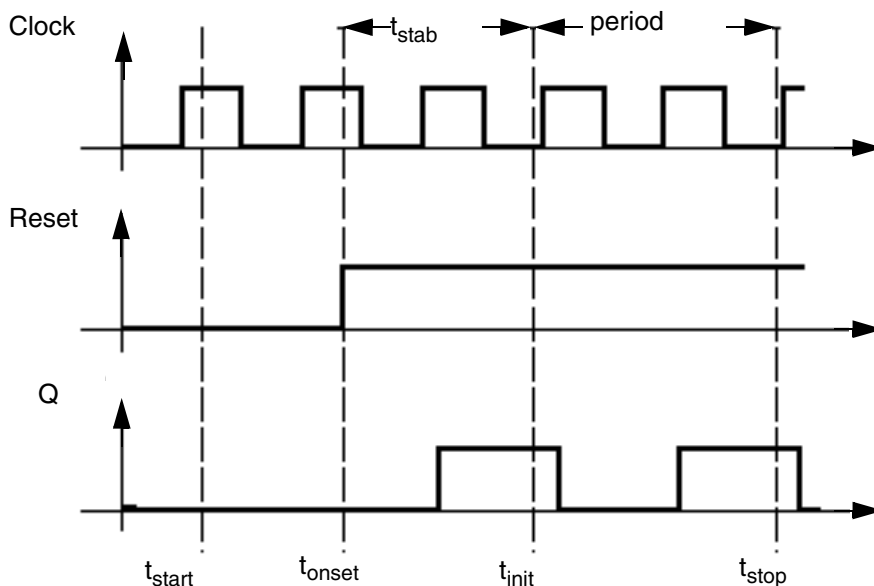
The first interval begins at `tstart`, which is normally 0, and continues through the onset of periodicity t_{onset} for the independent sources. The onset of periodicity, which is automatically generated, is the earliest time for which all sources are periodic. The second interval is an optional user 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.

Table 2-1 Timing Intervals for PSS Analysis

t$_{start}$ Start time, `tstart`, for the transient analysis that you specify. This value can be negative or positive and defaults to 0.

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. $t_{stop} = t_{init} + period$.

The Shooting Phase

After 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, if the convergence criteria are not satisfied, generates an improved estimate of the necessary initial condition.

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.

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.

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 Analyses 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 Analyses form.

You can save the initial transient results by setting `saveinit` to `yes`. 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.

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 $-0.1 * t_{stab}$.

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.

Autonomous PSS Analysis

Because autonomous circuits do not have drive signals and you do not know the actual period of oscillation before you run a simulation, you estimate the oscillation period and the PSS analysis computes the precise period along with the periodic solution waveforms. Autonomous circuits, such as oscillators, however, have time-varying responses and generate non-constant waveforms even though the circuits themselves are time-invariant.

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 *Spectre Circuit Simulator and*

Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide for an example.

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 first stage of the transient phase, PSS extracts the fundamental frequency from the set of nodes specified in the PSS statement. During the second stage, the PSS analysis examines all the nets in your design to verify the accuracy of the extracted PSS fundamental frequency. This enhancement improves PSS analysis of circuits with frequency dividers by re-evaluating the PSS fundamental to take into account the frequency division in your circuit.

Increasing the tstab Interval

In some cases when an autonomous PSS analysis does not converge after a few iterations, increasing the `tstab` interval makes convergence faster and easier.

Adjusting the `tstab` interval might improve the shooting interval starting point by moving it such that signals are quiescent or changing slowly. This allows strongly nonlinear circuits to converge faster.

For oscillator circuits, Spectre RF increases the `tstab` interval when:

■ The current Iteration is close to the maximum number of iterations, but the related Norm is still large.

```
(Iter >= 0.8 * MaxIters && Norm > 1000.0
(Iter >= 0.9 * MaxIters && Norm > 100.0)
(Iter >= 0.95* MaxIters && Norm > 10.0)
(Iter >= MaxIters    && Norm > 1.0)
```

■ The PssPeriod has shrunk to zero

   ```
   (PssPeriod < UsrPssPeriod/1.0e3)
   ```

■ The initial transient analysis has failed.

When any of these conditions occur, Spectre RF increases the tstab interval as follows.

```
4.0*PeriodEstimate + PeriodEstimate*DynamicTstabCounter/6.0
```

where `DynamicTstabCounter` is an integer from 1 to 6.

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.

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 191.

/ *Important*

   The `errpreset` parameter works differently for autonomous and driven circuits. For detailed information, see "The errpreset Parameter in PSS Analysis" on page 191.

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` parameter 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. This is the default.

■   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 <u>Table 2-2</u> on page 191 lists the `relref` parameter options.

Table 2-2 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.
`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

⚠ *Important*

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

The `errpreset` parameter works differently for autonomous and driven circuits. For more information see,

■ <u>"Using the errpreset Parameter With Driven Circuits"</u> on page 191.

■ <u>"Using the errpreset Parameter With Autonomous Circuits"</u> on page 193.

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 2-3 shows the effect of the `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the values of the other parameters used with driven circuits.

Table 2-3 Default Values and Noise Floor for `errpreset` in Driven Circuits

errpreset	reltol	relref	method	Iteratio	steadyratio	maxstep
liberal	$1e^{-3}$	sigglobal	traponly	3.5	0.001	period/50
moderate	$1e^{-3}$	alllocal	gear2only	3.5	0.001	period/200
conservative[1]	$1e^{-4}$	alllocal	gear2only	*	0.01	period/200

1. Iteratio=10.0 for conservative `errpreset`. Only if user-specified reltol <= 1e-4 * 10.0/3.5, Iteratio is set to 3.5.

Several parameters determine the accuracy of the PSS analysis. `Reltol` and `abstol` 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. The integration error, or the errors in the computation of the circuit dynamics (such as time constants) relative to `reltol` and `abstol` are set by the `lteratio` parameter.

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 with the `options` statement. The only way to *relax* the `reltol` value is to change the `errpreset` setting.

For a weakly nonlinear circuit, the estimated numerical noise floor is -70 dB for `liberal`, -90 dB for `moderate`, and -120 dB for `conservative errpreset` settings. For a linear circuit, the noise floor is even lower. Multi-interval Chebyshev (MIC) is activated when you explicitly set `highorder=yes`, which drops the numerical noise floor by at least 30 dB. MIC falls back to the original method if it encounters difficulty converging. You can tighten the `psaratio` to further drop the numerical noise floor. Spectre RF sets the value of `maxstep` so that it cannot be larger than the value given in Table 2-3. Except for the `reltol` and `maxstep` parameters, `errpreset` does not change the value of any parameters you have explicitly set. The actual values used for the PSS analysis are given in the log file. If `errpreset` is not specified in the netlist, `liberal` settings are used.

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 2-4 shows the effect of the `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the values of parameters used with autonomous circuits.

Table 2-4 Default Values and maxstep for `errpreset` 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	0.01	period/200
conservative[1]	$1e^{-5}$	alllocal	gear2only	*	0.1	period/400

1. * : Iteratio=10.0 for conservative `errpreset` by default. Only if user-specified reltol <= 1e-5*10.0/3.5, Iteratio is set to 3.5..

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 *increase* the `reltol` value is to change the `errpreset` setting. Spectre RF sets the `maxstep` parameter value so that it is no larger than the value given in Table 2-4. 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 PSS analysis are given in the log file.

The value of `reltol` can be decreased from the default value in the options statement. The only way to increase `reltol` is to relax `errpreset`. Spectre RF sets the value of `maxstep` so that it cannot be larger than the value given in Table 2-4. Except for `reltol` and `maxstep`, `errpreset` does not change the value of any parameters you have explicitly set. The actual values used for the PSS analysis are given in the log file. If `errpreset` is not specified in the netlist, `liberal` settings will be used. Multi-interval Chebyshev (MIC) is activated when you explicitly set `highorder=yes`, which will drop the numerical noise floor by at least 30 dB. MIC falls back to the original method if it encounters difficulty converging. You can tighten `psaratio` to further drop the numerical noise floor.

Other Parameters

A long stabilization (by specifying a large tstab) can help with the PSS convergence. However it can slow down simulation. By default, in the stabilization stage, reltol=1e-3;

maxstep=period/25; relref=sigglobal; and method=traponly. They are overwritten when maxstep, relref, or tstabmethod are specified explicitly in pss statement, or reltol is specified explicitly in options statement.

If the circuit you are simulating can have infinitely fast transitions (for example, a circuit that contains nodes with no capacitance), Spectre RF might have convergence problems. To avoid this, you must prevent the circuit from responding instantaneously. You can accomplish this by setting `cmin`, the minimum capacitance to ground at each node, to a physically reasonable nonzero value. This often significantly improves convergence.

You may specify the initial condition for the transient analysis by using the `ic` statement or the `ic` parameter on the capacitors and inductors. If you do not specify the initial condition, the DC solution is used as the initial condition. The `ic` parameter on the transient analysis controls the interaction of various methods of setting the initial conditions. The effects of individual settings are

`ic=dc`	Any initial condition specifiers are ignored, and the DC solution is used.
`ic=node`	The `ic` statements are used, and the `ic` parameter on the capacitors and inductors are ignored.
`ic=dev`	The `ic` parameters on the capacitors and inductors are used, and the `ic` statements are ignored.
`ic=all`	Both the `ic` statements and the `ic` parameters are used, and the `ic` parameters override the `ic` statements.

If you specify an initial condition file with the `readic` parameter, initial conditions from the file are used, and any `ic` statements are ignored.

After you specify the initial conditions, Spectre RF computes the actual initial state of the circuit by performing a DC analysis. During this analysis, Spectre RF forces the initial conditions on nodes by using a voltage source in series with a resistor whose resistance is `rforce` (see `options`).

With the `ic` statement it is possible to specify an inconsistent initial condition (one that cannot be sustained by the reactive elements). Examples of inconsistent initial conditions include setting the voltage on a node with no path of capacitors to ground or setting the current through a branch that is not an inductor. If you initialize Spectre RF inconsistently, its solution jumps; that is, it changes instantly at the beginning of the simulation interval. You should avoid such changes if possible because Spectre RF can have convergence problems while trying to make the jump.

You can skip the DC analysis entirely by using the parameter `skipdc`. If the DC analysis is skipped, the initial solution will be either trivial, or given in the file you specified by the `readic` parameter, or, if the `readic` parameter is not given, the values specified on the ic statements. Device-based initial conditions are not used for `skipdc`. Nodes that you do not specify with the `ic` file or `ic` statements will start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution; it usually takes longer to follow the initial transient spikes that occur when the DC analysis is skipped than it takes to find the real DC solution. The `skipdc` parameter might also cause convergence problems in the transient analysis.

The possible settings of parameter `skipdc` and their meanings are

`skipdc=no`	Initial solution is calculated using the normal DC analysis (default).
`skipdc=yes`	Initial solution is given in the file specified by the `readic` parameter or the values specified on the ic statements.
`skipdc=sigrampup`	Independent source values start at 0 and ramp up to their initial values in the first phase of the simulation. The waveform production in the time-varying independent source is enabled after the rampup phase.

- ■ The rampup simulation is from `tstart` to time=0 s, and the main simulation is from time=0 s to `tstab`.

- ■ If the `tstart` parameter is not specified, the default `tstart` time is set to -0.1*`tstab`.

Nodesets help find the DC or initial transient solution. You can supply them in the circuit description file with nodeset statements, or in a separate file using the `readns` parameter. When nodesets are given, Spectre RF computes an initial guess of the solution by performing a DC analysis while forcing the specified values onto nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre RF then removes these voltage sources and resistors and computes the true solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator towards computing the desired one. Second, they are a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or dramatically speed convergence.

When you simulate the same circuit many times, we suggest that you use both the `write` and `readns` parameters and give the same file name to both parameters. The DC analysis then converges quickly even if the circuit has changed somewhat since the last simulation, and the nodeset file is automatically updated.

Nodesets and initial conditions have similar implementation but produce different effects. Initial conditions actually define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre RF forms and solves a set of differential equations. However, differential equations have an infinite number of solutions, and a complete set of initial conditions must be specified in order to identify the desired solution. Any initial conditions you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator towards finding the solution closest to the nodeset values.

The `method` parameter specifies the integration method. The possible settings and their meanings are

`method=euler`	Backward-Euler is used exclusively.
`method=traponly`	Trapezoidal rule is used almost exclusively.
`method=trap`	Backward-Euler and the trapezoidal rule are used.
`method=gear2only`	Gears second-order backward-difference method is used almost exclusively.
`method=gear2`	Backward-Euler and second-order Gear are used.

The trapezoidal rule is usually the most efficient when you want high accuracy. This method can exhibit point-to-point ringing, but you can control this by tightening the error tolerances. For this reason, though, if you choose very loose tolerances to get a quick answer, either backward-Euler or second-order Gear will probably give better results than the trapezoidal rule. Second-order Gear and backward-Euler can make systems appear more stable than they really are. This effect is less pronounced with second-order Gear or when you request high accuracy.

Spectre RF provides two methods for reducing the number of output data points saved: `strobing`, based on the simulation time, and `skipping` time points, which saves only every Nth point.

The parameters `strobeperiod` and `strobedelay` control the strobing method. `strobeperiod` sets the interval between points that you want to save, and `strobedelay` sets the offset within the period relative to `skipstart`. The simulator forces a time step on each point to be saved, so the data is computed, not interpolated.

The skipping method is controlled by `skipcount`. If this is set to N, then only every Nth point is saved.

The parameters `skipstart` and `skipstop` apply to both data reduction methods. Before `skipstart` and after `skipstop`, Spectre RF saves all computed data.

The default value for `compression` is no. The output file stores data for every signal at every time point for which Spectre RF calculates a solution. Spectre RF saves the x axis data only once, because every signal has the same x value. If `compression=yes`, Spectre RF writes data to the output file only if the signal value changes by at least 2*the convergence criteria. In order to save data for each signal independently, x axis information corresponding to each signal must be saved. If the signals stay at constant values for large periods of the simulation time, setting `compression=yes` results in a smaller output data file. If the signals in your circuit move around a lot, setting `compression=yes` results in a larger output data file.

Periodic AC Analysis (PAC)

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, such as PAC, PSP, PXF, PNoise, can follow a PSS analysis.

PAC analysis is a small-signal analysis like AC analysis, except the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing about a periodically time-varying operating point allows transfer-functions that include frequency translation, whereas simply linearizing about a DC operating point could not because linear time-invariant circuits do not exhibit frequency translation. Also, the frequency of the sinusoidal stimulus is not constrained by the period of the large periodic solution.

Computing the small-signal response 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. The second step is applying the small stimulus to the periodically varying linear representation to compute the small signal response. This is done using the PAC 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 <u>Figure 2-2</u> on page 199.

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 Spectre RF 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

Label	Definition
0	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.

Label	Definition
-1	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.
+1	For up-conversion mixers, the transfer function or sideband is labeled +1.
	For samplers, sidebands far from zero might be used.

In Figure 2-2, 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 2-2 The Small-Signal Response of a Mixer as Computed by PAC Analysis
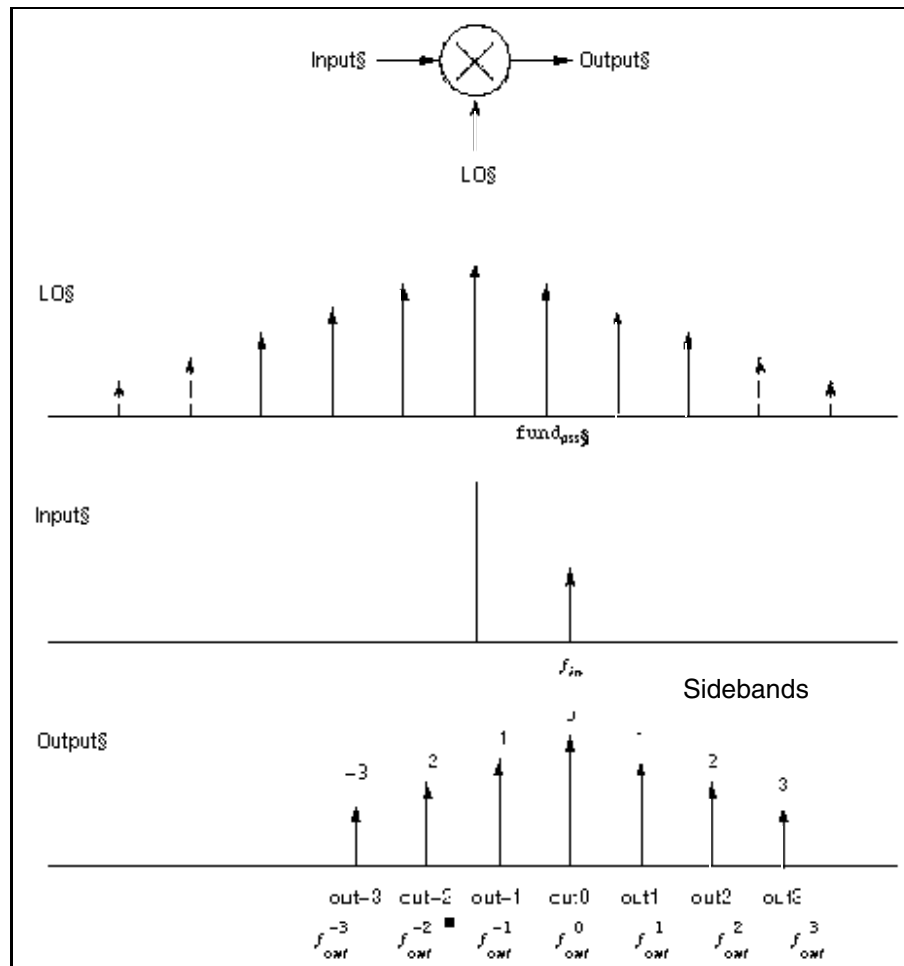
PAC performance is not reduced if the input and LO frequencies are close or equal.
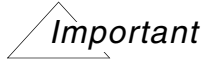
> ⚠️ *Important*
>
> Unlike other analyses in Spectre RF, the PAC analysis can only sweep frequency.

PAC Synopsis

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

For a set of *n* integer numbers representing the sidebands

$$K_1, K_2, \dots K_n$$

the output signal frequency at each sideband is computed as

$$f(out) = f(in) + k_i \times fund(pss)$$

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.

The number of requested sidebands does not substantially change the simulation time. However, the `maxacfreq` of the corresponding PSS analysis should be set to guarantee that $|\max\{f(out)\}|$ is less than `maxacfreq`, otherwise the computed solution might be contaminated by aliasing effects. The PAC simulation is not executed for $|f(in)|$ greater than `maxacfreq`. Diagnostic messages are printed for those extreme cases that indicate how to set `maxacfreq` in the PSS analysis. In the majority of simulations, however, this is not an issue because `maxacfreq` is never allowed to be smaller than 40x the PSS fundamental.

Intermodulation Distortion Computation

A PSS analysis followed by a PAC analysis measures 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 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. After 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 2-3.

In Figure 2-3, 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 2-3, all of the signals are mapped into positive frequencies, which is the most common way of viewing such results.

Figure 2-3 Intermodulation Distortion Measured with PAC Analysis
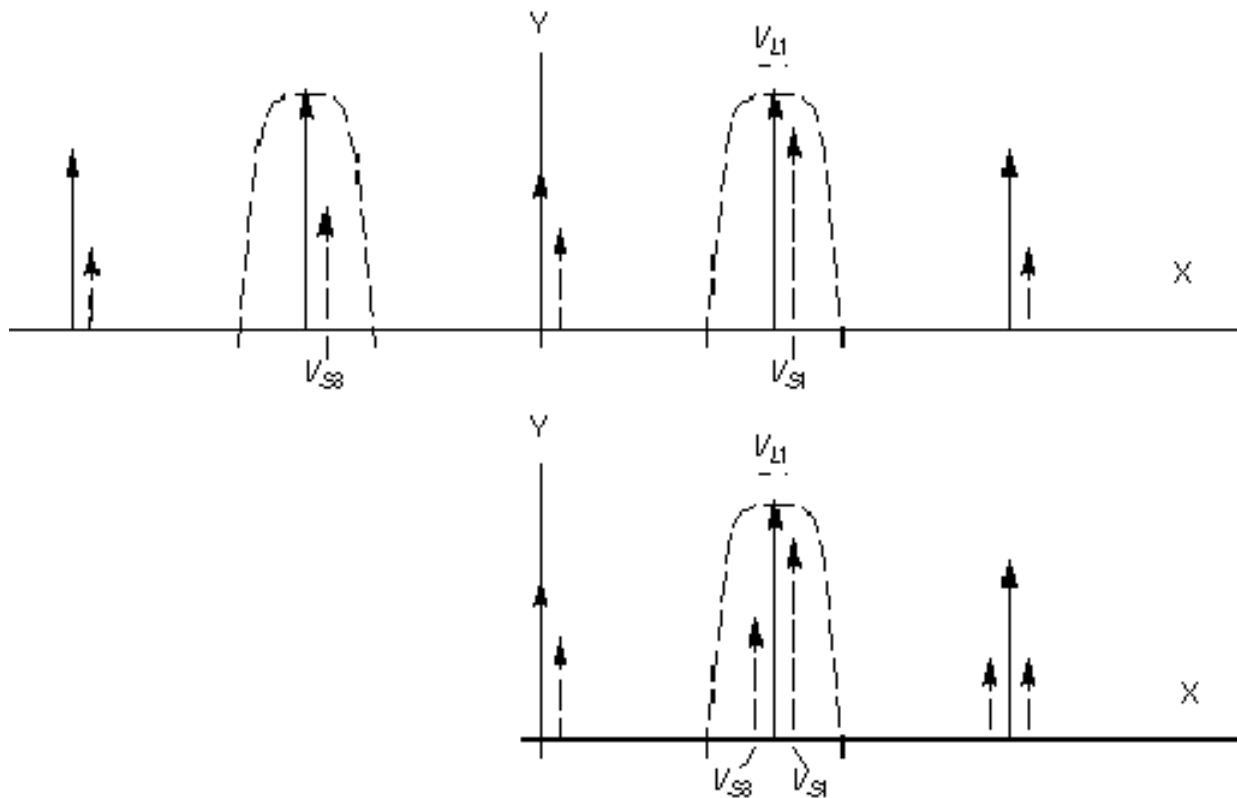
Intermodulation distortion is efficiently measured by applying one large tone (L1), performing a PSS analysis, and then applying the second small tone (S1) with a PAC analysis. In this case, the first tone drives the circuit hard enough to cause distortion and the second tone is used to measure only the intermodulation distortion. After 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.

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

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 the *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide*.

▱ *Important*

> 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.

For the PAC analysis the frequencies of the stimulus and response are usually different. This is an important difference between the PAC analysis and the AC analysis. The `freqaxis` parameter specifies whether the results should be output versus the input frequency, `in`, the output frequency, `out`, or the absolute value of the output frequency, `absout`.

You can make modulated small signal measurements using the Analog Circuit Design Environment (ADE). The `modulated` option for the PAC analysis and other modulated parameters are set in ADE. A PAC analyses with the `modulated` option produces results which might have limited use outside of ADE. The Direct Plot form is configured to analyze modulated small signal measurements and combine several waveforms to measure AM and PM response due to single sideband or modulated stimuli.

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 Hz.

Modulated Small-Signal Analyses

You can make modulated small signal measurements using the Analog Design Environment (ADE). The `modulated` option for PAC and other modulated parameters are set by ADE. PAC analyses with the `modulated` option produce results which might have limited use outside the ADE environment. The ADE Direct Plot form is configured to analyze these results and combine several wave forms to measure AM and PM response due to single sideband or modulated stimuli.

Sampled Small-Signal Analysis

Sampled small signal PXF and PAC analyses use the Analog Design Environment (ADE) environment. The sampled options are set by ADE. The Sampled option produces results which might have limited use outside ADE. Direct Plot is configured to analyze the results and make Sampled measurements due to single sideband or sampled stimuli. A sampled analysis is a small-signal analysis with a use model similar to the sampled (timedomain) PNoise analysis. Specifically, you first create a circuit (schematic or netlist description) and place port or source components to specify the key elements where the transfer function to the output is of interest.

The ability to sample the noise, signal slope and transfer function at a particular time point is a valuable investigative tool for design and verification. For example it will be used in a design of switched-capacitor filters, logic circuits and in the evaluation of the power supply rejection. A Sampled analysis computes the transfer function from different parts of the circuit to the output at a particular time point. The output signal is sampled at the clock rate (large signal period which is used in PSS).

A new choice in the *Specialized Analysis* cyclic menu, *Sampled*, opens the *Sampled* analysis fields in the Choosing Analyses form. Several fields are available to specify the sampling event. First, you select a control signal to observe in search for the triggering event(s). It could be a single or differential voltage signal or a probe, either voltage or current. The threshold value and the crossing direction are the parameters of a triggering timing event. In case of special need, you can specify a delayed measurement from the time of the crossing event.

You can also specify actual time points for sampling the output. The same form will be used and either one or the combination of both approaches is used to do the sampled small signal analysis.

Parameters for PAC Analysis

For information on PAC analysis parameters, refer to the *Periodic AC Analysis (pac)* section in the *Spectre Circuit Simulator Reference* manual

Periodic S-Parameter Analysis (PSP)

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 Spectre RF 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 analysis 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.

Using `sweeptype=absolute` and sweeping the frequency from *900->950MHz* would calculate the same quantities, since *f (abs)=900->950MHz*, and *f (rel) = f (abs) - K1 * fund(pss) = 0->50MHz*, because *K1=9* and *fund(pss) = 100MHz*.

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 substantially.

To ensure 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

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

When the results are output, IRN is named `in`, G is named `gain`, F, NF, Fdsb, NFdsb, Fieee, and NFieee are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively.

To ensure 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

For information on PSP analysis parameters, refer to the *Periodic S-Parameter Analysis (psp)* section in the Spectre Circuit Simulator Reference manual.

Periodic Transfer Function Analysis (PXF)

A conventional transfer function (XF) analysis computes the transfer function from every source in the circuit to a single output. An XF analysis differs from a conventional AC analysis in that the AC analysis computes the response from a single stimulus to every node in the circuit.

The difference between the PXF and PAC analyses 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 2-4.

The PXF analysis directly computes such useful quantities as

■   Conversion efficiency (the transfer function from input to output at a desired frequency)

■   Image and sideband rejection (input to output at an undesired frequency)

■   LO feed-through and power supply rejection (undesired input to output at all frequencies)

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

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 2-4 are those that end up at the output frequency.

Computing transfer functions for 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.

■   Second, using the PXF analysis, small-signal excitations are applied to compute the transfer functions.

Figure 2-4 Mixer Output Signals Shown by PXF Analysis
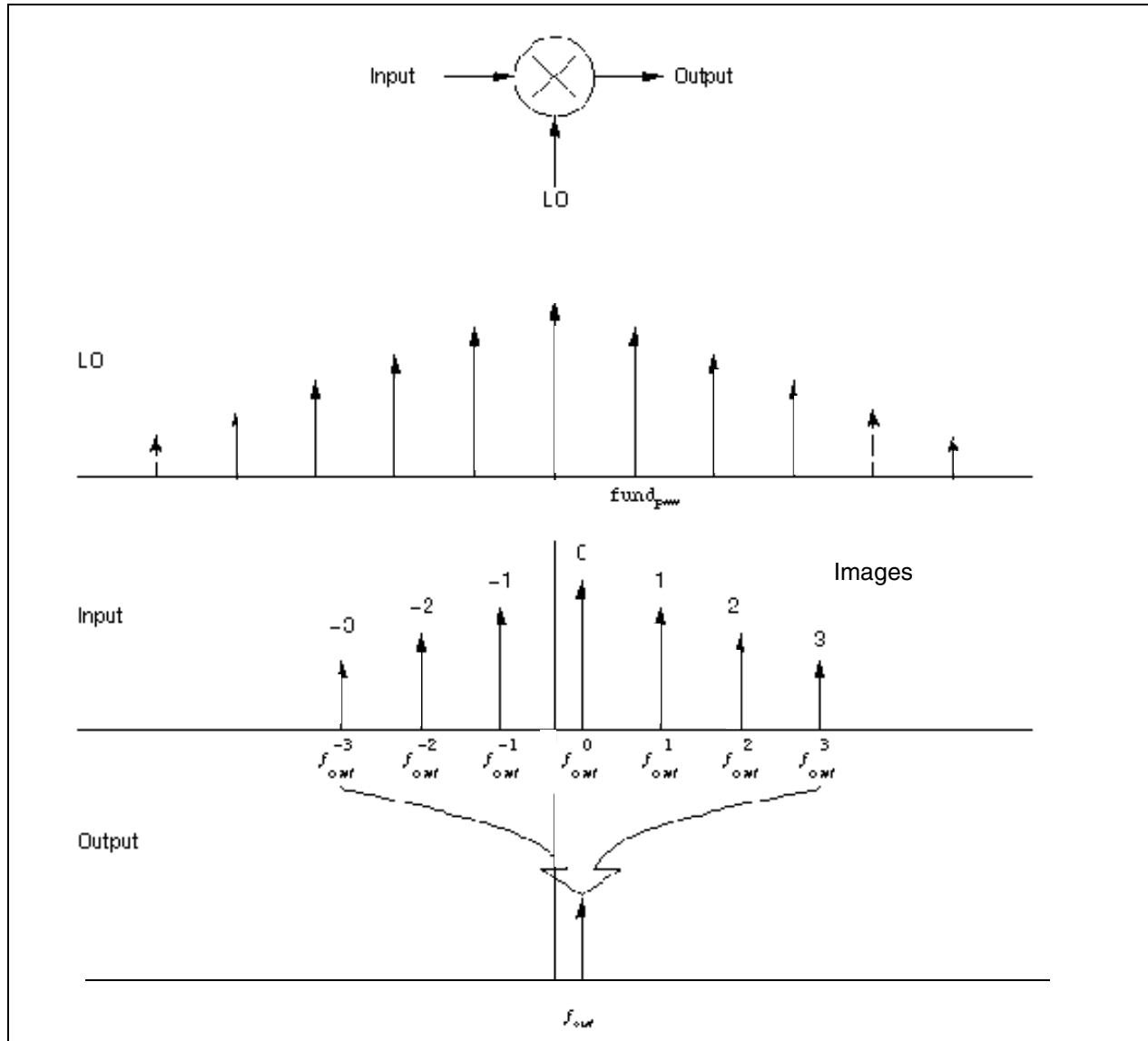
A PXF analysis cannot be used alone, it must follow a PSS analysis. However, any number of periodic small-signal analyses such as PAC, PSP, and Pnoise, can follow a single PSS analysis.

△ *Important*

> Unlike other Spectre RF small-signal analyses, the PXF analysis can sweep only frequency.

Parameters for PXF Analysis

For information on PXF analysis parameters, refer to the *Periodic Transfer Function Analysis (pxf)* section in the Spectre Circuit Simulator Reference manual.

Output Parameters

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 one of the `maxsideband` or `sideband` parameters.

For a set of *n* integer numbers representing the sidebands

$$K_1, K_2, \ldots K_n$$

the input signal frequency at each sideband is computed as

$$f(in) = f(out) + k_i \times fund(pss)$$

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, $k_i = +1$ for the RF input represents the first upper sideband, and $k_i = -1$ for the RF input represents the first lower sideband. If you specify the maximum sideband value by setting the `maxsideband` value to k_{max}, you are selecting all $2 \times k_{max} + 1$ sidebands from $-k_{max}$ to $+k_{max}$.

For PXF analysis, the number of sidebands you select with `maxsideband` does not substantially increase the simulation time. However, to ensure accurate PXF analysis results, set the `maxacfreq` parameter for the corresponding PSS analysis to guarantee that

$$|max\{f(in)\}| < maxacfreq$$

otherwise the computed solution might be contaminated by aliasing effects. The PXF simulation does not run when

$$|f(out)| > maxacfreq$$

For these extreme cases, diagnostic messages indicate how you should change the `maxacfreq` parameter value in the PSS analysis. In a majority of simulations, however, this is not an issue because the `maxacfreq` value is never allowed to be smaller than 40 times the PSS fundamental.

With PXF analysis the frequency of the stimulus and response are usually different (this is an important way in which the PXF analysis differs from the XF analysis). Use the `freqaxis` parameter to specify whether the results should be output versus the input frequency, `in`, the output frequency, `out`, or the absolute value of the input frequency, `absin`.

Probe Parameters

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. When there are more than two terminals, they are grouped in pairs. Use the `portv` parameter to select the appropriate pair of terminals. Alternatively, you can simply specify a voltage to be the output by giving a pair of nodes on the PXF analysis statement.

Any component that naturally computes current as an internal variable, such as a voltage source, can be a current probe. If the probe component computes more than one current, you use the `porti` parameter to select the appropriate current. Do not specify both the `portv` and `porti` parameters. If you do not specify either parameter, the probe component provides a reasonable default.

The Analog Circuit Design Environment (ADE) provides two ways to set probes on the Choosing Analyses form

■   When you specify input or output sources

■   When you specify positive or negative output nodes

Output Parameters

The `stimuli` parameter (on the PXF Options form) specifies the transfer function inputs. You select one of two choices:

■   Use `stimuli` = `sources` to use the sources present in the circuit. To compensate for gains or losses in the test fixture, use the `xfmag` source component parameter to adjust the computed gain to compensate for gains or losses in a test fixture. In hierarchical netlists, limit the number of sources using the `save` and `nestlvl` parameters.

■ Use `stimuli = nodes_and_terminals` to compute all possible transfer functions. Use this option when you cannot anticipate which transfer functions you might need to examine. This is useful when you do not know in advance which transfer functions are 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 PXF analysis computes the transfer functions from a small set of terminals.

For transfer functions from specific terminals, specify the terminals in the `save` statement. Use the `:probe` modifier (for example, `Rout:1:probe`) or specify `useprobes=yes` on the options statement. For transfer functions from all terminals, specify `currents=all` and `useprobes=yes` on the `options` statement.

Modulation Parameters

You can make modulated small-signal measurements from the Analog Circuit Design Environment (ADE). The `modulated` option for PXF analysis and other modulated parameters are set by ADE. The PXF analysis with the `modulated` option produces results which might have limited use outside the ADE environment. The Direct Plot form is configured to analyze modulated small-signal measurement results. Direct Plot can combine several wave forms to measure AM and PM transfer functions from single sideband or modulated stimuli to the specified output. For details, refer to the *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide*.

Sampled Small-Signal Analysis

Sampled small signal PXF and PAC analyses use the Analog Design Environment (ADE) environment. The sampled options are set by ADE. The Sampled option produces results which might have limited use outside ADE. Direct Plot is configured to analyze the results and make Sampled measurements due to single sideband or sampled stimuli. A sampled analysis is a small-signal analysis with a use model similar to the sampled (timedomain) PNoise analysis. Specifically, you first create a circuit (schematic or netlist description) and place port or source components to specify the key elements where the transfer function to the output is of interest.

The ability to sample the noise, signal slope and transfer function at a particular time point is a valuable investigative tool for design and verification. For example it will be used in a design of switched-capacitor filters, logic circuits and in the evaluation of the power supply rejection. A Sampled analysis computes the transfer function from different parts of the circuit to the

output at a particular time point. The output signal is sampled at the clock rate (large signal period which is used in PSS).

A new choice in the *Specialized Analysis* cyclic menu, *Sampled*, opens the *Sampled* analysis fields in the Choosing Analyses form. Several fields are available to specify the sampling event. First, you select a control signal to observe in search for the triggering event(s). It could be a single or differential voltage signal or a probe, either voltage or current. The threshold value and the crossing direction are the parameters of a triggering timing event. In case of special need, you can specify a delayed measurement from the time of the crossing event.

You can also specify actual time points for sampling the output. The same form will be used and either one or the combination of both approaches is used to do the sampled small signal analysis.

Swept PXF Analysis

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

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, use the `values` parameter to specify the particular values that the sweep parameter should take. 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.

Periodic Noise Analysis (Pnoise)

The Periodic Noise analysis (Pnoise) is similar to the conventional noise analysis except that it models frequency conversion effects. Hence Pnoise analysis is useful for predicting the noise behavior of mixers, switched-capacitor filters and other periodically driven circuits. The Pnoise analysis is particularly useful for predicting the phase noise of autonomous circuits, such as oscillators.

PNoise analysis linearizes the circuit about the periodic operating point computed in the prerequisite PSS analysis. It is the periodically time-varying nature of the linearized circuit that accounts for the frequency conversion. In addition, the affect of a periodically timevarying bias point on the noise generated by the various components in the circuit is also included.

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 2-5. 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 2-5 How noise is moved around by a mixer
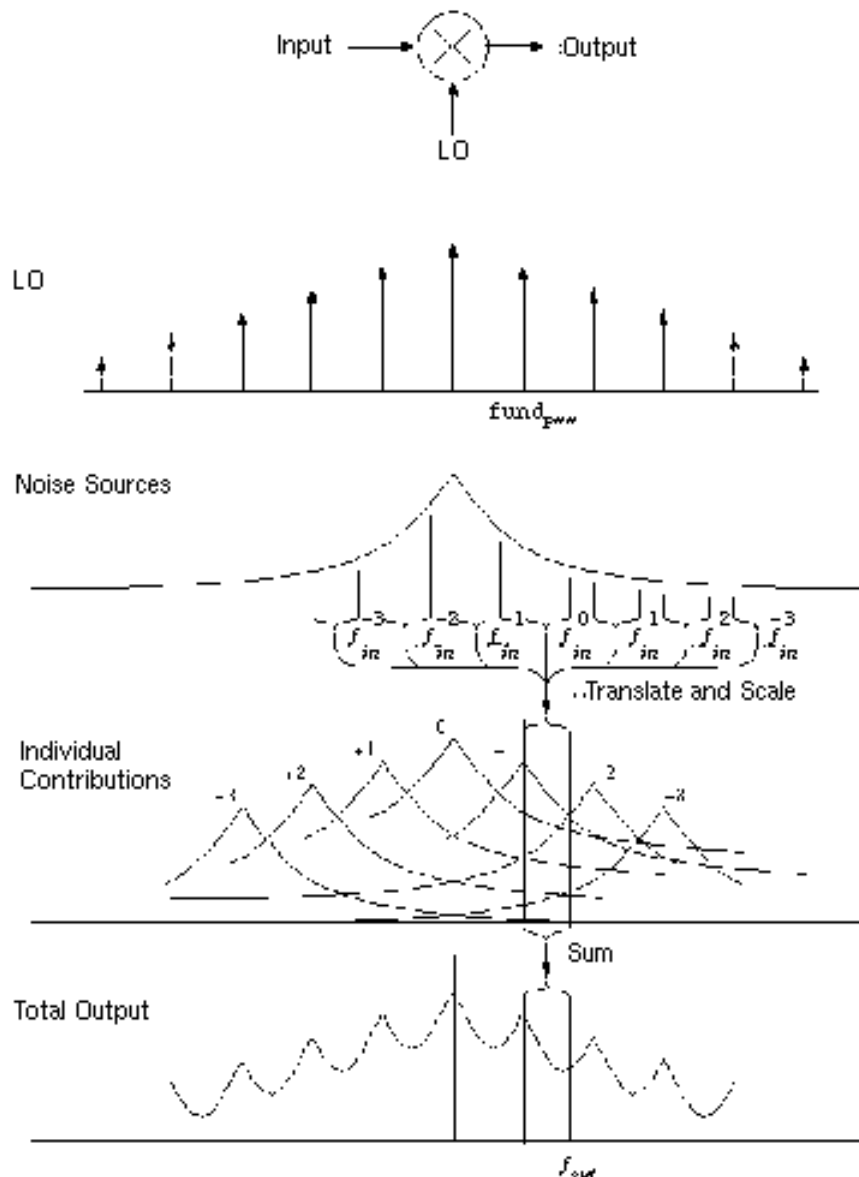
The time-average of the noise at the output is computed as a spectral density versus frequency. You identify the output by specifying a probe component or a pair of nodes. To specify the output with a probe, the preferred approach, use the `oprobe` parameter. If the output is voltage (or potential), 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.

To compute the input-referred noise or the noise figure, specify the input source using the `iprobe` parameter. For input-referred noise, use either a `vsource` or `isource` as the input probe; for noise figure, use a `port` as the probe. 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 its `noisetemp` must be set to 16.85C (290K). In addition, the output load must be a `resistor` or `port` and must be identified as the `oprobe`.

If `port` is specified as the input probe, then both input-referred noise and gain are referred back to the equivalent voltage source inside the `port`. S-parameter analysis calculates those values in the traditional sense.

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)	

Noise Calculations Performed by Pnoise

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$_{ieee}$**

Spectre RF 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 $f(\mathrm{out}) - maxsideband\ fund(\mathrm{pss})$ and greater than $f(\mathrm{out}) + maxsideband\ fund(\mathrm{pss})$. 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 practice, noise can mix with each of the harmonics of the periodic drive signal applied in the PSS analysis and end up at the output frequency. However, the PNoise analysis only includes the noise that mixes with a finite set of harmonics that are typically specified using the `maxsideband` parameter, but in special circumstances may be specified with the `sidebands` parameter. If Ki represents sideband i, then

$\quad$ *f (noise_source) = f (out) + Ki * fund(pss)*

The `maxsideband` parameter specifies the maximum *|Ki|* included in the PNoise calculation. Thus, noise at frequencies less than *f(out)-maxsideband*fund(pss)* and greater than *f(out)+maxsideband*fund(pss)* are ignored. If selected sidebands are specified using the `sidebands` parameter, then only those are included in the calculation. You should take care when specifying the sidebands because the results will be in error if you do not include a sideband that contributes significant noise to the output.

The number of requested sidebands does not change substantially the simulation time. However, the `maxacfreq` of the corresponding PSS analysis should be set to guarantee that *|max{f(noise_source)}|* is less than `maxacfreq`, otherwise the computed solution might be contaminated by aliasing effects. The PNoise simulation is not executed for *|f(out)|* greater than `maxacfreq`. Diagnostic messages are printed for those extreme cases, indicating which `maxacfreq` should be set in the PSS analysis. In the majority of the simulations, however, this is not an issue, because `maxacfreq` is never allowed to be smaller than 40 times the PSS fundamental.

Phase Noise measurements are possible using the Analog Design Environment (ADE). Two Pnoise analyses are preconfigured for this simulation and most of the parameters are set by ADE.

■ The `mod1` Pnoise analysis is a regular noise analysis and can be used independently.

■ The `mod2` Pnoise analysis is a correlation analysis and has limited use outside of the ADE environment.

The Direct Plot form in ADE is configured to analyze these results and combine several waveforms to measure AM and PM components of output noise.

You can specify sweep limits by giving the end points or by providing the center value and the span of the sweep. Steps can be linear or logarithmic, and you can 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 specified using a sweep range, the two sets are merged and collated before being used. All frequencies are in Hz.

Parameters for Pnoise Analysis

For information on pnoise analysis parameters, refer to the *Periodic Noise Analysis (pnoise)* section in the Spectre Circuit Simulator Reference manual.

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.

Frequency-Aware PPV Analysis for Oscillators with Large Time Constants

Phase noise, because it has an impact on overall system performance, is a major concern in oscillator circuit design. The perturbation projection vector (PPV) models often used to analyze oscillators overestimate the phase noise in oscillators with large time constants, such as oscillators that include switch capacitors, DC bias, or digital dividers. The overestimation

error happens because the slow nodes in oscillators filter out the noise from nearby devices and the PPV does not consider these filtering effects.

Setting the `augmented` parameter to `yes` turns on an analysis that considers the PPV as a frequency dependent quantity. This frequency-aware analysis takes into account the fact that different perturbation frequencies have different PPV waveforms. The analysis provides the same answers as ordinary PPV analysis for small, fast oscillators but provides a more accurate result for oscillators with large time constants.

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 2-6 shows the true output noise of a mixer with flicker noise.

Figure 2-6 Actual Mixer Noise Output Including Flicker Noise

True output noise of a mixer with flicker noise
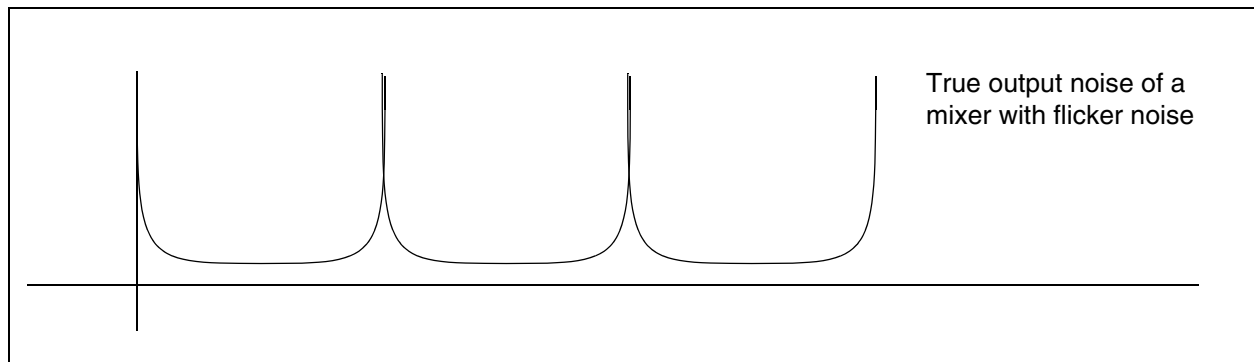
Figure 2-7 shows the output noise computed with a typical choice of points. In this case, total output noise is typically exaggerated by many orders of magnitude.

Figure 2-7 Noise Output Computed With Typical Points

Noise output computed with typical choice of points. Total output noise is typically exaggerated by many orders of magnitude.
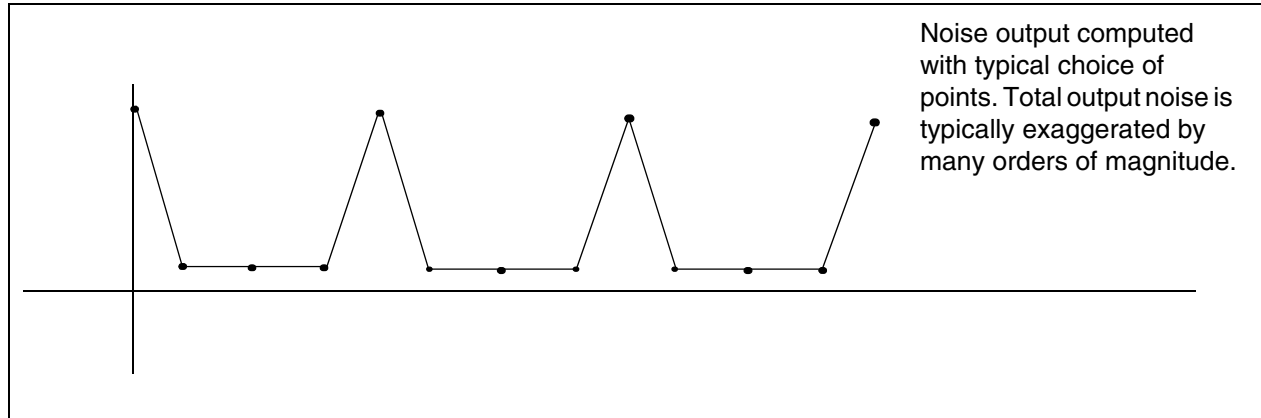
Figure 2-8 shows noise output computed when the values parameter is used to cluster points near harmonics. By comparing Figures 2-7 and 2-8, you can see that total output noise is computed accurately when points are carefully chosen.

Figure 2-8 Noise Output Computed With Clustered Points

Noise output computed when *values* parameter is used to cluster points near harmonics. Total output noise is computed accurately.
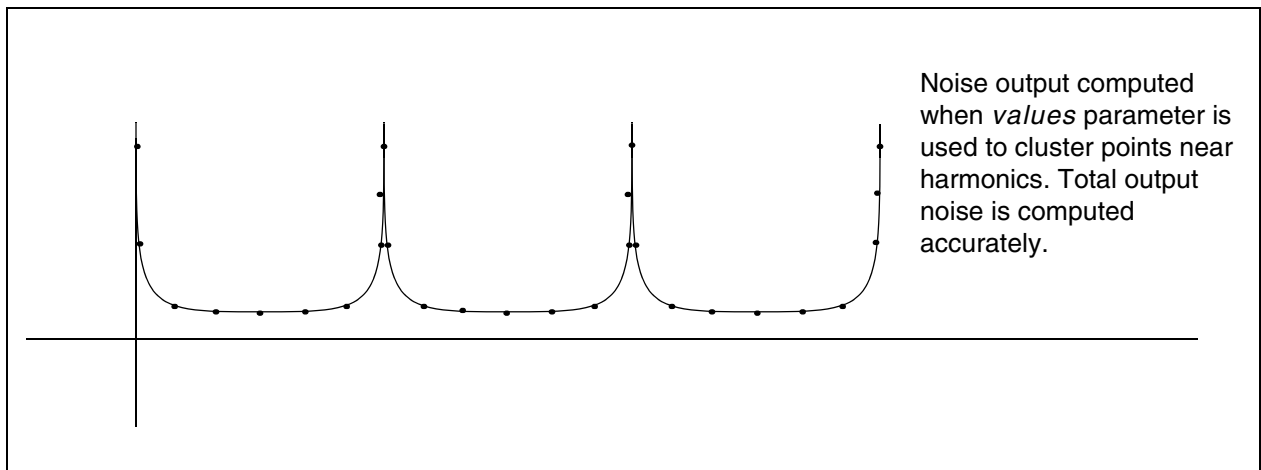
Flicker Noise Spectrum

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

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

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

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

Periodic Stability Analysis (PSTB)

The periodic stability (PSTB) analysis evaluates the local stability of a periodically time-varying feedback circuit. It is a small-signal analysis, like STB analysis, except that the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing about a periodically time-varying operating point allows the stability evaluation to include the effect of the time-varying operating point.

The stability evaluation 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 a 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, the small stimulus is applied to compute the loop gain of the zero sideband with a `probe` component. The local stability can be evaluated using gain margin, phase margin, or a Nyquist plot of the loop gain. To perform PSTB analysis, you must use a `probe` instance and specify it with the `probe` parameter.

The loop-based algorithm requires that you place the `probe` on the feedback loop to identify and characterize the particular loop of interest. The introduction of the `probe` component should not change any of the circuit characteristics. Because of the time-varying properties of the circuit, the loop gain at different places might be different but you can use the loop gain at any point to evaluate stability.

The loop-based algorithm provides stability information for both single loop circuits and for multi-loop circuits in which you can place a `probe` component on a critical wire to break all loops. For a general multi-loop circuit, such a critical wire might not be available. The loop-based algorithm can only be performed on individual feedback loops to ensure they are stable.

The device based algorithm requires the probe be a gain instant, such as a bjt transistor or a mos transistor. The device-based algorithm evaluates the loop gain around the probe, which can be involved in multi-loops.

Unlike other analyses in Spectre RF, this analysis can only sweep frequency.

Parameters for PSTB Analysis

For information on PSTB analysis parameters, refer to the *Periodic STB Analysis (pstb)* section in the Spectre Circuit Simulator Reference manual.

Sweep

You can specify sweep limits by providing

- The end points of the sweep

- The center value and the span of the sweep

- An array of specific values to 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 provide 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.

Understanding Loop-Based and Device-Based Algorithms

Both loop-based and device-based algorithms are available for periodic small-signal stability analysis. When the `probe` parameter points to a current probe or voltage source instance, the loop-based algorithm is used; when it points to a supported active device instance, the device-based algorithm is used.

About the PSTB Loop-Based Algorithm

The PSTB loop-based algorithm is based on a subset of Nyquist criteria. The analysis outputs the loop gain waveform.

The PSTB loop-based algorithm calculates the true loop gain, which consists of both normal loop gain and reverse loop gain. The loop-based algorithm requires that you place a `probe` component in the feedback loop to identify and characterize the particular loop of interest. Introducing the `probe` component should not change circuit characteristics.

The loop-based algorithm provides accurate stability information for single loop circuits. It also provides accurate stability information for multi-loop circuits in which you can place a `probe` component on a critical wire to break all loops. For general multi-loop circuits, such a critical wire may not be available. The loop-based algorithm can only be performed on individual feedback loops to ensure they are stable. Although the stability of all feedback loops

is a necessary condition for the whole circuit to be stable, the multi-loop circuit tends to be stable if all individual loops are associated with reasonable stability margins.

Device-Based Algorithm

The device-based algorithm calculates the loop gain around a particular active device, which must be a gain instance such as a bjt or mos transistor. This algorithm is often applied to assess the stability of circuit designs in which local feedback loops cannot be neglected. The loop-based algorithm cannot be used for such designs because the local feedback loops are inside the devices where a `probe` component cannot be inserted.

When the `probe` parameter points to a particular active device, the dominant controlled source in the device is nulled during the analysis. The device-based algorithm produces accurate stability information for a circuit in which a critical active device can be identified such that nulling the dominant gain source of this device renders the whole network to be passive.

Quasi-Periodic Steady-State Analysis (QPSS)

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 dynamics in multiple fundamental frequencies. Closely spaced or incommensurate fundamentals cannot be efficiently resolved by PSS analysis. (Incommensurate frequencies are those for which there is no period that is an integer multiple of the period of each frequency.) QPSS analysis allows you to compute circuit responses to several moderately large input signals in addition to a strongly nonlinear tone which represents the LO or clock signal. A typical example is the intermodulation distortion measurements of a mixer with two closely spaced moderate input signals. QPSS treats one particular input signal (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 using the shooting engine 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.

Like 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. When you set up a QPSS analysis, you determine the number of integrations performed by the number of harmonics of moderate fundamentals you select.

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

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

QPSS always treats k1 as the maximum harmonic of the large signal and the total number of integrations for the simulation is calculated as

$$(2k_2 + 1) \times (2k_3 + 1) \times \ldots \times (2k_n + 1)$$

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 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:

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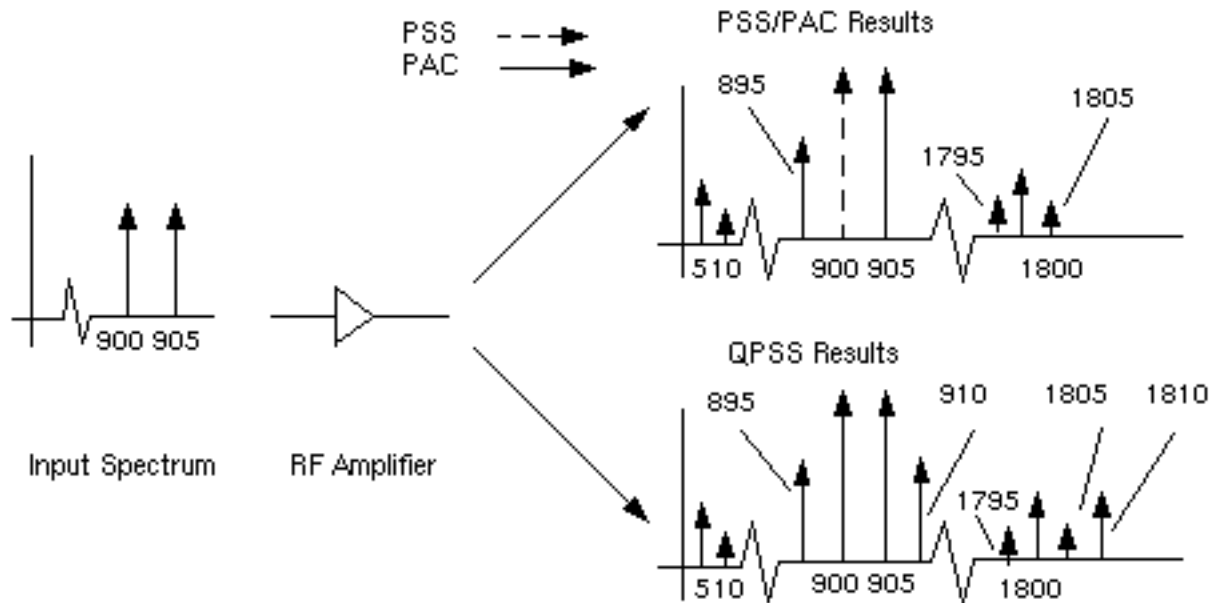
where

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

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 2-9 shows that more information is available from a QPSS analysis than from a PSS analysis followed by a PAC analysis.

Figure 2-9 Comparison of Information From QPSS and PSS/PAC Analyses

QPSS Parameters

For information on QPSS analysis parameters, refer to the _Quasi-Periodic Steady State Analysis (qpss)_ section in the Spectre Circuit Simulator Reference manual.

The QPSS analysis uses the same parameters as the PSS analysis with a couple of parameters added and a few parameters extended. The most important parameters for QPSS analysis are the `funds` and `maxharms` parameters, which replace and extend the `fund` (or `period`) and `harms` parameters that are used in PSS analysis.

The `funds` parameter accepts a list of names of fundamentals that are present in the sources. (These fundamental names are specified by the source parameter `fundname`.) The simulator automatically 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, provided that all the sources have the same fundamental name. The fundamental frequency for each fundamental name is the greatest common factor of all the frequencies associated with the name. Simulation terminates if you do not list all the fundamental names on the `funds` parameter.

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 the large signal. You can use a few heuristics to pick the large fundamental.

■ Pick the fundamental that is not sinusoidal.

■ Pick the fundamental that causes the most nonlinearity.

■ Pick the fundamental that 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. Its default value is 50. 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.

The errpreset Parameter in QPSS Analysis

The `errpreset` parameter quickly adjusts several simulator accuracy parameters to fit your needs.

⚠ *Important*

In most cases, `errpreset` should be the only accuracy 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 a value for `steadyratio`, it is always 1.0, and it is not affected by `errpreset`.

Table 2-5 shows the effect of `errpreset` settings (`liberal`, `moderate`, and `conservative`) on the default values of a set of accuracy parameters.

Table 2-5 Parameter Default Values for `errpreset` Settings

errpreset	reltol	relref	method	lteratio	maxstep
`liberal`	$1e^{-3}$	sigglobal	gear2only	3.5	clock period/80
`moderate`	$1e^{-4}$	sigglobal	gear2only	3.5	clock period/100
`conservative`[1]	$1e^{-5}$	sigglobal	gear2only	*	clock period/200

1. * : lteratio=10.0 for conservative errpreset by default. However, when the specified reltol <= 1e-5*10.0/3.5, lteratio is set to 3.5..

These `errpreset` settings include a default `reltol` value which is an enforced upper limit for `reltol`. An increase of `reltol` above the default is ignored by the simulator. The only way to *decrease* the `reltol` value is in the `options` statement. The only way to *increase* the `reltol` value is to relax the `errpreset` setting. Spectre RF sets the `maxstep` parameter value so that it is no larger than the value given in Table 2-5.

With the exception of 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. If `errpreset` is not specified in the netlist, `liberal` settings are used.

For HB, only reltol is affected by errpreset, and the effect is the same as that in shooting. However, `lteratio` remains 3.5 and `steadyratio` remains 1 with all values of `errpreset`.

With parameter `hbhomotopy`, you can specify harmonic balance homotopy selection methods. The possible values of parameter hbhomotopy and their meanings are as follows:

`hbhomotopy=tstab`: Simulator runs a transient analysis and generates an initial guess for harmonic balance analysis; it is recommended for nonlinear circuits or circuits with frequency dividers.

`hbhomotopy=source`: For driven circuit, simulator ignores tstab and accordingly increases the source power level; for oscillators, the simulator accordingly adjusts the probe magnitude until the probe has no effect on the oscillators. It is recommended for strongly nonlinear or high Q circuits.

`hbhomotopy=tone`: This method is valid only for multi-tone circuit. The simulator first solves a single-tone circuit by turning off all the tones except the first one, and then solves the multitone circuit by restoring all the tones and using the single-tone solution as its initial guess; it is recommended for multi-tone simulation with a strong first tone.

`hbhomotopy=inctone`: Simulator first solves a single tone, then turns on moderate tones incrementally till all tones are enabled. It is recommended for circuits with one strong large tone.

`hbhomotopy=gsweep`: A resistor, whose conductance is g, is connected with each node, and the sweep of g is controlled by gstart, gstop, and glog. It is recommended for circuits containing high-impedance or quasi-floating nodes.

Quasi-Periodic Noise Analysis (QPnoise)

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`.

If `port` is specified as the input probe, then both input-referred noise and gain are referred back to the equivalent voltage source inside the port. S-parameter analysis calculates those values in traditional sense.

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:

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

The reference sideband option (`refsidebandoption`) specifies whether to consider the input at the frequency or the input at the individual quasi-periodic sideband specified. Note that different sidebands can lead to the same frequency.

Sidebands are vectors in QPnoise analysis. Assume you have one large tone and one moderate tone in a QPSS analysis. A sideband K_i will be a vector *[K_i_1 K_i_2]*. It gives the frequency at

*K_i_1 * fund(large tone of QPSS) + K_i_2 * fund(moderate tone of QPSS)*

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 computed and 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)	

Noise Calculations Performed by QPnoise

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
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 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.

Spectre RF 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 of the quasi-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 that are specified using the `clockmaxharm` and `sidevec` parameters.

The clockmaxharm parameter affects only clock frequency. It can be less or more than `maxharms[1]` in QPSS. Moderate tones are limited by `maxharms` specified in QPSS. Only the selected sidebands specified using the `sidevec` parameter are included in the calculation. Care should be taken when specifying sidevec or clockmaxharm in QPNOISE and maxharms in QPSS. Noise results are erroneous if you do not include the sidebands that contribute 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

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

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

```
K¹ = { K¹₁, ..., K¹ⱼ, ..., K¹ₗ}
K² = { K²₁, ..., K²ⱼ, ..., K²ₗ}
…
Kⁿ = { Kⁿ₁, ..., Kⁿⱼ, ..., Kⁿₗ}
```

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`$=[k^0_{max} \; k^2_{max} \ldots \; k^n_{max}]$

■   In the QPnoise analysis input, `clockmaxharm`$=K_{max}$

The QPnoise analysis output generates the following number of sidebands

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

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

For information on QPnoise analysis parameters, refer to the _Quasi-Periodic Noise Analysis (qpnoise)_ section in the Spectre Circuit Simulator Reference manual.

Quasi-Periodic AC Analysis (QPAC)

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

$$K^1_1 \times fund(qpss - large - tone) + K^1_2 \times fund(qpss - moderate - tone)$$

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 QPAC 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.

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 `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

- `maxharms=[`k^1_{max} k^2_{max}`...` k^n_{max}`]`

- `clockmaxharm=`K_{max}

The following sidebands are generated:

`(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

For information on QPAC analysis parameters, refer to the *Quasi-Periodic AC Analysis (qpac)* section in the Spectre Circuit Simulator Reference manual.

Quasi-Periodic S-Parameter Analysis (QPSP)

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

$$
K^1_1 \times fund(qpss - l\arg e - tone) + K^1_2 \times fund(qpss - moderate - tone)
$$

$$
= \sum_{i=1}^{j} K^j_i \cdot f_j
$$

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

```
K¹ = { K¹₁, ..., K¹ⱼ, ..., K¹ₗ}
K² = { K²₁, ..., K²ⱼ, ..., K²ₗ}
...
Kⁿ = { Kⁿ₁, ..., Kⁿⱼ, ..., Kⁿₗ}
```

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.

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

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 IF port with $f(scattering) = f(IF) = f(RF) + \Delta f(RF \, to \, IF) = f(incident) - f(LO)$.

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

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

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

⚠ *Important*

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

$$\left[K_1^1 K_2^1 \right]$$

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.

Spectre RF 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 ensure 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

For information on QPSP analysis parameters, refer to the _Quasi-Periodic S-Parameter Analysis (qpsp)_ section in the Spectre Circuit Simulator Reference manual.

Quasi-Periodic Transfer Function Analysis (QPXF)

A conventional transfer function (QPXF) analysis computes the transfer function from every source in the circuit to a single output. It differs from a conventional AC analysis in that the AC analysis computes the response from a single stimulus to every node in the circuit. The differences between the QPAC and QPXF analyses are similar. The Quasi Periodic Transfer Function or QPXF analysis computes the transfer functions from any source at any frequency to a single output at a single frequency. Thus, like QPAC analysis, QPXF analysis includes frequency conversion effects.

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 transfer function from the input to output at an undesired frequency.

■   LO Feed-Through and Power Supply Rejection – The transfer function from an 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

$$K^1_1 \times fund(qpss - large - tone) + K^1_2 \times fund(qpss - moderate - tone)$$

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

$$
\begin{aligned}
K^1 &= \{ \; K^1_1, \; \ldots, \; K^1_j, \; \ldots, \; K^1_L \} \\
K^2 &= \{ \; K^2_1, \; \ldots, \; K^2_j, \; \ldots, \; K^2_L \} \\
&\ldots \\
K^n &= \{ \; K^n_1, \; \ldots, \; K^n_j, \; \ldots, \; K^n_L \}
\end{aligned}
$$

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

For information on QPXF analysis parameters, refer to the *Quasi-Periodic Transfer Function Analysis (qpxf)* section in the Spectre Circuit Simulator Reference manual.

Harmonic Balance Steady State Analysis (HB)

This analysis uses harmonic balance (in the frequency domain) to compute the response of circuits that have either one fundamental frequency (periodic steady-state, PSS) or that have multiple fundamental frequencies (quasi-periodic steady-state, QPSS). The simulation time required for an HB analysis is independent of the time-constants of the circuit. This analysis also determines the circuit's periodic or quasi-periodic operating point, which can then be used during a periodic or quasi-periodic time-varying small-signal analysis, such as HBAC or HBnoise.

Usually, harmonic balance (HB) analysis is a very efficient way to simulate weakly nonlinear circuits. Also, HB analysis works better than shooting analysis (in the time domain) for frequency dependent components, such as delay, transmission line, and S-parameter data.

An HB analysis consists of two phases. The first phase calculates an initial solution, which the second phase then uses to compute the periodic or quasi-periodic steady-state solution, using the Newton method.

The two most important parameters for HB analysis are `funds` and `maxharms`. The funds parameter accepts a list of names of fundamentals that are present in the sources. These names are specified in the sources by the `fundname` parameter. When only one name appears, the analysis is an HB PSS analysis. When more than one name appears, the analysis is an HB QPSS analysis. The `maxharms` parameter accepts a list of numbers of the harmonics that are required to adequately model the responses due to the different fundamentals.

The annotate parameter has two values specific to HB analysis: `detailed_hb` and `internal_hb`. When annotate is set to `detailed_hb` or `internal_hb`, additional analysis debug information is printed to the log files. In the case of `internal_hb`, encrypted debug information is stored in the internal log file. Both options are valid for pss and qpss analyses with `flexbalance=yes`.

Using Multi-rate Harmonic Balance to Improve Performance

When the signals in different parts of an RF circuit have different fundamental frequencies and numbers of harmonics, it is often unnecessary and computationally costly to use a single harmonic set for all parts of the circuit. The multi-rate harmonic balance capability, by dividing the circuit into parts based on the signals contained in them, provides a way to avoid this problem.

For example, consider an RF circuit with three parts (a divider, a mixer, and an output module) and three tones (`LO`, `RF1`, and `RF2`).
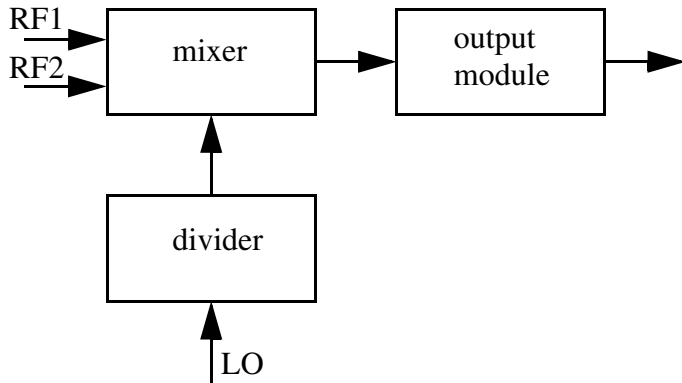
The divider has no interaction with RF1 and RF2 so the harmonic set can be [10 0 0]; the mixer interacts with all the tones so that harmonic set can be [5 3 3]; and the output module harmonic set can be [3 3 3]. If the HB engine had to use a single harmonic set, it would be [10 3 3] for all the modules, but the multi-rate harmonic balance capability allows different harmonic sets to be used for different parts of the circuit. In this example, the design can be set up to use [10, 0, 0] for the divider; [5, 3, 3] for the mixer, and [3, 3, 3] for the output.

In other words, HB multi-rate harmonic balance is a method of decomposing a circuit so that multi-rate behavior can be exploited to increase simulation performance. Of course, if every part of a circuit has the same spectrum structure (such as fundamental frequency and bandwidth), there is no need to use multi-rate harmonic balance.

Note: Multi-rate harmonic balance is only supported for driven circuits, but will be supported for autonomous circuits at a future time.

HB Synopsis

Name ([*p*] [*n*]) hb <*parameter=value*> ...

HB Parameters

For information on harmonic balance parameters, refer to the *Harmonic Balance Steady State Analysis (hb)* section in the Spectre Circuit Simulator Reference manual.

Details about Using HB Analysis Parameters

The initial transient analysis provides a flexible mechanism to direct the circuit to a particular steady-state solution of interest and to avoid undesired solutions. The initial transient simulation also helps convergence by eliminating the large, but fast decaying, modes that are present in many circuits.

In some circuits, the linearity of the relationship between the initial and final states depends on when HB analysis begins. In practice, starting analysis at a good point can improve convergence, and starting at a bad point can degrade convergence and slow the analysis.

oscic Parameter

When HB analysis is used for oscillators, initialization is performed to obtain an initial guess of the steady state solution and of the oscillating frequency. Two initialization methods are implemented, based on transient and linear analysis.

- When `oscic=default` is specified, transient initialization is used and the length of the transient is specified by `tstab`. You must start the oscillator using initial conditions or using a brief impulsive stimulus, just as you would if you were simulating the turn-on transient of the oscillator using transient analysis. Initial conditions would be provided for the components of the oscillator's resonator. If an impulsive stimulus is used, it should be applied so as to couple strongly into the oscillatory mode of the circuit and poorly into any other long-lasting modes, such as those associated with bias circuitry. The Designer's Guide to Spice and Spectre [K. S. Kundert, Kluwer Academic Publishers, 1995] describes in depth some techniques for starting oscillators.

- When `oscic=lin` is specified, linear initialization is used. In this method, both oscillation frequency and amplitude are estimated based on linear analysis at the DC solution. No impulsive stimulus or initial conditions are needed. Linear initialization is suitable for linear oscillators such as LC and crystal oscillators. Note that `tstab` transient is still performed after linear initialization, though it can be significantly shortened or skipped. Either way, specifying a non-zero `tstab` parameter can improve convergence.

funds Parameter

For the `funds` parameter, the frequencies associated with fundamentals are figured out automatically by the simulator. An important feature is that each input signal can be a composition of more than one source. However, these sources must have the same fundamental name. For each fundamental name, the fundamental frequency is the greatest common factor of all frequencies associated with the name. Omitting a fundamental name in the `funds` parameter is an error that stops the simulation. If `maxharms` is not given, a

warning message is issued, and the number of harmonics defaults to 1 for each of the fundamentals in a multi-tone simulation and to 10 in a single-tone simulation.

hbpartition_defs Parameter

HB signal partition is a method of decomposing a circuit so that multi-rate behavior can be exploited to increase simulation performance. If every part of a circuit has the same spectrum structure, such as fundamental frequency and bandwidth, there is no need to apply signal partition. However, if the RF circuit has multiple tones, the signals in different parts can have various spectrum structures, such as different fundamental frequency and number of harmonics. With HB signal partition, you can divide the circuit into several parts based on the signals contained in them. The parameter hbpartition_defs defines the partitions. Each partition can be made up of one or more instances. For example,

```
hbpartition_defs = ["I9 I10" "I11 I12" "I13 I14"]
```

defines three partitions. The first partition consists of instance "I9" and "I10" while the second partition consists of instances "I11" and "I12".The third one has "I13" and "I14". The number of instances for each partition should not be less than 1 and there is no upper limit for the number.

hbpartition_harms Parameter

The principle for dividing a circuit is that the subcircuits or instances with the same spectrum properties should be put into one partition. The parameter `hbpartition_harms` specifies the maximum number of positive harmonics of each tone for every partition. For example:

```
hbpartition_harms=["10 0 0" "5 3 3" "3 3 3"]
```

So, the maximum number of positive harmonics for the first partition is 10, 0 and 0, respectively. For the second partition, it is 5, 3 and 3. For the last one, it is 3, 3 and 3.

hbpartition_fundratios Parameter

The parameter `hbpartition_fundratios` indicates the fundamental frequency ratio of each tone for each partition. With these ratios, it is easy to know the fundamental frequencies of each tone of the partitions. For example:

```
hbpartition_fundratios=["2 1 1" "1 1 1" "1 1 1"]
```

If three global fundamental frequencies are defined as: LO=1GHz, RF1=1.1GHz and RF2=1.13GHz, it indicates the fundamental frequencies of each tone of the first partition is 2*LO, 1*RF1, and 1*RF2, respectively. The second and third partition has the same frequencies for their each tone: 1*LO, 1*RF1 and 1*RF2. However, you have to make sure that the global fundamental frequencies for each tone are the smallest among all the partitions and the ratios are integers.

cmin Parameter

If the circuit you are simulating can have infinitely fast transitions (for example, a circuit that contains nodes with no capacitance), the simulator might not converge well. To avoid this, you must prevent the circuit from responding instantaneously. You can accomplish this by setting `cmin`, the minimum capacitance to ground at each node, to a physically reasonable nonzero value. This often significantly improves convergence.

hbhomotopy Parameter

The convergence rate of large signal analyses is mainly determined by two factors: the initial condition and the nonlinearity of the circuit. When the initial condition is close to the true solution and the nonlinearity is small, convergence is typically fast. One way to calculate the initial condition is by running a transient analysis but with this method it is hard to determine how long the transient analysis needs to run.

The homotopy method uses a different approach. Given a circuit with strong nonlinearity, the homotopy method obtains a solution by starting from the same circuit but with an altered, much lower nonlinearity. Then the method increases the nonlinearity, using the solution from the lower nonlinearity as the initial condition. When the nonlinearity of the circuit reaches the original nonlinearity, a good initial condition is available from the previous solution so convergence is fast. Source stepping is used to change the nonlinearity of the circuit.

In another method, a resistor and a parallel capacitor are added between each circuit node and the ground. These components reduce the dynamic range of the circuit and help with convergence. Initially, this method obtains a solution by assuming a small resistor value. Then the method increases the resistance and obtains a new solution, using as the initial value the solution obtained on the previous iteration. When the resistance becomes large enough, the inserted components become negligible and the iteratively derived solution provides a good guess of the true solution.

The parameter used to control HB homotopy is called `hbhomotopy`. The parameter has the possible values `source`, `tone`, `tstab` (which is the default), and `gsweep`.

■ For the `source` value, the initial condition is generated by stepping the RF source level. This value is appropriate when the power level of the large tone or of the RF tones is high. Because there is no source available in autonomous circuits, this value is not applicable to oscillators.

■ For the `tone` value, a single tone solution is generated by turning off all the tones except the first one. The multi-tone circuit is then solved by restoring all the tones and using the single-tone solution as the initial guess. This value is appropriate for multitone designs with a strong first tone.

■ For the `tstab` value, the initial condition is generated by the usual transient analysis described above. This value is appropriate when the circuit contains devices, such as digital frequency dividers, that display strong non-linear behavior.

■ For the `gsweep` value, a resistor of conductance g is connected with each node. The conductance g is swept under the control of the `gstart`, `gstop`, and `glog` parameters. This value is appropriate for circuits containing high-impedance or quasi-floating nodes.

ic Parameter

You can specify the initial condition for the transient analysis by using the `ic` statement or the `ic` parameter on the capacitors and inductors. If you do not specify the initial condition, the DC solution is used as the initial condition.

If you specify an initial condition file with the `readic` parameter, initial conditions from the file are used, and any `ic` statements are ignored.

When you specify initial conditions, the simulator computes the actual initial state of the circuit by performing a DC analysis. During this analysis, the simulator forces the initial conditions on nodes by using a voltage source in series with a resistor whose resistance is `rforce` (see options).

With the `ic` statement, it is possible to specify an inconsistent initial condition (one that cannot be sustained by the reactive elements). Examples of inconsistent initial conditions include setting the voltage on a node with no path of capacitors to ground or setting the current through a branch that is not an inductor. If you initialize the simulator inconsistently, its solution jumps; that is, it changes instantly at the beginning of the simulation interval. You should avoid such changes because the simulator can have convergence problems while trying to make the jump.

Initial conditions and nodesets have similar implementations but produce different effects. Initial conditions actually define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. However, differential equations have an infinite number of solutions, and a complete set of initial conditions must be specified to identify the desired solution. Any initial conditions you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator toward finding the solution closest to the nodeset values.

readns Parameter

Nodesets help the simulator find the DC or initial transient solution. You can supply nodesets in the circuit description file with `nodeset` statements, or in a separate file using the `readns` parameter. When nodesets are given, Spectre computes an initial guess of the solution by performing a DC analysis while forcing the specified values onto nodes by using a voltage source in series with a resistor whose resistance is `rforce`. Spectre then removes these voltage sources and resistors and computes the true solution from this initial guess.

Nodesets have two important uses. First, if a circuit has two or more solutions, nodesets can bias the simulator toward computing the desired one. Second, they are a convergence aid. By estimating the solution of the largest possible number of nodes, you might be able to eliminate a convergence problem or dramatically speed convergence.

Nodesets and initial conditions have similar implementations but produce different effects. Initial conditions actually define the solution, whereas nodesets only influence it. When you simulate a circuit with a transient analysis, Spectre forms and solves a set of differential equations. However, differential equations have an infinite number of solutions, and a complete set of initial conditions must be specified to identify the desired solution. Any initial conditions you do not specify are computed by the simulator to be consistent. The transient waveforms then start from initial conditions. Nodesets are usually used as a convergence aid and do not affect the final results. However, in a circuit with more than one solution, such as a latch, nodesets bias the simulator toward finding the solution closest to the nodeset values.

skipdc Parameter

You can skip the DC analysis entirely by using the `skipdc` parameter. If the DC analysis is skipped, the initial solution will be either trivial or given in the file you specified by the `readic` parameter, or, if the `readic` parameter is not given, the values specified on the `ic` statements. Device-based initial conditions are not used for `skipdc`. Nodes that you do not specify with the `ic` file or `ic` statements start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution; it usually takes longer to follow the initial transient spikes that occur when the DC analysis is skipped than it takes to find the real DC solution. The `skipdc` parameter might also cause convergence problems in the transient analysis.

Harmonic Balance AC Analysis (HBAC)

The harmonic balance AC (HBAC) analysis computes transfer functions for circuits that exhibit single or multi-tone frequency translation. Such circuits include mixers, switched-capacitor filters, samplers, phase-locked loops, and the like. HBAC is a small-signal analysis like AC analysis, except the circuit is first linearized about a periodically or quasi-periodically varying operating point rather than about a simple DC operating point. Linearizing about a periodically or quasi-periodically time-varying operating point allows transfer-functions that include frequency translation, whereas simply linearizing about a DC operating point cannot because linear time-invariant circuits do not exhibit frequency translation. Also, the frequency of the sinusoidal stimulus is not constrained by the period of the large periodic solution.

Computing the small-signal response of a periodically or quasi-periodically varying circuit is a two step process. First, the small stimulus is ignored and the periodic or quasi-periodic steady-state response of the circuit to possibly large periodic stimuli is computed using HB analysis. As a normal part of the HB analysis, the periodically or quasi-periodically time-varying representation of the circuit is computed and saved for later use. Second, the small stimulus is applied to the periodically varying linear representation to compute the small signal response. This is done using the HBAC analysis. A HBAC analysis cannot be used alone, it must follow a HB analysis. However, any number of periodic or quasi-periodic small-signal analyses, such as HBAC or HBnoise, can follow a HB analysis.

Modulated small signal measurements are possible using the Analog Design Environment (ADE). The `modulated` option for HBAC and other modulated parameters are set by ADE. HBAC analyses with this option produce results that can have limited use outside of ADE. Direct Plot is configured to analyze these results and combine several wave forms to measure AM and PM response due to single sideband or modulated stimuli. For details, see the *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide*.

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

HBAC Synopsis

Name hbac <*parameter=value*> ...

HBAC Parameters

For information on hbac analysis parameters, refer to the *HB AC Analysis (hbac)* section in the Spectre Circuit Simulator Reference manual.

Details about Using HBAC Analysis Parameters

freqaxis Parameter

With HBAC, the frequency of the stimulus and of the response are usually different—this is one important way that HBAC differs from AC. The `freqaxis` parameter is used to specify whether the results should be output versus the input frequency (`in`), the output frequency (`out`), or the absolute value of the output frequency (`absout`).

maxsideband and sidevec Parameters

You can select the set of periodic small-signal output frequencies of interest by setting either the `maxsideband` or the `sidevec` parameters.

When there is only **one tone in HB analysis**, sidebands are n integer numbers, K1, K2, ..., Kn, and the output frequency at each sideband is computed as

```
f(out)= f(in) + Ki * fund(hb)
```

where `f(in)` represents the (possibly swept) input frequency, and `fund(hb)` represents the fundamental frequency used in the corresponding HB analysis. Thus, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is Ki= -1. When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is Ki= 1. By setting the `maxsideband` value to Kmax, all 2 * Kmax + 1 sidebands from -Kmax to +Kmax are generated.

When there are **multiple tones in HB analysis**, sidebands are vectors. Assume we have one large tone and one moderate tone in HB. A sideband, K1, is represented as [K1_1 K1_2]. The corresponding frequency is

```
K1_1 * fund(large tone of HB) + K1_2 * fund(moderate tone of HB)
```

We assume that there are L large and moderate tones in HB analysis and a given set of n integer vectors representing the sidebands, K1 = { K1_1, ..., K1_j, ..., K1_L}, K2, ..., Kn. The output frequency at each sideband is computed as

```
f(out)= f(in) + SUM_j=1_to_L{Ki_j * fund_j(hb)}
```

where `f(in)` represents the (possibly swept) input frequency, and `fund_j(hb)` represents the fundamental frequency used in the corresponding HB analysis. Thus, when analyzing a down-converting mixer, while sweeping the RF input frequency, the most relevant sideband for IF output is {-1, 0}. When simulating an up-converting mixer, while sweeping IF input frequency, the most relevant sideband for RF output is {1, 0}. You enter `sidevec` as a sequence of integer numbers, separated by spaces. The set of vectors {1 1 0} {1 -1 0} {1 1 1} becomes `sidevec=[1 1 0 1 -1 0 1 1 1]`. For `maxsideband`, only the large tone, the

first fundamental, is affected by this entry. All the other tones, the moderate tones, are limited by `maxharms`, specified for a HB analysis. Given maxharms=[k1max k2max ... knmax] in HB and maxsideband=Kmax, all

```
(2*Kmax + 1)*(2*k2max+1)*(2*k3max+1) *...*(2*knmax+1)
```

sidebands are generated.

The number of requested sidebands has a significant impact on the simulation time.

Sweep Interval Parameters

You can specify sweep limits by giving the end points or by providing the center value and the span of the sweep. Steps can be linear or logarithmic, and you can 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. All frequencies are in Hertz.

Harmonic Balance Noise Analysis (HBnoise)

The harmonic balance periodic or quasi-periodic noise (HBnoise) analysis is similar to conventional noise analysis, except that HBnoise analysis includes frequency conversion effects. As a consequence, it is useful for predicting the noise behavior of mixers, switched-capacitor filters, and other periodically driven circuits. It is particularly useful for predicting the phase noise of autonomous circuits, such as oscillators.

HBnoise analysis linearizes the circuit about the periodic or quasi-periodic operating point computed in the prerequisite HB analysis. It is the periodically or quasi-periodically time-varying nature of the linearized circuit that accounts for the frequency conversion. In addition, the effect of a periodically or quasi-periodically time-varying bias point on the noise generated by the various components in the circuit is also included.

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.

The noise analysis always computes the total noise at the output, including contributions from the input source 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, which includes the noise from the input source itself, is computed. Finally, if the input source is identified (using `iprobe`) and is noisy, as is the case with ports, the noise factor and noise figure are computed. Thus if

`No` = total output noise
`Ns` = noise at the output due to the input probe (the source)
`Nsi` = noise at the output due to the image harmonic at the source
`Nso` = noise at the output due to harmonics other than input at the source
`Nl` = noise at the output due to the output probe (the load)
`IRN` = input referred noise
`G` = gain of the circuit
`F` = noise factor
`NF` = noise figure
`Fdsb` = double sideband noise factor
`NFdsb` = double sideband noise figure
`Fieee` = IEEE single sideband noise factor
`NFieee` = IEEE single sideband noise figure

then,

```
IRN = sqrt(No^2/G^2)
F = (No^2 - Nl^2)/Ns^2
NF = 10*log10(F)
Fdsb = (No^2 - Nl^2)/(Ns^2+Nsi^2)
NFdsb = 10*log10(Fdsb)
Fieee = (No^2 - Nl^2 - Nso^2)/Ns^2
NFieee = 10*log10(Fieee)
```

When the results are output, `No` is named `out`, `IRN` is named `in`, `G` is named `gain`, `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee` are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee` respectively.

The computation of gain and IRN for quasi-periodic noise in HBnoise assumes that the circuit under test is impedance-matched to the input source. This can introduce inaccuracy into the gain and IRN computation.

An HBnoise analysis must follow an HB analysis.

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

HBnoise Synopsis

Name ([*p*] [*n*] ...) hbnoise <*parameter=value*> ...

The optional terminals (*p* and *n*) specify the output of the circuit. If you do not give the terminals, then you must specify the output with a probe component.

HBnoise Parameters

For information on hbnoise parameters, refer to the *HB Noise Analysis (hbnoise)* section in the *Spectre Circuit Simulator Reference* manual.

Details about Using HBnoise Analysis Parameters

iprobe Parameter

If you want the input-referred noise or noise figure, specify the input source using the `iprobe` parameter. For input-referred noise, use either a vsource or isource as the input probe; for noise figure, use a port as the probe. 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 computes 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 its noisetemp must be set to 16.85C

(290K). In addition, the output load must be a resistor or port and must be identified as the oprobe.

If port is specified as the input probe, then both input-referred noise and gain are referred back to the equivalent voltage source inside the port. S-parameter analysis calculates those values.

maxsideband Parameter

In practice, noise can mix with each of the harmonics of the periodic drive signal applied in the HB analysis and end up at the output frequency. However, the HBnoise analysis only includes the noise that mixes with a finite set of harmonics that are typically specified using the `maxsideband` parameter.

If Ki represents sideband i, then for periodic noise,

```
f(noise_source) = f(out) + Ki * fund(hb)
```

For quasi-periodic noise with multiple tones in HB analysis, assuming there are one large tone and one moderate tone, Ki is represented as [Ki_1 Ki_2]. The corresponding frequency shift is

```
Ki_1 * fund(large tone of HB) + Ki_2 * fund(moderate tone of HB)
```

Assuming that there are L large and moderate tones in HB analysis and a set of n integer vectors representing the sidebands

```
K1 = { K1_1,...K1_j..., K1_L}, K2, ..., Kn.
```

Then

```
f(noise_source) = f(out) +  SUM_j=1_to_L{ Ki_j * fund_j(hb) }
```

The `maxsideband` parameter specifies the maximum |Ki| included in the HBnoise calculation. For quasi-periodic noise, only the large tone, the first fundamental, is affected by this entry. All the other tones, the moderate tones, are limited by `maxharms`, specified for a HB analysis.

The number of requested sidebands changes the simulation time substantially.

When HBnoise analysis does only an xf analysis (`xfonly=yes`), the variable of interest at the output can be voltage or current, and its frequency is not constrained by the period of the large periodic solution. While sweeping the selected output frequency, you can select the periodic small-signal input frequencies of interest by setting the `maxsideband` parameter. With this analysis, the frequency of the stimulus and of the response are usually different (this is an important way that this analysis differs from XF).

refsideband Parameter

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

```
|f(input)| = |f(out) + refsideband frequency shift|
```

For periodic noise (only **one tone in HB analysis**), the `refsideband` is a number. Use `refsideband=0` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters). When `refsideband` differs from 0, the single side-band noise figure is computed.

While for quasi-periodic noise (**multiple tones in HB analysis**), reference sidebands are vectors. Assume we have one large tone and one moderate tone in HB. A sideband Ki will be a vector [Ki_1 Ki_2]. It gives the frequency at

```
Ki_1 * fund(large tone of HB) + Ki_2 * fund(moderate tone of HB)
```

Use `refsideband=[0 0 ...]` when the input and output of the circuit are at the same frequency (such as with amplifiers and filters).

stimuli Parameter

You can use the `stimuli` parameter to specify what serves as the inputs for the transfer functions.

- `stimuli=sources` indicates that the sources present in the circuit are to be used. You can use the `xfmag` parameters provided by the sources to adjust the computed gain to compensate for gains or losses in a test fixture. You can limit the number of sources in hierarchical netlists by using the `save` and `nestlvl` parameters.

- `stimuli=nodes_and_terminals` indicates that all possible transfer functions are to be computed. This is useful when you do not know in advance which transfer functions are 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 from a small set of terminals are computed. If you want transfer functions from specific terminals, specify the terminals in the `save` statement. You must use the `:probe` modifier (for example, `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.

With HBnoise analysis, transfer function outputs are always available.

Sweep Interval Parameters

You can specify sweep limits by giving the end points or by providing the center value and the span of the sweep. Steps can be linear or logarithmic, and you can 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. All frequencies are in Hertz.

xfonly Parameter

When option `xfonly` is set to `yes`, HBnoise analysis does only a conventional transfer function (xf) analysis, which computes the transfer function from every source in the circuit to a single output. This analysis differs from a conventional AC analysis in that the AC analysis computes the response from a single stimulus to every node in the circuit. HBnoise computes the transfer functions from any source at any frequency to a single output at a single frequency. Thus, like HBAC analysis, it includes frequency conversion effects. It directly computes such useful quantities as conversion efficiency (transfer function from input to output at desired frequency), image and sideband rejection (input to output at undesired frequency), and LO feed-through and power supply rejection (undesired input to output at all frequencies).

HB S-Parameter Analysis (hbsp)

The periodic or quasi-periodic SP (HBSP) analysis is used to compute scattering and noise parameters for n-port circuits such as mixers that exhibit frequency translation. It is a small-signal analysis similar to SP analysis, except that in HBAC and HBNOISE, the circuit is first linearized about a periodically varying operating point as opposed to a simple DC operating point. Linearizing about a periodically or quasi-periodically time-varying operating point allows the computation of S-parameters between circuit ports that convert signals from one frequency band to another. HBSP can also calculate noise parameters in frequency-converting circuits. In addition, HBSP computes noise figure (both single-sideband and double-sideband), input referred noise, equivalent noise parameters, and noise correlation matrices. Similar to HBNOISE, but unlike SP, the noise features of the HBSP analysis include noise folding effects due to the periodic 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 or quasi-periodic steady-state response of the circuit to possibly large periodic stimulus is computed using HB analysis. As a part of the HB analysis, the periodically time-varying representation of the circuit is computed and saved for later use. The second step is applying small-signal excitations to compute the n-port S-parameters and noise parameters. This is done using the HBSP analysis. HBSP analysis cannot be used independently; it must follow HB analysis. However, any number of periodic small-signal analyses such as HBAC, HBSP, HBNOISE, can follow an HB analysis.

Note: Unlike other analyses in Spectre, this analysis can only sweep frequency.

HBSP Parameters

For information on hbsp analysis parameters, refer to the *HB S-Parameter Analysis (hbsp)* section in the *Spectre Circuit Simulator Reference* manual.

To specify the HBSP analysis, the port and port harmonic relations must be specified. You can select the ports of interest by setting the `port` parameter, and select the set of periodic small-signal output frequencies of interest by setting `portharmsvec` or `harmsvec` parameters. For a given set of n integer numbers representing the harmonics K1, K2, ... Kn, the scattering parameters at each port are computed at the following frequencies:

For periodic SP in one-tone HB analysis, frequency is:

```
f(scattered)= f(rel) + Ki * fund(HB)
```

For quasi-periodic noise with multi-tone in HB analysis, sidebands are vectors. Consider that you have one large tone and one moderate tone in HB. Then, the above sideband K1 will be represented as [K1_1 K1_2]. In this case, the corresponding frequency is:

```
K1_1 * fund(large tone of HB) + K1_2 * fund(moderate tone of HB)=
SUM_j=1_to_L{Ki_j * fund_j(HB)}
```

If there are L (1 large and L-1 moderate) tones in HB analysis and a given set of n integer vectors representing the sidebands:

```
K1 = { K1_1,...K1_j..., K1_L } , K2, ... Kn
```

If you specify the relative frequency, the scattering parameters at each port are computed at the frequencies:

```
f(scattered)= f(rel) + SUM_j=1_to_L{Ki_j * fund_j(hb)},
```

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`(one-tone HB) or `fund_j`(multi-tone HB) represents the fundamental frequency used in the corresponding HB analysis.

During analysis of a down-converting mixer with a blocker and the signal in the upper sideband, we sweep the input frequency of the signal coming into RF port. In case of periodic SP with one-tone HB, the most relevant harmonic for RF input is Ki= 1 and for IF output Ki= 0. Therefore, we can associate K2=0 with the IF port and K1=1 with the RF port. S21 represents the transmission of signal from the RF to IF, and S11 represents the reflection of signal back to the RF port. If the signal was in the lower sideband, a choice of K1=-1 is more appropriate. For quasi-periodic SP with multi-tone HB, the most relevant sideband for this input is Ki= {1 , 0} and for IF output Ki= {0 , 0}. Therefore, we can associate K1={1 , 0} with the RF port and K2={0 , 0} with the IF port. If the signal was in the lower sideband, then a choice of K1={-1 , 0} is more appropriate.

`portharmsvec` or `harmsvec` parameters can be used to specify the harmonics of interest. If `portharmsvec` is specified, the harmonics must be in one-to-one correspondence with the ports, with each harmonic associated with a single port. If harmonics are specified in the optional `harmsvec` parameter, all possible frequency-translating scattering parameters associated with the specified harmonics are computed.

With HBSP the frequency of the input and of the response are usually different (this is an important area in which HBSP differs from SP). Because the HBSP computation involves inputs and outputs at frequencies that are relative to multiple harmonics or sidebands, the `freqaxis` and `sweeptype` parameters behave differently in HPSP than in HBAC and HBNOISE.

The `sweeptype` parameter controls the way the frequencies in the HBSP analysis are swept. Specifying a `relative` sweep indicates the sweep to be relative to the analysis harmonics or port sideband (not the HB fundamental) and specifying an `absolute` sweep indicates the sweep of the absolute input source frequency.

For example, in case of periodic SP with one-tone HB and HB fundamental of 100MHz, `portharmsvec` set to [9 1] to examine a downconverting mixer, `sweeptype=relative`, and a sweep range of f(rel)=0->50MHz, S21 represents the strength of signal transmitted from the input port in the range 900->950MHz to the output port at frequencies 100->150MHz. Using `sweeptype=absolute` and sweeping the frequency from 900->950MHz would calculate the same quantities, because f(abs)=900->950MHz, f(rel) = f(abs) - K1 * fund(hb) = 0->50MHz, and K1=9 and fund(hb) = 100MHz.

For quasi-periodic noise with multi-tone HB, and HB fundamentals of 1000MHz (LO) and 966MHz (blocker in RF channel), `portharmsvec` could be set to [0 1 -1 1] to examine a downconverting mixer. Consider setting `sweeptype=relative` and a sweep range of f(rel)=-10MHz<->10MHz. Then, S21 will represent the strength of the signal transmitted from the input port in the range 956->976MHz to the output port at frequencies 24<->44MHz. Using `sweeptype=absolute` and sweeping the frequency from 966<->976MHz will calculate the same quantities, because f(abs)=956<->976MHz, f(rel) = f(abs) - (K1_1 * fund_1(hb) + K1_2 * fund_2(hb) = -10MHz<->10MHz, andK1_1=0, K1_2=1 and fund_1(hb) = 1000MHz, fund_2(hb) = 966MHz.

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`).

HBSP analysis also computes noise figures, equivalent noise sources, and noise parameters. The noise computation, which is skipped only when `donoise=no`, requires additional simulation time. If:

No = total output noise at frequency f

Ns = noise at the output due to the input probe (the source)

Nsi = noise at the output due to the image harmonic at the source

Nso = noise at the output due to harmonics other than input at the source

Nl = noise at the output due to the output probe (the load)

IRN = input referred noise

G = gain of the circuit

F = noise factor (single side band)

NF = noise figure (single side band)

Fdsb = double sideband noise factor

NFdsb = double sideband noise figure

Fieee = IEEE single sideband noise factor

NFieee = IEEE single sideband noise figure

Then:

IRN = sqrt(No^2/G^2)

F = (No^2 - NI^2)/Ns^2

NF = 10*log10(F)

Fdsb = (No^2 - NI^2)/(Ns^2+Nsi^2)

NFdsb = 10*log10(Fdsb)

Fieee = (No^2 - NI^2 - Nso^2)/Ns^2

NFieee = 10*log10(Fieee).

When the results are output, IRN is named `in`, G is named `gain`, F, NF, Fdsb, NFdsb, Fieee, and NFieee are named `F`, `NF`, `Fdsb`, `NFdsb`, `Fieee`, and `NFieee`, respectively. Note that the gain computed by HBSP 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.

To ensure accurate noise calculations, the `maxsideband` or `sidebands` parameters must be set to include the relevant noise folding effects. `maxsideband` is only relevant to the noise computation features of HBSP.

You can specify sweep limits by giving the end points or by providing the center value and span of the sweep. Steps can be linear or logarithmic, and you can specify the number of steps or the size of each step. In addition, 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, and logarithmic when this ratio is 10 or greater. Alternatively, you use the `values` parameter to specify the values that the sweep parameter should take. If you use 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.

Envelope Analysis (ENVLP)

Efficient and accurate prediction of the envelope transient response of RF circuits is important for RF circuit designers who are simulating communications systems. You can apply ENVLP analysis to efficiently and accurately analyze modulation signals in large communication circuits. Important applications include

■ Predicting the spectral regrowth of amplifiers and mixers

■ Designing feedback loops such as Automatic Gain Control (AGC) loops

■ Predicting the transient behavior of switched capacitor filters

■ Simulating large transients in phase lock loops

■ Helping the oscillator designer identify the load pull effect for the communication systems with VCO and power amplifier.

It is important to know that ENVLP Analysis is *not* designed to simulate circuits having a filter with nodes that have higher frequencies than clock. A transient analysis is faster for these circuits.

Modulation Signals

Many RF circuits process narrowband signals in the form of modulated carriers. Modulated carriers are characterized as having both a periodic high-frequency carrier signal and a low-frequency modulation signal. The modulation signal acts on either the amplitude, phase, or frequency of the carrier. In general, the modulation is arbitrary. The ratio between the lowest frequency in the modulation and the frequency of the carrier is a measure of the relative frequency resolution required of the simulation. Traditional transient analysis is inefficient for the resolution of low modulation frequencies in the presence of a high carrier frequency because the high-frequency carrier forces a small time step while the low-frequency modulation forces a long simulation interval.

A Mixer Example

As a typical example, a designer might be interested in simulating a receiver transmit path involving a modulator, in particular, to predict the spectral regrowth of the modulator. As shown in Figure 2-10, the inputs to a modulator can be one complete digital low-frequency (not necessarily periodic) modulation and one high frequency LO. The result is a modulated high-frequency signal as shown in Figure 2-10. However, due to the nonlinearity of the modulator, unwanted harmonics might be generated and, to validate the design, it is important to predict the signal level at these unwanted harmonics.

Figure 2-10 Time-Domain Modulation
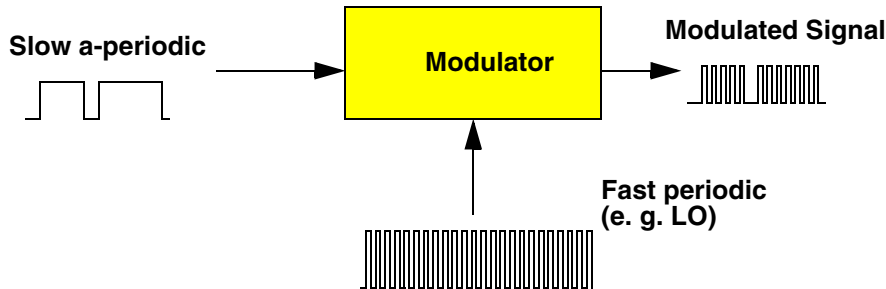
Figure 2-11 shows a typical scenario in the receiver signal path. Due to the nonlinearity of the mixer, it is important to predict the resulting spectral regrowth. Spectral regrowth is expensive to simulate using traditional transient analysis because spectral regrowth requires a very long time interval to resolve the required frequency resolution.

Figure 2-11 Spectrum-Domain Modulation
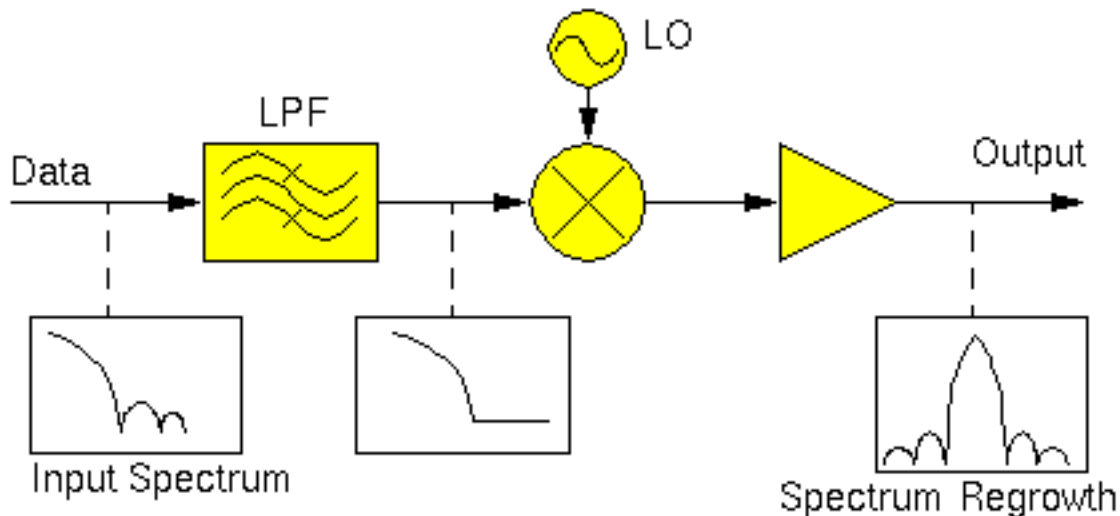
ENVLP analysis overcomes this difficulty with traditional transient analysis. ENVLP analysis reduces simulation time without compromising accuracy by exploiting the property that the behavior of a circuit in a given high frequency clock cycle is similar, but not identical, to its behavior in the preceding and following cycles. In particular, the envelope of the high-frequency clock can be followed by accurately computing the circuit behavior over occasional cycles, which accurately capture the fast transient behavior. The slow varying modulation is accurately followed by a smooth curve. As a result, the spectrum of the circuit response can be obtained by combining the spectrum of the smooth curve and the spectrum of occasional clock cycles.

RF analyses such as PSS (periodic steady-state analysis) [1] or QPSS (quasi-periodic steady-state analysis) [2] might not work directly because the modulation signal might be neither periodic nor quasi-periodic.

The clock is referred to differently in different applications.

■   For mixers, the clock is called the LO

■   For detectors, the clock is called the carrier

■   For switched-capacitor filters, the clock is called the clock

The clock is normally the most rapidly changing signal in the circuit and thus causes the most nonlinearity.

The Time Domain and Harmonic Balance ENVLP Algorithms

The Spectre RF ENVLP analysis uses a multi-stage multi-past-point integration algorithm that is an extension to a method introduced by Petzold [3] and further explored by Kundert, White and Sangiovanni-Vincentelli [4]. The method approximates the sample envelope as a piecewise polynomial in a manner that is analogous to conventional transient analysis. The ENVLP algorithm is based on two schemes:

■   Time domain shooting

■   Harmonic balance

In time domain shooting, the clock nonlinearity is resolved by time-domain integration. In harmonic balance, the clock nonlinearity is expressed as harmonics of fundamental frequencies.

Most RF circuits used in communication systems are clocked at a high frequency. The clock (such as LO) usually causes the most nonlinearity in the circuit response. Time-domain integration is a more efficient and accurate method of resolving strong nonlinear circuits with sharp transitions and transient details, while harmonic balance is more efficient for linear and nearly linear circuits. ENVLP analysis samples the circuit waveforms at the clock frequency, and assumes the resulting envelope can be accurately represented by a piecewise polynomial. The nonlinearity caused by the clock signal is resolved by occasional integrations of a period of circuit responses.

Time Domain Envelope (TD ENVLP) Analysis

Most circuits can be described by a system of differential equations of the form shown in Equation 2-1.

(2-1) $\quad \dfrac{d}{dt}q(v(t)) + i(v(t)) \ = \ u(t)$

where

■ $u(t) \in \Re^M$ is the vector of input sources

■ $v(t) \in \Re^N$ (the state) is the node voltages

■ $q(v(t)) \in \Re^N$ is the vector of node charges and fluxes

■ $i(v(t)) \in \Re^N$ is the vector of resistive node currents

If the state v is known at some time t_0, it is possible to solve Equation 2-1 and compute the state at some later time t_1. In general, you can write Equation 2-2.

(2-2) $\quad v(t_1) \ = \ \phi(v(t_0), t_0, t_1)$

where $\phi : \Re^N \times \Re^N \times \Re^N \rightarrow \Re^N$ is a state transition function for the differential equation.

Consider that the circuit to be simulated has an input clock with a known period T that is much smaller than the simulation interval. Assume the sequence formed by sampling the state at the beginning of each clock cycle, $v(0)$, $v(T)$, $v(2T)$, ..., $v(mT)$, ..., changes slowly as a function of m, the clock cycle number. A smooth continuous function can be defined to interpolate the sequence formed by sampling the state at every time interval T.

The waveform shown in Figure 2-12, is sampled at the clock period, resulting in a circuit envelope that reveals the slow varying modulation.

Figure 2-12 A Sample Envelope
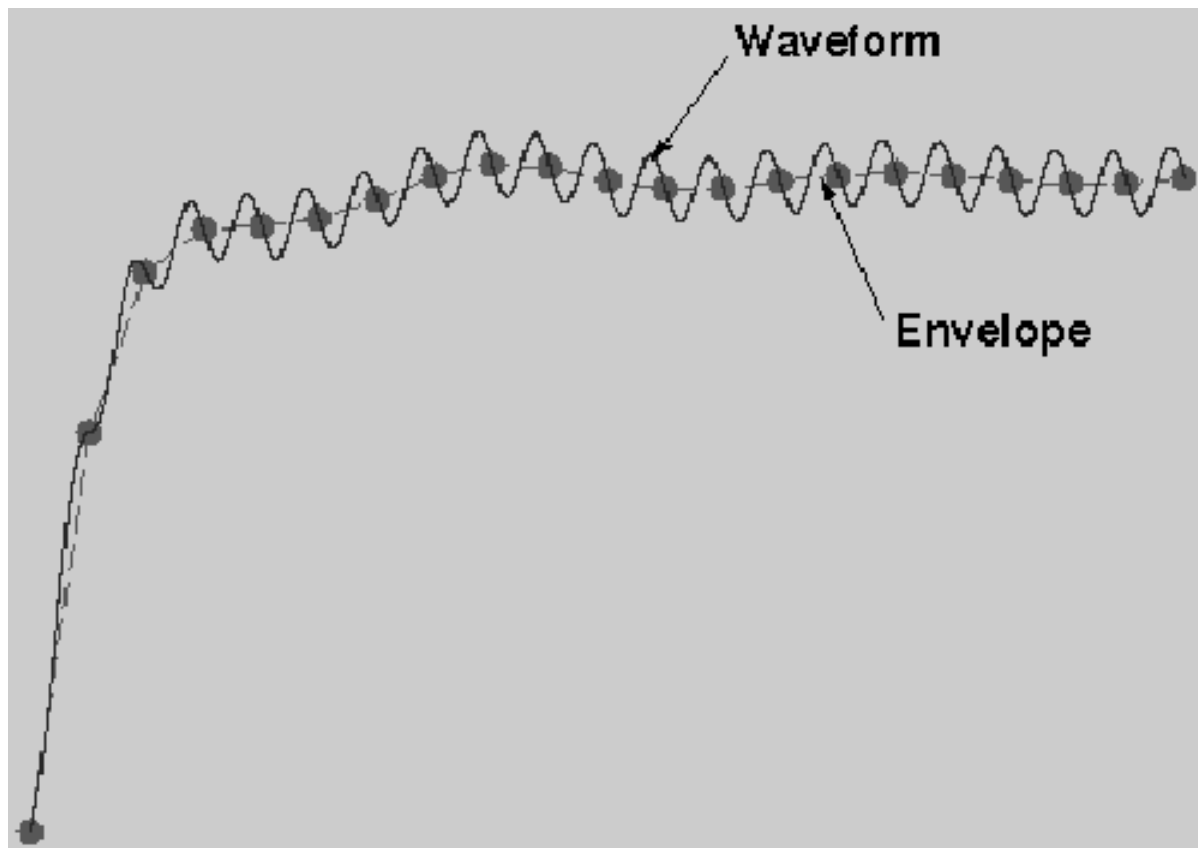
Figure 2-13 illustrates how the TD ENVLP (time domain envelope) algorithm works.

Figure 2-13 TD ENVLP Integration
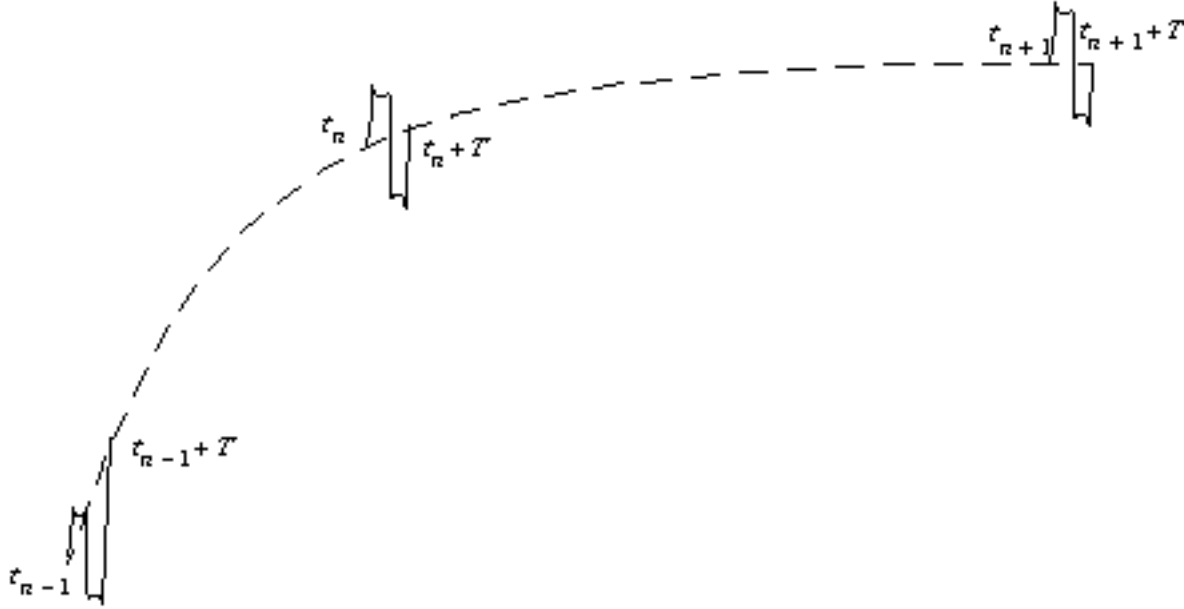
TD ENVLP integration is similar to transient integration. In this example, two past stages are used to compute a new stage. Assume the state values are known at $t_{(n-1)}$ and $t_{(n)}$, which are a number of clock cycles apart from each other, and are at the beginning of the clock cycle. The state values at $t_{(n-1)}$ and $t_{(n-1)}+T$ are related by integration of one clock cycle, that is,

(2-3) $v(t_{(n-1)} + T) = \phi(v(t_{(n-1)}), t_{(n-1)}, t_{(n-1)} + T)$

where ϕ is the state transition function.

Likewise, the states at t_n and t_n+T are related by $v(t_n+T) = \phi(v(t_n), t_n, t_n+T)$. The job of the algorithm is to find the state value at a new time point $t_{(n+1)}$ many cycles from t_n, such that the pair $v(t_{n+1})$, $t_{(n+1)}$ interpolates a quadratic polynomial defined by the three state values $v(t_{(n-1)})$, $v(t_n)$ and $v(t_{(n+1)}+T)$ at $t_{(n-1)}$, t_n and $t_{(n+1)}+T$, respectively.

On one hand, the state value $v(t_{(n+1)}+T)$ is given by

(2-4) $v(t_{n+1} + T) = \phi(v(t_{n+1}), t_{n+1}, t_{n+1} + T)$

On the other hand, from the interpolation condition, $v(t_{(n+1)}+T)$ can be written as a linear combination of $v(t_{(n-1)})$, $v(t_n)$ and $v(t_{n+1})$, i.e., there exist scalars α_1, α_2 and α_3 such that

(2-5) $\quad v(t_{n+1}+T) = \alpha_1 v(t_{n-1}) + \alpha_2 v(t_n) + \alpha_3 v(t_{n+1})$

The scalars α_1, α_2 and α_3 can be obtained by standard techniques such as by first defining a quadratic function using three state values and then asking the fourth value to be interpolated by the quadratic function. Combining the two equations gives

(2-6) $\quad \alpha_1 v(t_{(n-1)}) + \alpha_2 v(t_n) + \alpha_3 v(t_{n+1}) - \phi(v(t_{n+1}), t_{n+1}, t_{n+1}+T) = 0$

which can be used to solve for $v(t_{n+1})$ by shooting Newton method.

After $v(t_{n+1})$ is obtained, we use t_n and $t_{(n+1)}$ as past points and solve for a new stage $v(t_{n+2})$. The two-past-point one-new-stage process is repeated until the desired stop time is reached.

Harmonic Balance Envelope (HB ENVLP) Analysis

A general signal in a modulation system has the form

(2-7) $\quad x(t) = \sum_k \tilde{X}_k(t) e^{j2\pi f_k t}$

when the Fourier coefficients

$\tilde{X}_k(t)$

are taken to be slowly varying transient waveforms.

$\tilde{X}_k(t)$

must vary slowly relative to f_k, because when the bandwidth of

$\tilde{X}_k$

is greater than $f_k/2$, the sidebands of adjacent harmonics begin to overlap and the representation is not unique. If there is an f_0 that satisfies

(2-8) $\quad f_k = kf_0$

then $x(t)$ is a one tone signal and the corresponding analysis is a single carrier HB envelope. Otherwise, $x(t)$ is a multi-tone signal and the corresponding analysis is a multi-carrier HB envelope.

Rewriting Equation 1-1 on page 30 in the form of Equation 2-7 on page 282 results in the following.

(2-9) $\quad \displaystyle\sum_k \left(\frac{d\tilde{Q}_k(\tilde{U}(t))}{dt} + j2\pi f_k \tilde{Q}_k(\tilde{U}(t)) + \tilde{I}_k(\tilde{U}(t)) - \tilde{S}_k(t) \right) = 0$

In vector form, this can be written as

(2-10) $\quad \dfrac{d\tilde{Q}(\tilde{U}(t))}{dt} + \Omega\tilde{Q}(\tilde{U}(t)) + \tilde{I}(\tilde{U}(t)) - \tilde{S}(t) = 0$

where Ω is a diagonal matrix with $j2\pi f_k$ on the kth diagonal. As the transient analysis does, an HB ENVLP analysis discretizes

$$\frac{d\tilde{Q}}{dt}$$

with a finite-difference approximation such as the backward-Euler, trapezoidal, or Gear method. For example, applying backward-Euler, Equation 2-9 on page 283 becomes

(2-11) $\quad \dfrac{\tilde{Q}(\tilde{U}(t_{n+1})) - \tilde{Q}(\tilde{U}(t_n))}{t_{n+1} - t_n} + \Omega\tilde{Q}(\tilde{U}(t_{n+1})) + \tilde{I}(\tilde{U}(t_{n+1})) - \tilde{S}(t_{n+1}) = 0$

Equation 2-11 on page 283 is a system of nonlinear algebraic equations.

$$\tilde{U}(t_{n+1})$$

can be solved by Newton's method. After that value is obtained, it is used to get

$$\tilde{U}(t_{n+2})$$

until the desired stop time is reached.

HB ENVLP analysis samples the modulation envelope in time and outputs a time-varying spectrum for time point t_1, t_2, ..., t_n, ... The spectrum is converted to time domain waveforms in each clock cycle.

In the first several clock cycles, a transient analysis is used in single carrier HB Envelope, as it is in TD Envelope. For multi-carrier HB Envelope, HB QPSS is adopted to get the steady state at the beginning.Then the time domain data is converted to frequency domain and Equation 2-11 on page 283 works. HB Envelope is like a special Transient for frequency data.

Figure 2-14 HB Envelope Analysis outputs a time-varying spectrum
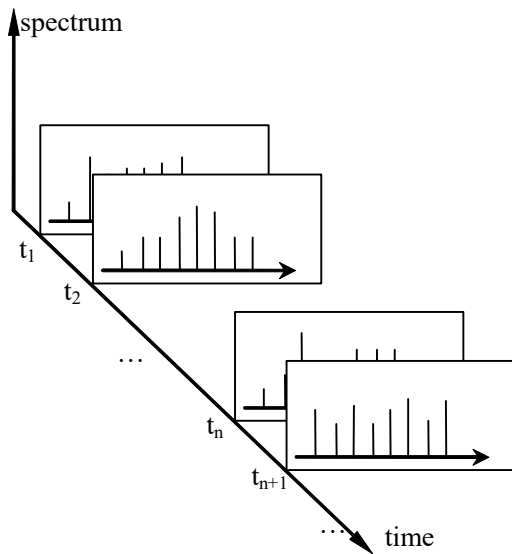
For linear and nearly linear circuits, HB Envelope is more efficient than TD Envelope analysis.

Both TD and HB Envelope analysis use Newton's method. In a manner similar to that used for Periodic Steady-State (PSS) analysis (for details see [1]), the Newton equation is solved efficiently by a matrix-implicit iterative method. Compared to harmonic balance-based envelope following algorithms, the Spectre RF Envelope Following algorithm has advantages and weaknesses similar to Spectre RF PSS analysis versus analogous harmonic balance-based steady-state computation approaches.

In TD and single carrier HB Envelope analysis, the clock is pointed directly. In multi-carrier HB Envelope analysis, the first one of the fundamentals is regarded as the clock. The period of the clock is the time unit of the envelope analyses.

Both TD and HB Envelope analysis attempt to skip as many clock cycles as possible to achieve a faster speed than Transient analysis. Similar to Transient analysis, after a sample point is computed, it is necessary to check that the trajectory is following the low-order polynomial as assumed. If it is not, the point is discarded and the time step is reduced. If rapid changes in the envelope are encountered, Envelope analysis reduces its step size down to the point where no cycles are skipped. In this case, both TD Envelope analysis and single carrier HB Envelope analysis degenerate to a simple Transient analysis. The frequency domain data are converted to time domain. After the simple Transient, the data are converted back to frequency domain.

Generally, HB Envelope is faster than TD Envelope for linear or weakly non-linear circuits. However, HB Envelope suffers from accuracy problems when dealing with strongly non-linear circuits.

HB Parameters

Parameters required by the HB ENVLP analysis are:

Basic Parameters for HB ENVLP Analysis

`flexbalance=yes`	The HB engine shares this analysis form with the TD engine. Use the HB engine by setting `flexbalance=yes` in the analysis statement. The default is `no`.

`harms`
or
`maxharms`
	Use of the `harms` or `maxharms` parameter is similar to the usage in driven HB analysis. For more information, see <u>Basic HB Parameters for Driven HB Analysis</u> table on page 35.

Optional Parameters for HB ENVLP Analysis

`oversamplefactor`	Use of the `oversamplefactor` parameter is similar to its usage in driven HB analysis. For more information, see <u>Optional HB Parameters for Driven HB Analysis</u> table on page 36.

Using ENVLP Analysis

This analysis computes the envelope response of a circuit. The user specifies the analysis **clockname**. The simulator automatically determines the clock period by looking through all the sources with the specified name. The envelope response is computed over the interval from **start** to **stop**. If the interval is not a multiple of the clock period, it is rounded off to the nearest multiple before the stop time. The initial condition is taken to be the DC steady-state solution or determined by **tstab** if not otherwise given. If **flexbalance** is `yes`, HB ENVLP will be used, otherwise TD ENVLP will be used. The default value is `no`.

As discussed previously, the clock might be referred to differently in different applications LO with mixers, carrier with detectors, and clock with switched-capacitor filters. The clock is normally the most rapidly changing signal in the circuit and thus causes the most nonlinearity.

ENVLP analysis is most efficient for circuits where the modulation bandwidth is orders of magnitude lower than the clock frequency. This is typically the case in RF circuits. Passing a narrowband signal through a nonlinear RF circuit results in a broadband signal whose spectrum is relatively sparse, as shown in Figure 2-15. The spectrum shows that due to nonlinearity, narrowband responses centered at the carrier harmonics are generated.

Figure 2-15 Spectrum of a Narrowband Signal
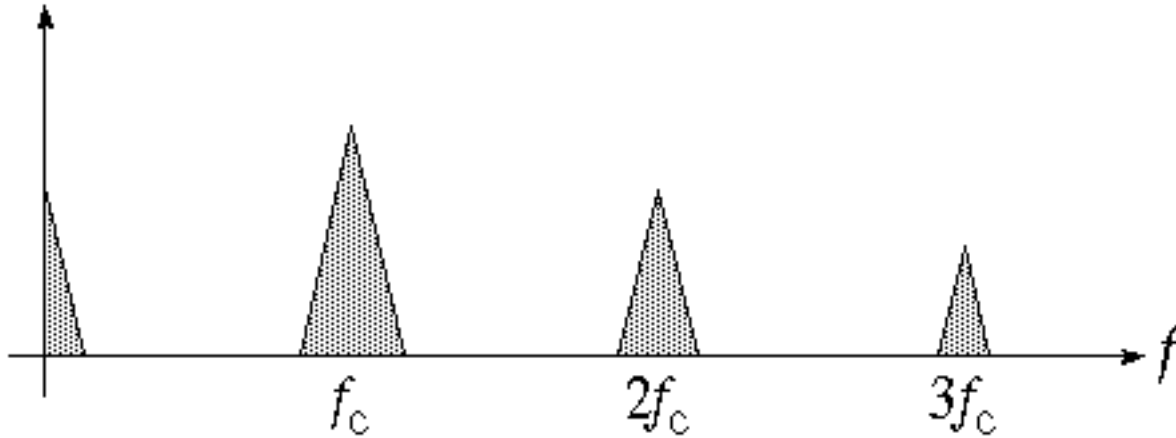
In general, this spectrum consists of clusters of frequencies near the harmonics of the carrier. When the modulation is periodic or quasi-periodic, these clusters take the form of a discrete set of frequencies. Otherwise they form a continuous distribution of frequencies.

Figures 2-16 and 2-17 present examples of calculating spectrum regrowth. Here the carrier is a periodic high-frequency signal, while the modulation is a low frequency arbitrary digital modulation. ENVLP analysis is used to accurately calculate the sparse spectrums centered at clock harmonics. A continuous spectrum in each cluster is expected because the modulation is neither periodic nor quasi-periodic.

In two typical situations, ENVLP analysis is very efficient and effective in simulating the transient behavior of circuits driven by two periodic signals.

In the first situation, one fundamental frequency is much higher than the other one, as would be the case for an up-conversion mixer. As shown in Figure 2-16, you can interpret a quasiperiodic signal with two widely separated fundamentals as a periodically modulated periodic signal. To do this, designate the high frequency signal as the carrier and the low frequency signal as the modulation. If the carrier is much higher in frequency than the modulation, then the carrier will appear to vary only slightly from cycle to cycle. When this is the case, the envelope can be efficiently followed with ENVLP analysis.

Figure 2-16 A Two-Fundamental Quasiperiodic Signal
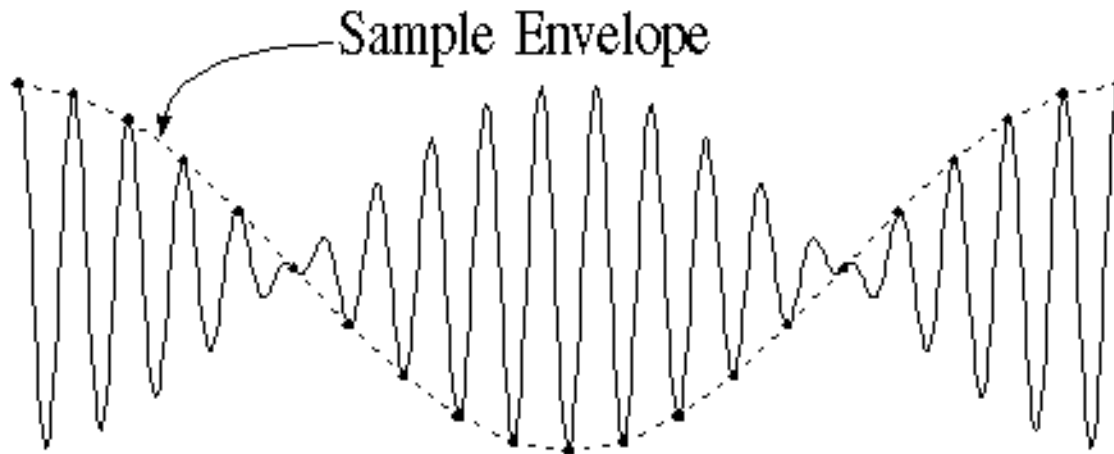
In the second situation, both inputs are high frequency signals but their frequencies are close to each other. The down-conversion of these closely placed frequencies can generate a slow-varying modulation envelope whose frequency is orders of magnitude lower than the input frequencies. For example, Figure 2-17 shows a down-conversion mixer where TD ENVLP analysis can be used to trace out the modulation envelope by choosing either of the fast varying signals as the clock. In general, you should choose as the clock signal the signal that causes the most nonlinearity. Be aware that single carrier HB ENVLP analysis does not handle this well because HB ENVLP treats signals other than the clock signal as DC signals in one cycle of the clock. Multi-carrier HB ENVLP can handle this case.

Figure 2-17 Down Conversion of Two Closely Placed Frequencies
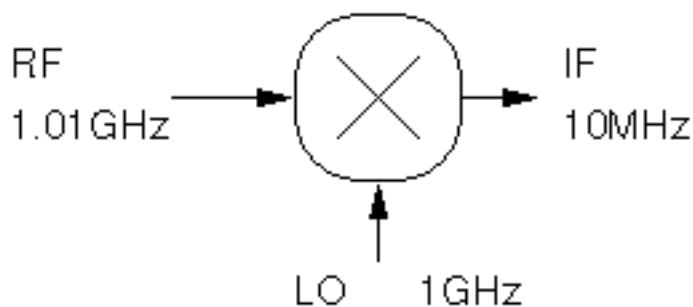
As is true for Periodic Steady-State (PSS) analysis, you can use ENVLP analysis for particular classes of circuits operating with multiple clock fundamentals. For PSS analysis, the multiple fundamentals are commensurate. For ENVLP analysis, you can use the greatest

common denominator of all fundamental clock frequencies as the clock beat frequency. The process of selecting the clock beat frequency for ENVLP analysis is similar to the process of figuring out the beat frequency for PSS analysis. In similar situations where multiple clock fundamentals are present, ENVLP analysis is also analogous to PSS in terms of efficiency. Each time the clock period is integrated, a large number of fast cycles might be involved. The efficiency of the method depends on how many fast cycles occur per beat period. The more fast cycles you need to simulate, the less efficient the method. Because HB ENVLP regards the signals other than clock as DC signals, currently HB ENVLP is *not* a good choice to simulate a circuit with fast cycles other than clock. Multi-carrier HB Envelope should be used instead by either

■   Setting LO and RF as the fundamentals, and regarding IF as slow signals.

■   Regarding IF as a modulated wave, and setting LO, RF and IF as the fundamentals.

One difference between the ENVLP and PSS analyses is that for ENVLP analysis, you use a single `clockname` to identify the fundamentals. The simulator automatically calculates the beat frequency. For PSS analysis, you figure out the beat frequency either at the UI level or by entering it as an analysis parameter.

In a certain sense, an ENVLP analysis might be considered as a fast transient analysis whose efficiency comes from skipping simulation cycles. However, when an ENVLP analysis cannot find cycles to skip, the analysis effectively reduces to a transient analysis. Therefore, ENVLP analysis might not always be more efficient than transient analysis due to the greater computational overhead required for ENVLP analysis when it cannot find enough cycles to skip. Nevertheless, ENVLP analysis is always as accurate as PSS analysis.

ENVLP Parameters

For information on ENVLP analysis parameters, refer to the *Envelope Following Analysis (envlp)* section in the Spectre Circuit Simulator Reference manual.

The Envelope Choose Analysis form is similar to the transient Choose Analysis form. Most ENVLP analysis parameters are inherited from either transient or PSS analysis and their meanings are consistent. However, a few parameters need clarification.

The procedure for setting up an ENVLP analysis is similar to the set-up procedure for transient analysis. Important parameters for ENVLP analysis include: `clockname`, `period`, `fund`, `funds`, `flexbalance`, `maxharms`, `harmonicbalance`, `harms`, `stop`, `envmaxstep`, `modulationbw`, `fixstepsize`, `swapfile`, and `fmspeedup`.

The following list summarizes several important differences between ENVLP analysis and transient analysis. The parameter names are those used inside Spectre RF. For ENVLP analysis, do the following:

■ Identify a periodic, built-in source with the `clockname` parameter. This requirement is similar to QPSS analysis requirements. However, for ENVLP analysis, you need only name the clock source. You can select any periodic, built-in source (`sine`, `pulse`, or `pwl`) as the clock. The simulator examines all the sources whose name matches the `clockname` to determine the clock frequency. If more than one frequency is found, the greatest common factor of these frequencies is used as the clock frequency.

When applied to autonomous circuits, ENVLP analysis requires that you specify a pair of nodes, `p` and `n`, as for PSS analysis. You cannot use `clockname` for autonomous circuits. If the period of the carrier is known, `period` of `fund` could be set to get the clock frequency directly, and `clockname` is ignored.

■ To use multi-carrier HB Envelope, `funds` and `maxharms` must be provided, just like QPSS. Multi-carrier HB Envelope uses the QPSS HB engine and adopts the same parameters `funds` and `maxharms`.

■ If `harmonicbalance` is `yes`, HB Envelope is used. If it is `no`, TD Envelope is used. The default is no. For multi-carrier HB Envelope, this parameter is regarded as yes automatically.

■ Specify which harmonics of the clock frequency are of interest with the `harms`/`harmsvec` parameter. For TD Envelope, the default for `harms` is 1, which is appropriate for most applications. The number of harmonics you specify affects both output and computation time, but it does not affect accuracy. Hence, you should avoid specifying unnecessary harmonics. For example, in a power amplifier, only the first harmonic is needed because the signal of interest is near the fundamental frequency. In a mixer, both the zeroth and the first harmonics are needed because signals at both the baseband and near the fundamental frequency are of interest. For single carrier HB Envelope, the default for `harms` is 3. The number of harmonics does affect accuracy because HB Envelope calculates in frequency domain. For multi carrier HB Envelope, `harms` is replaced by `maxharms`. For multi-carrier HB Envelope, `harmsvec` also specifies the harmonics which need output, each group of elements with the size equal to that of `funds` is a selection of specific harmonic combinations of fundamental frequencies.

■ Specify a `stop` time long enough to detect slow signals. For example, complete at least a few clock cycles.

■ The maximum envelope step size is affected by many parameters. It can be directly limited by `envmaxstep`. It is helpful to specify a `modulationbw` frequency that reflects how the envelope is varying. This parameter provides an estimate of the modulation bandwidth and the simulator puts at least eight points within the modulation period. An approximate value is sufficient. To improve the noise floor in power spectrum density computation, use `strobeperiod` to get equally spaced envelope points.

■ If `fixstepsize` is `yes`, ENVLP analysis skips cycles indicated by `stepsize`. The efficiency of Envelope analysis depends on how many cycles are skipped. The strategy

is designed for general situations. For some cases, Envelope is faster with an appropriate fixed `stepsize`.

■ Specify a `swapfile` for simulation of large circuits. The same requirement holds for PSS and QPSS analyses. The `swapfile` parameter is supported for only the shooting engine.

■ Specify `fmspeedup=yes` for circuits with frequency modulated input.

Most of the remaining parameter requirements are the same as for the Spectre RF transient (`tran`) analysis and their meanings are consistent.

The errpreset Parameter in ENVLP Analysis

The effect of `errpreset` on some particular ENVLP analysis parameters is shown in Table 2-6.

Table 2-6 Parameter Defaults as a Function of errpreset

errpreset	maxstep	envmaxstep	reltol	relref	steady ratio	envIteratio
liberal	T/20	Interval/10	0.01	sigglobal	0.1	0.35
moderate	T/20	Interval/25	0.001	sigglobal	0.1	3.5
conservative	T/50	Interval/50	0.0001	alllocal	1.0	35.0

The effect of `errpreset` on parameters such as `reltol`, `relref`, `method`, `maxstep`, and `lteratio` is the same as defined for transient analysis with one exception. The transient simulation interval in ENVLP analysis is always a clock period.

The default value for `compression=no`.

■ For `compression=no`, the output file stores data for every signal at every timepoint for which Spectre RF calculates a solution. Spectre RF saves the x axis data only once, because every signal has the same x value.

■ For `compression=yes`, Spectre RF writes data to the output file only when the signal value changes by at least twice the convergence criteria. To save data for each signal independently, x axis information corresponding to each signal must be saved.

❑ If the signals in your circuit stay at constant values for large periods of the simulation time, setting `compression=yes` results in a smaller output data file.

❑ If the signals in your circuit move around a lot, setting `compression=yes` results in a larger output data file.

ENVLP analysis generates two types of output files for each specified harmonic of the clock fundamental

■ A voltage versus time (`td`) file

■ An amplitude/phase versus time (`fd`) file

The `td` file contains real time-domain waveforms. This file is generated when `outputtype={`*envelope*, *both*`}`. The waveforms are similar to waveforms generated by Transient analysis. The difference between the two is that for ENVLP analysis the integration of a clock cycle is done occasionally. Consequently, you normally see gaps between integrated clock cycles. Note that multi-carrier HB Envelope analysis does not generate a `td` file.

The `fd` file contains time varying Fourier coefficients (complex) of the circuit response at clock harmonics. The `fd` file is generated when `outputtype={`*spectrum*, *both*`}`. Time varying Fourier coefficients are discussed in "ACPR Calculation" on page 300.

The actual spectrum of each harmonic response is calculated from the Analog Design Environment (ADE) GUI. This is useful for applications such as ACPR calculation.

Plotting the Results of ENVLP Analysis

The Envelope Following Direct Plot form plots

■ Three different waveform (or *Function*) types

    ❑ *Voltage*

    ❑ *Current*

    ❑ *Power*

■ Three *sweep* types

    ❑ *Time*

    ❑ *Harmonic Time*

    ❑ *Spectrum*.

As shown in Figure 2-18, waveforms are called *Functions* on the ENVLP analysis Direct Plot form.

Figure 2-18 A Typical ENVLP Direct Plot Form

Plotting Voltage vs Time Waveforms

When you select both a waveform (*Signal*) and the *sweep* type *Time*, you plot a cycle-sampled version of the waveform.

When you use an Amplitude Modulated (AM) example with a 1 GHz carrier and a 1 MHz baseband tone, the resulting plot is a sampled version of the AM waveform. Instead of taking a continuous set of points as a transient analysis would, the ENVLP analysis saves simulation time by taking intelligently spaced samples of the waveform. When you visually connect the tops of the waveform samples, you can see the envelope that results from an equivalent transient analysis.

Depending on the phase of the clock at which the sampling occurs, different time domain envelopes result. Figure 2-19 shows two envelopes that might result from sampling at different clock phases. Either envelope might be traced out by an ENVLP analysis.

Figure 2-19 Sample Envelopes

Sample Envelopes
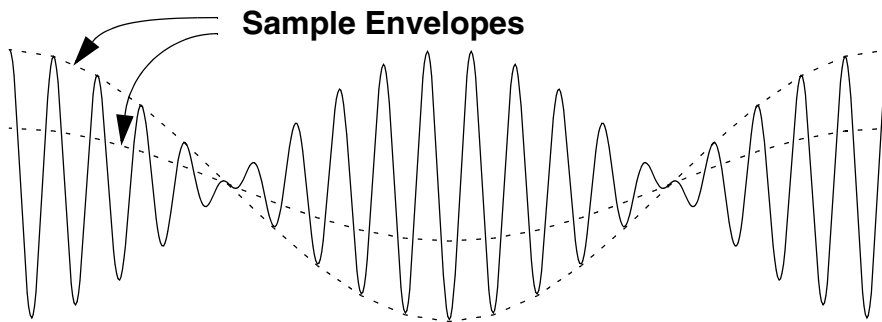
The other two *sweep* types (*harmonic time* and *spectrum*) plot *voltage* as a complex waveform. For complex waveforms, you can plot either the magnitude/phase or the real/ imaginary parts of the waveform. Note that if you change the clock phase at which sampling occurs, these complex waveforms are not affected.

When you select both *signal* and *harmonic time* as the *sweep* type, you plot the complex, time-varying Fourier coefficient for each harmonic. Using the AM example, when you select the 1st harmonic you see the 1 MHz baseband signal. In other words, the ENVLP analysis essentially strips off the 1 GHz carrier.

Plotting Power Spectral Density

When you select both *signal* and *spectrum*, the *Power Spectral Density Parameter* fields appear at the bottom of the plot form. After you provide values for the fields, the Direct Plot form calculates the *total number of samples*, *window size*, and *number of bins,* and then calls the *psdbb* calculator function.

Table 2-7 describes the *Power Spectral Density Parameter* fields.

Table 2-7 Power Spectral Density Parameters

From time and *to* time	Starting and ending times, respectively, for the time interval during which the spectral analysis is to be performed. Normally, these parameters are set to the simulator start and stop times, respectively. However, you might want to make the *From* time different than zero to exclude the start-up transient from the analysis. The Time Interval should be long enough to support the frequency resolution you want.

Table 2-7 Power Spectral Density Parameters

Nyquist half-bandwidth	Normally this parameter is set to 3 to 5 times the maximum modulation bandwidth frequency. The *Nyquist half-bandwidth* indirectly determines the spacing of the time points used in the FFT calculations. When the original time domain data points are too far apart to support the Nyquist bandwidth, you might see strange results such as unexpected spikes in the spectrum. The Nyquist half-bandwidth should be less than half the inverse of the smallest time step. To avoid aliasing, all signals in the system must have negligible power at the Nyquist half-bandwidth and beyond.
bin-width	The *Frequency bin width* determines the frequency resolution. A smaller frequency resolution usually produces a noisier spectrum. The power spectral density is a frequency-by-frequency average of the FFT taken over several time windows within the main time interval. A smaller *Frequency bin width* produces fewer samples to average together at each frequency. If you chose a small bin width, the resulting PSD looks "noisy" and has a jagged appearance. If you make the bin width too large you might soften what should be sharp edges in the spectrum.
Max plotting frequency and *Min plotting frequency*	The *Min* and *Max plotting frequencies* are the minimum and maximum frequency you wish to display. The *Min plotting frequency* is negative because the resulting plot is the power spectral density of a baseband signal. The power spectral density of a baseband signal does not necessarily have complex conjugate symmetry because the signal is generally complex.
windowing name	A preset list of available windowing functions used during the spectrum calculation. The *Window* selection determines how data at the edges of a time window is attenuated to control spectral leakage. Spectral leakage occurs because an FFT is always taken over a finite time interval. That is like multiplying the original waveform by a pulse of amplitude one and duration equal to the duration of the data. Multiplication in the time domain corresponds to convolution in the frequency domain. The transform of the long pulse is a high and narrow sinc function. As the pulse length goes to infinity, the main lobe of the sinc function approaches a Dirac delta function. When estimating the spectrum at a given frequency, the sinc function's side lobes cause spectral components of the untruncated signal to leak into the estimation of the spectrum at the main lobe.

Table 2-7 Power Spectral Density Parameters

detrending	Allows you to remove certain trends from the data before the spectral analysis. The options are *None, Mean* and *Linear*.

- *None* means do not detrend

- *Mean* removes the mean

- *Linear* removes the linearly growing component

The following typical parameter settings are used for the AM example just mentioned. These numbers assume that the *Stop time* is set to 60 us:

Start time	0
Stop time	60us
Nyquist half-bandwidth	5e6
Bin-width	20e3
Max plotting freq	5e6
Min plotting freq	-5e6
Windowing name	hamming
Detrending	none

For the AM example, you see a waveform displayed as $V^2/(Hz)$ versus frequency. You can think of this as

```
(rms passband volts)*(rms passband volts)/Hz
```

The spectrum plot is the estimated power spectral density of the complex envelope divided by two. The division by two occurs because the envelope is 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 it can then be directly compared against an input baseband signal.

The calculated *total number of samples*, *window size*, and *bin-width* are printed to the CIW (Command Interpreter Window) so you have the option of using the *psdbb* calculator function. As shown later, direct use of the *psdbb* function together with strobing can dramatically drop the noise floor. The following list displays the calculations that generate the *psdbb* parameters:

- $L = To - From$ (the *To* and *From* times are user inputs)

- *fmax* (**Nyquist half-bandwidth**) and **bin-width** are user inputs

- *#bins = floor(L*binwidth)*, compute with *#bins >= 1*. Here, *floor* means *take the integer part of...,* or truncate to the nearest integer.

- Compute the smallest m such that

  $2^m * (\#bins) > 2*L*fmax$

- *windowsize = 2^m*

- *number of samples = #bins * windowsize*

AGC Example

One application of ENVLP analysis is to assess the Automatic Gain Control (AGC) loop dynamics. AGC loops are important in communication systems where wide amplitude variations in the output signal leads to a loss of information. These signals need a good control to maintain a constant signal level at the output.

The PSS and QPSS analyses cannot compute the AGC response because the response is not periodic. The loop might have a periodic response if it were unstable but PSS and QPSS would be difficult to apply because the period of the instability would not be known before running the simulation. Transient analysis is not usually a good choice because the frequencies of the amplitude oscillation and carrier are too far apart. ENVLP analysis is an efficient and accurate alternative for assessing the stability of AGC loops as this example shows.

Consider the AGC loop shown in Figure 2-20. The behavioral blocks are simple multipliers from the *ahdl* library. The upper left multiplier is the variable gain amplifier (VGA). After filtering, the amplitude detector produces the square of the amplitude of the VGA output. The AGC loop input is a fixed-amplitude 100 Mhz carrier. A stable version of this circuit would generate an output signal with the same phase as the input signal but with the amplitude determined by the feedback loop. This design has an extra integration in the feedback loop that makes the loop unstable. The simulator's challenge here is to detect the design error by quickly and accurately simulating the unstable behavior.

Figure 2-20 AGC Loop
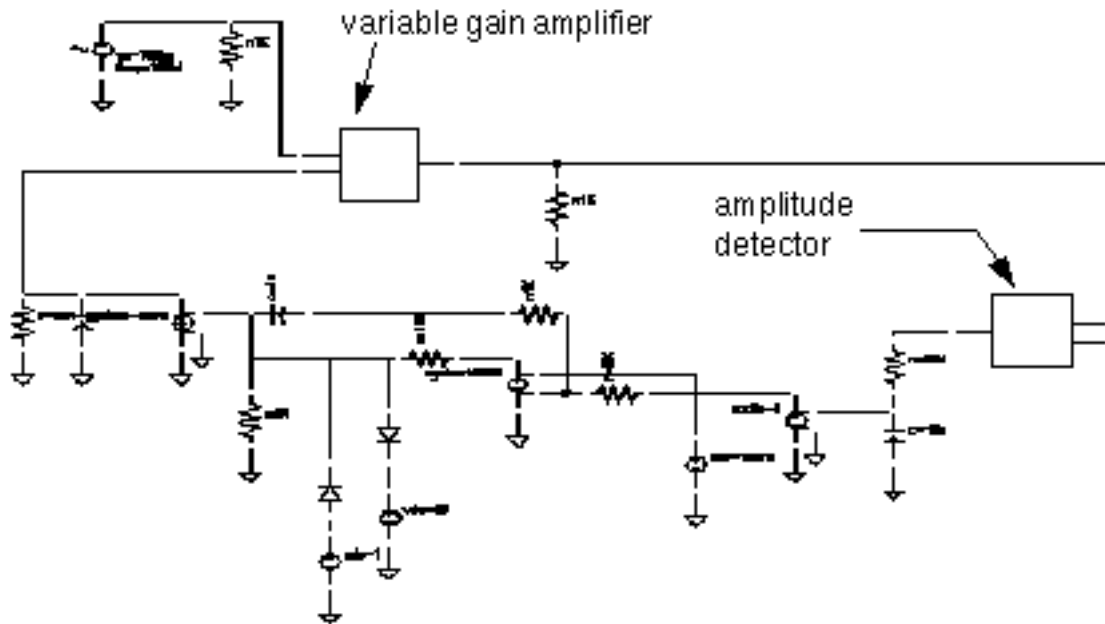
Figure 2-20 compares the results of transient analysis with the TD ENVLP and HB ENVLP analyses.

The common parameters for all three analyses are

Stop time=2m

Accuracy Defaults(errpreset)=moderate

For both ENVLP analyses

Number of harmonics=1

In Figure 2-21, waveforms for all three analyses show the instability in the output amplitude. Both ENVLP analyses (top) avoid a number of redundant carrier cycles and run much faster than the transient analysis (bottom). The TD ENVLP analysis is about 47 times faster than the transient analysis. The HB ENVLP analysis is 293 times faster than the transient analysis because this case is quite linear. The time savings is even greater with a 1 GHz carrier.

Figure 2-22 compares the ENVLP analyses with the transient analysis (bottom) for a single cycle. The ENVLP simulation is not only fast, it is accurate.

The Spectre RF workshop includes this example. It shows how to set up an envelope analysis for this case and how to see further results. Refer to the workshop for more information.

Figure 2-21 Comparison Between Transient and ENVLP Analyses Results
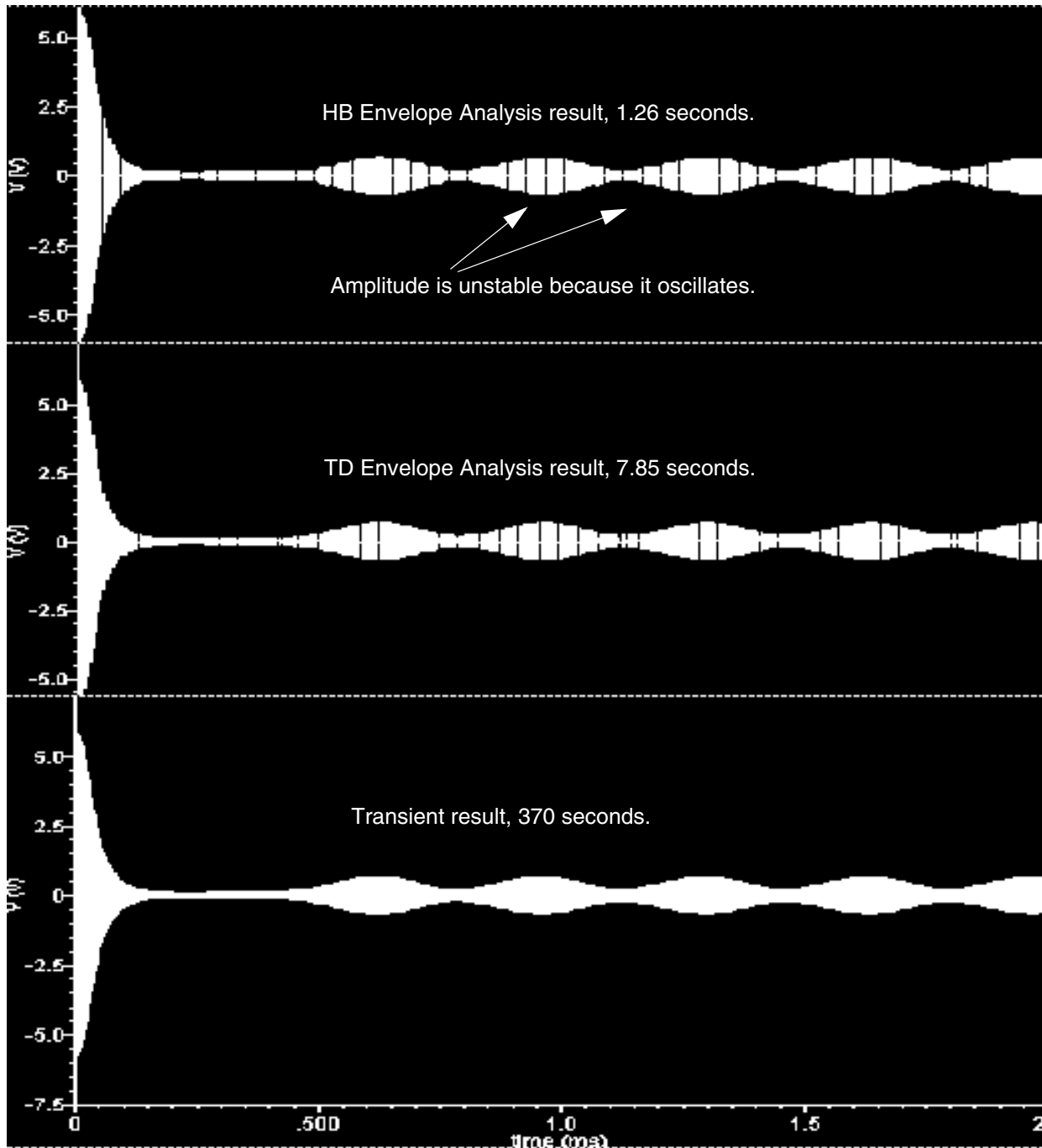
Figure 2-22 Blow-up of One ENVLP Cycle with the Transient Waveform

Cycle simulated by Envelope Analysis, others skipped
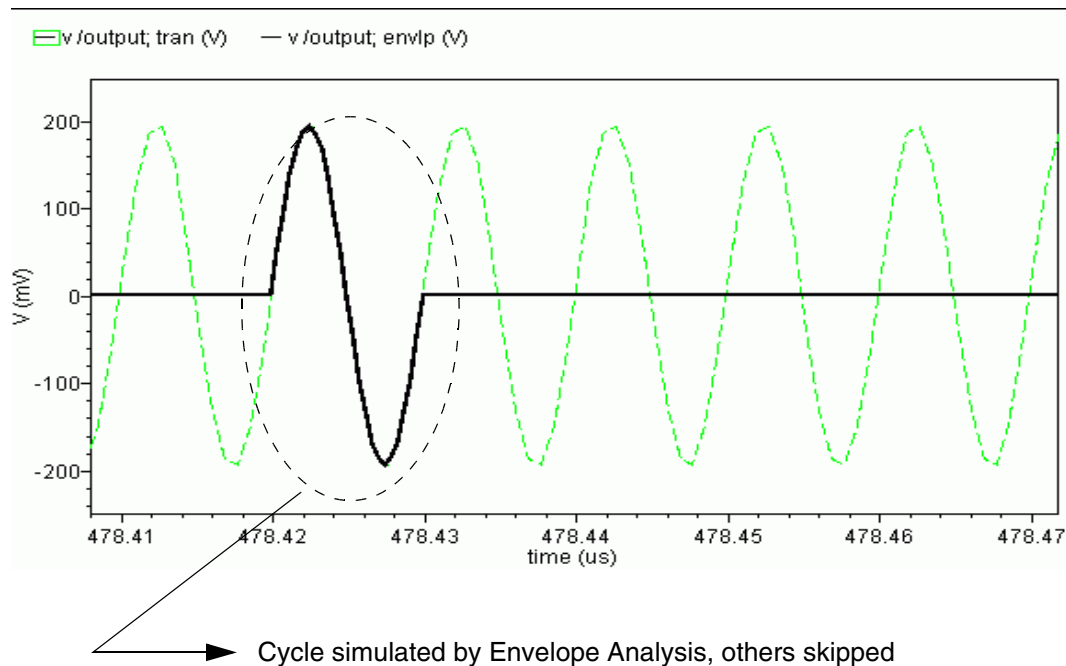
ACPR Calculation

ACPR (Adjacent Channel Power Ratio) is a common measure of how much power a transmitter emits outside its allotted frequency band. It is the ratio of the power in an adjacent band divided by the power in the allotted band. Regardless of exactly how you chose the frequencies and bands for the ACPR measurement, it is always extracted from the power spectral density of the transmitted signal. This section describes how the Analog Design Environment (ADE) estimates power spectral density (PSD).

Note: You cannot compute the PSD directly from PSS and QPSS analyses because the input baseband signals carry information and are therefore not periodic. The J-model (see [5] for details) is a very fast indirect method for computing ACPR, but like all behavioral models, has limitations. Using an ENVLP analysis, you can check J-model ACPR calculations much faster than you can with transient analysis.

PSDs are 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 must involve an averaging process. Any empirically derived average is an estimate because one can never take an infinite number of samples.

The PSD is a frequency-by-frequency average of a set of DFTs (discrete Fourier transforms) of the baseband signal. Here, the baseband signal is the harmonic-time result of an ENVLP analysis.

The waveform is first interpolated, if necessary, to generate evenly spaced data points in time. The spacing of the data points is the inverse of the DFT sampling frequency. The PSD is computed by first breaking the time Interval up into overlapping segments. Each segment is multiplied, 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, the frequency of the main lobe. Ideally, the sinc function would be a Dirac delta function but that requires an infinite time record. Good windowing functions have smaller side lobes than a 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. This is why a small bin width produces a jagged PSD. A smaller 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 just the square of the magnitude of the DFT of a stochastic process. At the other extreme, large 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.

A PSD Example

This section describes the various ENVLP outputs and how to use them to generate a PSD. Figure 2-23 shows the ACPR test circuit.

■   The behavioral blocks are two multipliers and one adder that model an ideal I/Q modulator

■   The carrier frequency is 1 GHz

■   The amplifiers are transistor-level models

The baseband input signals are read in with piecewise linear sources. The piecewise linear sources read data files. In this example, the data files contain CDMA baseband signals and

are stored in the *rfLib*. The files are listed as `cdma_2ms_idata` and `cdma_2ms_qdata`. The stored signals were created with the CDMA signal generator, which is also in the *rfLib*. You cannot use the CDMA signal generator directly because its internal DSP filters have a hidden state. Spectre RF does not work with any AHDL model that has a hidden state. You can also generate the baseband signals with SPW. The piecewise linear sources also read SPW's data format.
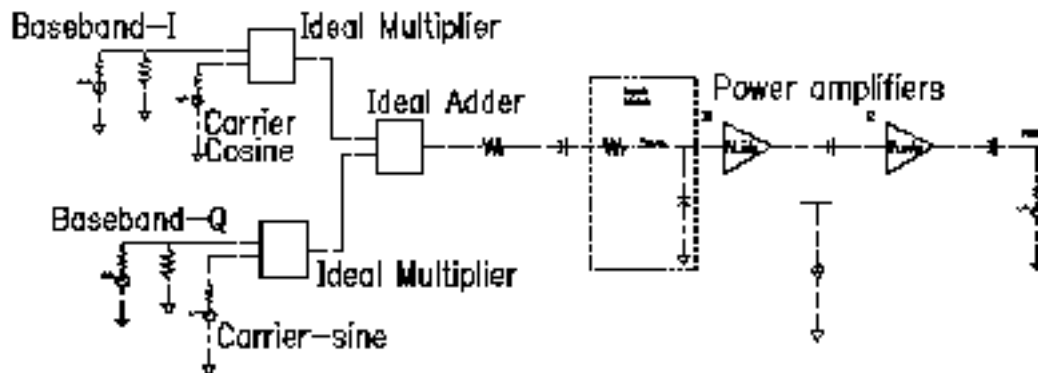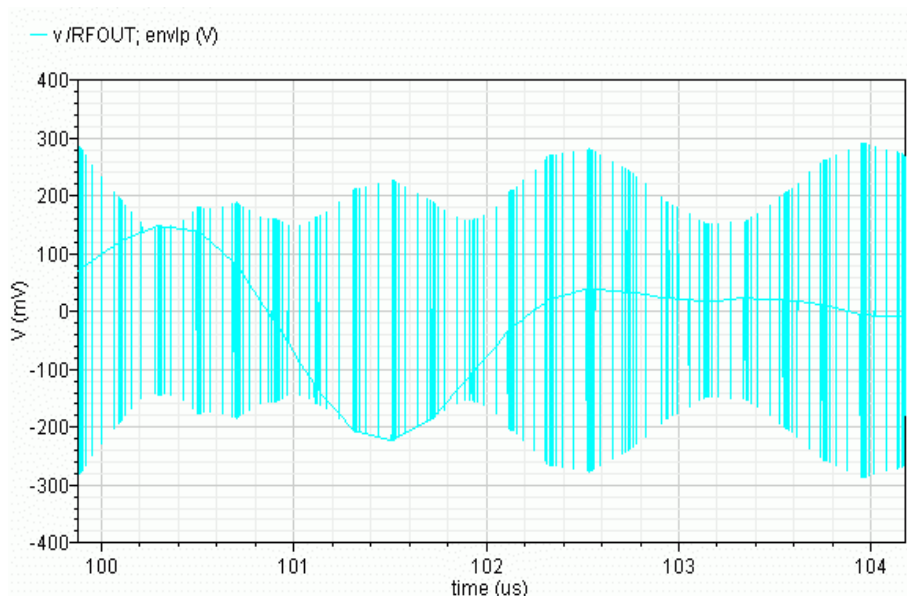
Figure 2-23 ACPR Test Circuit

Figure 2-24 displays the *time* response of the RF output. The top picture shows some of the cycles. A number of cycles are missing. This illustrates why ENVLP analysis is faster than transient analysis.

Figure 2-24 ENVLP Analysis Time Response

Although you can see something resembling an envelope in the *time* response, the detailed cycles can obscure it on larger time scales and it is not guaranteed to represent any particular phase of the envelope. The *harmonic time* response is designed to extract the baseband components or any other harmonic of interest. Most of the time the fundamental, or first harmonic, is the most interesting.

The *harmonic* analysis plots the real and imaginary parts of the specified Fourier component of each cycle as a function of its location in time. Figure 2-25 on page 303 and Figure 2-26 on page 304 show how the real and imaginary parts of the fundamental components of the RF input and output signals evolve with time. These signals are the baseband representations of the RF signals.

Figure 2-25 Input (top) and Output (bottom) Baseband Waveforms
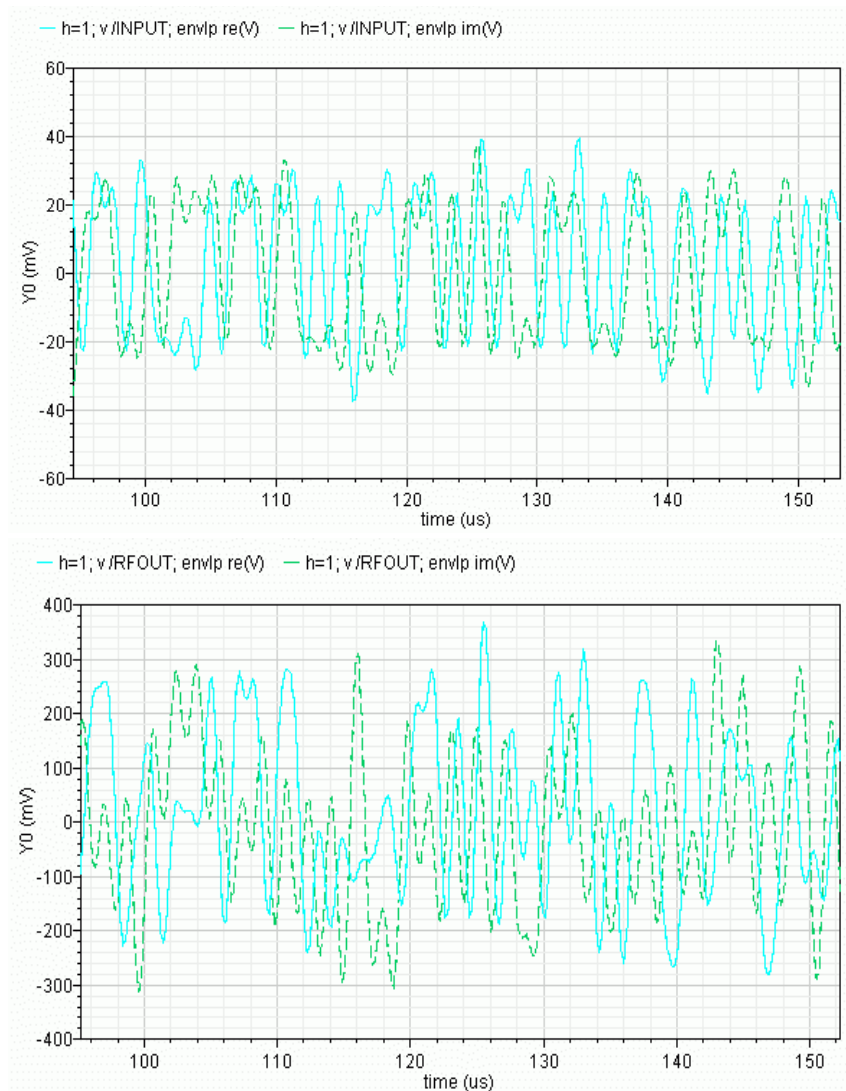
Not much can be ascertained directly from time-domain baseband waveforms. However, x-y plots of real and imaginary waveforms show the trajectory traced out by the baseband signal in the symbol constellation space. Figure 2-26 shows the input and output baseband trajectories. A trajectory is displayed by changing the x-axis to be the real waveform. The output trajectory is a scaled and rotated version of the input trajectory. The scaling factor is evident in the time domain waveforms but the trajectories also make the phase shift obvious. The transmitter introduces about 45 degrees of phase shift.

Figure 2-26 Baseband Input and Output Trajectories
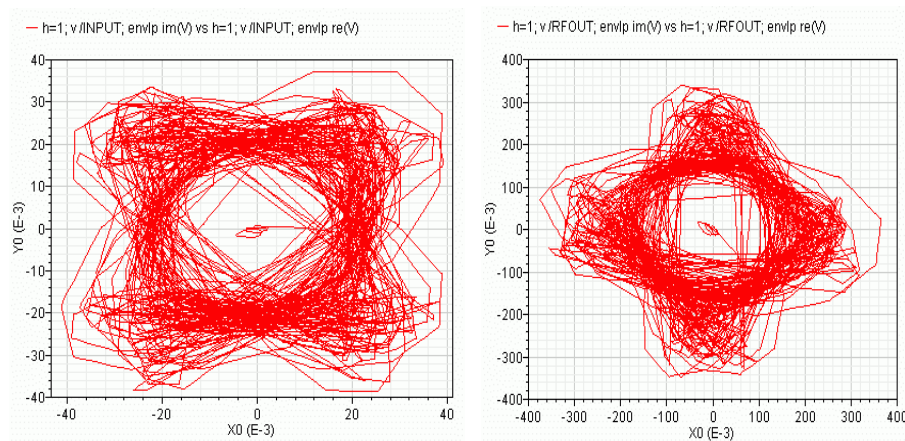
ACPR estimation requires power spectral densities. Use the *spectrum* button on the ENVLP Results form to estimate the power spectral density of a baseband signal. The baseband signal is the time-varying fundamental Fourier component described above. Figure 2-27 compares input and output power spectral densities. Because the input and output power spectral densities have the same shape, you can tell that this example shows very little spectral regrowth

Figure 2-27 Power Spectral Densities of the Input and Output RF Signals
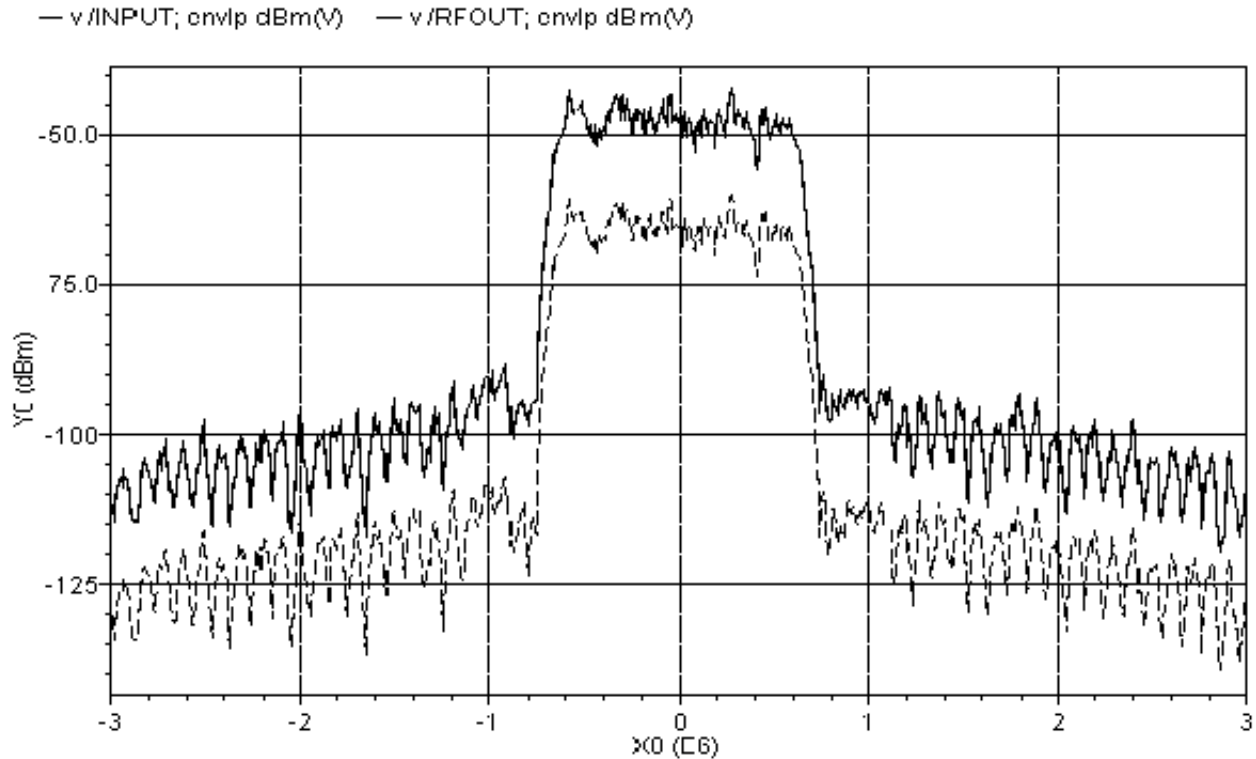
The power spectral densities shown in Figure 2-27 were created with the following option values.

From	0 s
To	300 us
Nyquist half-bandwidth	3 MHz
Frequency bin width	10 KHz
Max. plotting frequency	3 MHz
Min plotting frequency	-3 MhZ
Window	Hanning
Detrending	None

The baseband signal driving the transmitter dominates the transmitted PSD. In most cases, the baseband signals come from digital filters and 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. In measuring ACPR it is crucial to drive the transmitter with the proper baseband signals.

It is not practical to model digital filters in Spectre RF because Spectre RF cannot simulate state variables inside Verilog-A modules. Consequently, for now, you must pre-compute and store the baseband inputs and read them into the Spectre RF analysis through the `ppwlf` sources found in the *analogLib*, as shown in this example. 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 40dB below the peak of the output power spectral density, you need to create baseband drive signals with a noise floor at or below the required noise floor. If you use a DSP tool like SPW to create the signals, the filters in the baseband signal generator have to operate perhaps hundreds of times faster than those in the actual generator. Otherwise the signals do not have enough resolution. The noise floor depends heavily on interpolation error. You can strobe the harmonic time results to eliminate interpolation of the output but you can not eliminate interpolation of the baseband drive signals. The only way to 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. It is up to the user to generate ultra-high resolution drive signals.

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 it.

The Spectre RF workshop includes this example. To see how to setup envelope analysis for this case or to get further information, see the workshop.

Validation

A practical transmitter circuit [6] was simulated using the ENVLP analysis. The circuit has 185 MOSFETs, 79 nodes and 127 equations. It is driven by a 1.7475 GHz carrier and GSM baseband signals. The transient behavior was simulated for 2.5 ms, or roughly 4.5 million carrier cycles. It would take traditional transient analysis 154 days to simulate on an UltraSparc 10 with a 299 MHz CPU and generate 10 GB of data. In contrast, it took the ENVLP analysis 17.5 hours to finish in the same computing environment and it generated 1.2 MB of data. More importantly, the ACPR calculation from the ENVLP analysis results

matched very well with measured data as shown in Figure 2-28. The two results diverge past about 400 KHz but that is irrelevant to ACPR in this case because GSM channels are only 200 KHz wide.

Figure 2-28 ACPR Simulation Results
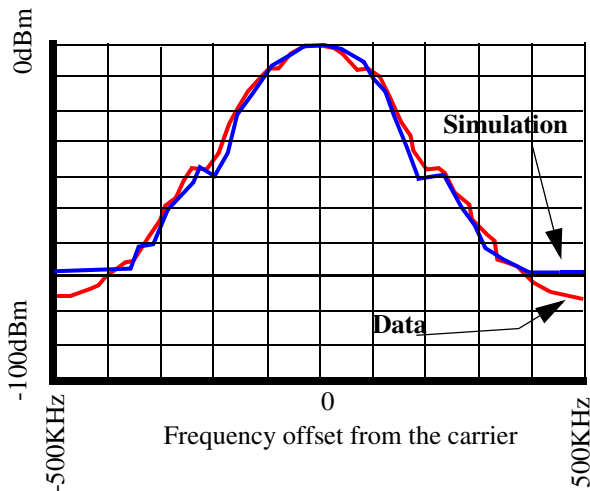
Autonomous ENVLP Analysis

RF circuits used in communication systems make great demands on the accuracy and efficiency of simulators. This is especially true when a strongly nonlinear circuit operates at high-frequency oscillating modes and contains slowly time-varying driving sources.

Circuits exhibiting these characteristics include

■   A PLL with time-varying reference

■   An RF mixer with a local oscillator

■   An AGC plus VCO

■   An oscillator followed by a power amplifier (PA)

Usually, the dynamics of these circuits involve multiple time scales and frequency-modulated (FM) signals. In these circuits, the fast time-scale oscillating frequency is modulated by slow time-scale signals.

Because transient analysis is very expensive for circuits with multi-rate dynamics, Cadence uses various envelope technologies to speed up the simulation. Traditional TD ENVLP analysis is efficient for non-autonomous circuits with AM, PM, FM or mixed modulated signals. However, in the FM case, if the modulation index is large, the signal spectrum

contains Bessel functions that are distributed in a wide frequency range. Standard ENVLP analysis does not handle this situation accurately and efficiently. Furthermore, ENVLP analysis requires a fixed frequency so it cannot handle the unknown frequencies of oscillator circuits. On the other hand, periodic steady-state analyses, such as HB PSS, can be used to simulate oscillator circuits but they do not allow any time-varying sources in the circuit.

The autonomous ENVLP analysis combines ENVLP analysis and autonomous steady-state PSS analysis. In this form of analysis, the oscillator problem is solved for each slow-varying time step. Then, the slow-varying characteristics of the circuit (including instantaneous oscillating frequencies and other envelope information) is obtained by integrating occasional cycles.

A Faster Steady State Oscillator Analysis

In the steady state (PSS) oscillator analysis, the initial values specified for oscillating frequency and other conditions are important for Newton convergence. To estimate the value of the stabilized frequency, a transient analysis can be performed before the steady state oscillator analysis. In some oscillator circuits, such as high Q circuits, it takes a long time for the oscillation to build up and stabilize. In this case, autonomous envelope can be used to speed up the analysis.

Figure 2-29 The rfOsc Oscillator Schematic
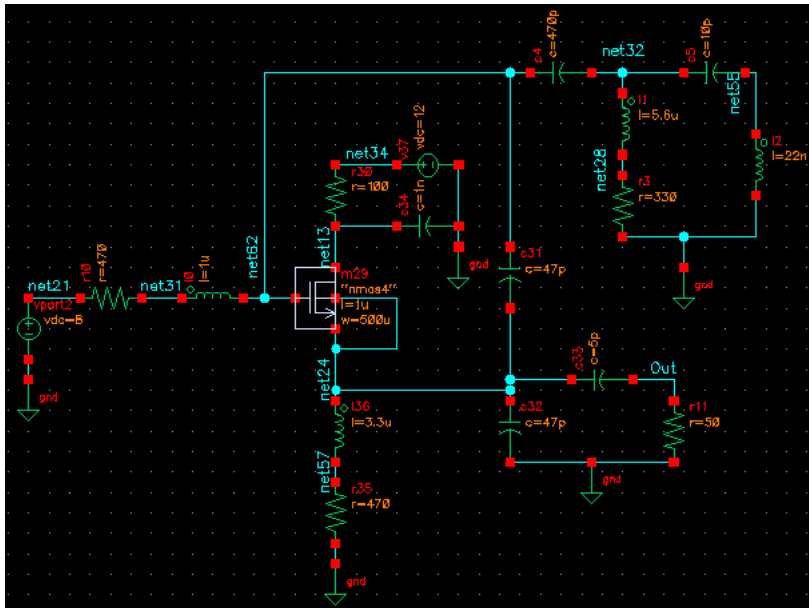
For the `rfOsc` example shown in Figure 2-29, the oscillator takes about 1.5 microseconds to start up and 3.5 microseconds to stabilize. The results are shown in Figure 2-30.

Use the following analysis commands to speed up the PSS analysis.

1. Do enough microseconds of autonomous ENVLP analysis to ensure the oscillation is stable and write the solution into the env.dat file.

2. Do an autonomous PSS analysis that skips the DC analysis and reads the initial condition from the env.dat file.

3. Do a Pnoise analysis and calculate the phase noise for the oscillator.

Example 2-1 illustrates the three analysis statements.

Example 2-1 Analysis Commands for a Faster PSS Analysis

```
envlp (Out 0) envlp harms=3 stop=20u maxstep=100p envmaxstep=100n flexbalance=1
+ tstab=2u fund=400M method=traponly envmethod=traponly errpreset=moderate
+ annotate=status writefinal="env.dat"
pss  ( Out 0 )  pss  fund=410M  harms=10 skipdc=yes readic="env.dat"
+ annotate=status  tstabenvlp=no
pnoise  ( Out )  pnoise  sweeptype=relative relharmnum=1 start=1K stop=300M
+ dec=5 maxsideband=10  annotate=status
```

To run autonomous ENVLP analysis efficiently

■ To use HB Envelope, flexbalance must be set to yes. Otherwise, TD Envelope is used. If flexbalance is yes, harms is the maxharms of the clock fundamental and the default value is 3. The harms is an output parameter for TD but it is an input parameter for HB. Consequently, harms has a significant impact on the performance.

■ A reference port, for example, (Out 0), must be specified to indicate autonomous. This is similar to autonomous PSS. The reference port is used for two purposes:

❑ Estimating the initial fundamental frequency and calculating instantaneous frequencies at each envelope time step.

❑ Checking the phase=0 condition for the port when doing shooting or HB Newton iterations.

The reference port for ENVLP and the reference port for PSS can be different. For a better estimate of the period, select one of the output ports of an oscillator whose output waveform is close to sinusoidal.

■ In a non-autonomous envelope analysis (but not in an autonomous analysis), the clockname must be specified. However, in an autonomous envelope analysis, the fund or period option must be specified as an initial guess of the fundamental frequency.

■ Usually, you must set a tstab value. The default value of tstab for an autonomous envelope analysis is 4 cycles of the fundamental, which might not be enough to ensure

convergence. During an HB Envelope analysis, especially for high-Q designs, enough `tstab` is needed to ensure that the oscillation starts.

■ Sometimes tightening the `reltol` parameters for the HB autonomous envelope analysis can make the result more accurate at the calculated cycle, which might improve the prediction and allow more cycles to be skipped. However, do not set `reltol` tighter than 1e-6 because that decreases the performance dramatically.

The oscillating frequency is 410 MHz, which corresponds to the period 2.44 nanoseconds. Figure 2-30 shows that there are about 0.1 microseconds between two occasional ENVLP analysis cycles. This implies that about 40 cycles were skipped for each ENVLP analysis step.

Figure 2-30 Autonomous ENVLP Analysis Results over 20 Microseconds
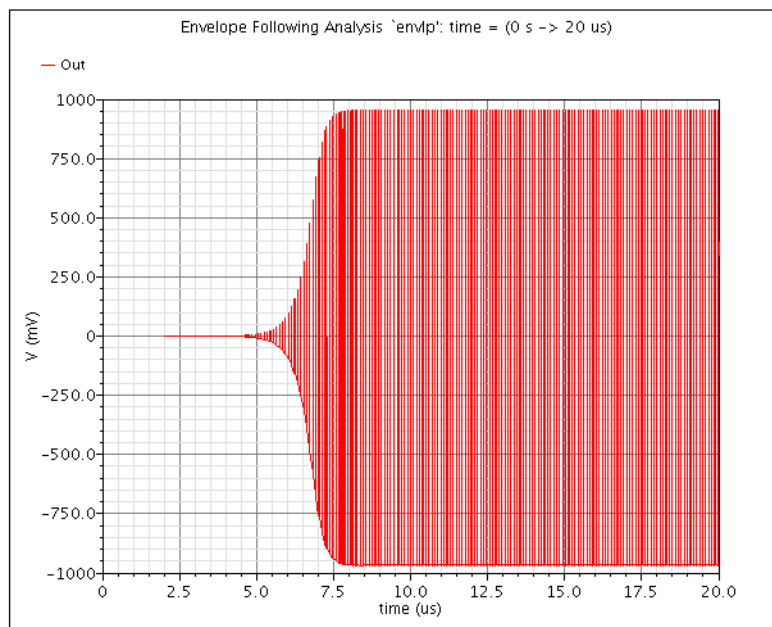
Figure 2-31 Transient Analysis Results over 20 Microseconds
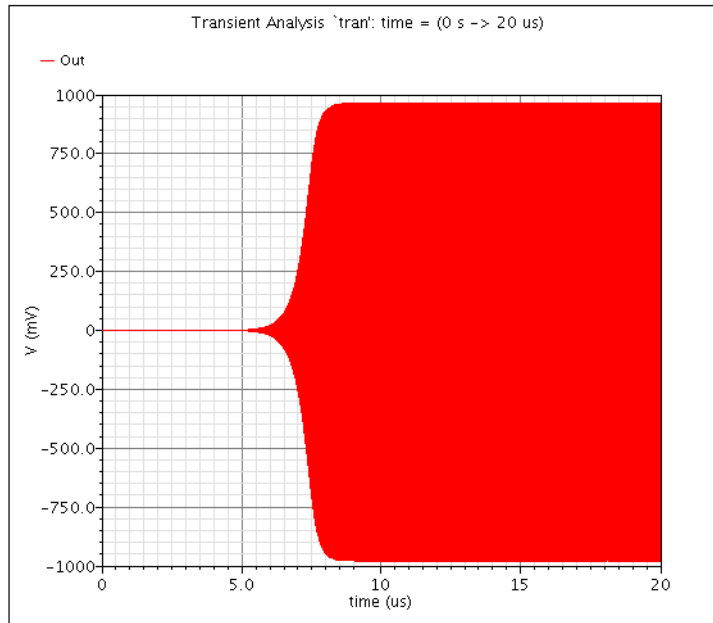
After the autonomous ENVLP analysis, you can plot the time-varying instantaneous frequency. For example, applying a 1 MHz sinusoidal source on the control voltage of a 3.46 GHz VCO circuit and then displaying the instantaneous frequency produces the plot shown in Figure 2-32. The ripple on the curve indicates that the VCO frequency follows the 1 MHz modulation signal. This circuit cannot be analyzed using either autonomous PSS or non-autonomous ENVLP analysis. The transient analysis can be used but it is very time consuming as discussed in the previous example.

Figure 2-32 Instantaneous Frequency of VCO with Sinusoidal Control Voltage
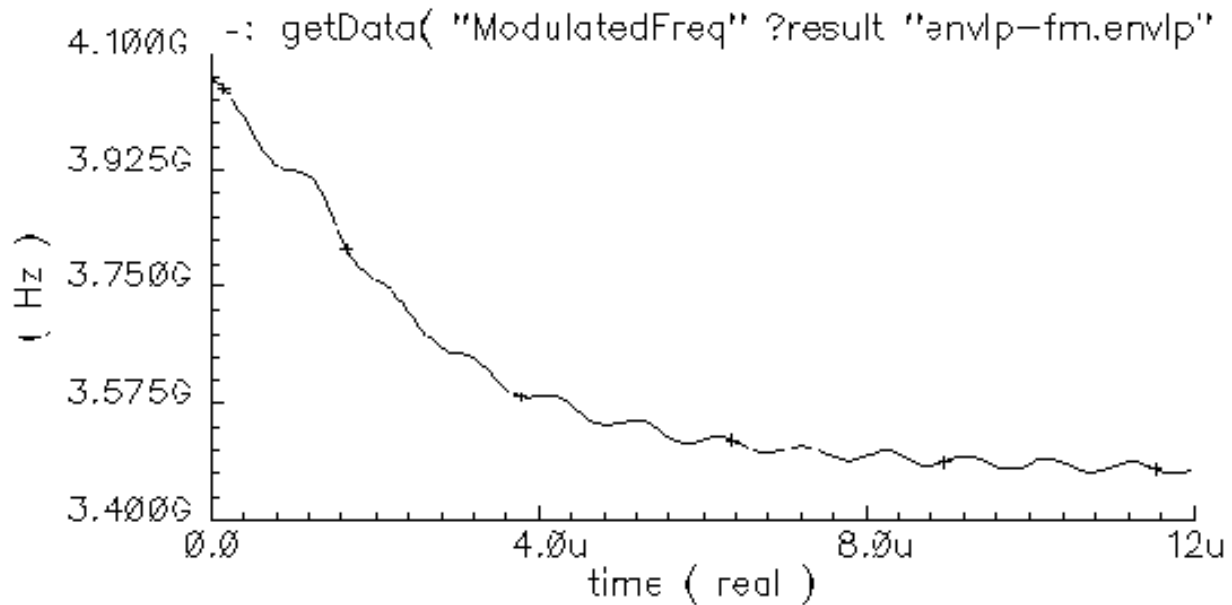
Simulating a Mixer with a VCO

This example simulates a mixer with two inputs

■   Baseband data

■   Output from the VCO circuit (LO)

The results show that the VCO frequency is affected by the baseband data signal.

Figure 2-33 is the circuit schematic for the example. The details of the VCO are not shown in the schematic. Figure 2-34 and Figure 2-35 show the modulation effects to oscillating frequency from a sinusoidal and a digital data input of port `Prf` respectively.

Figure 2-33 Schematic of Mixer with VCO
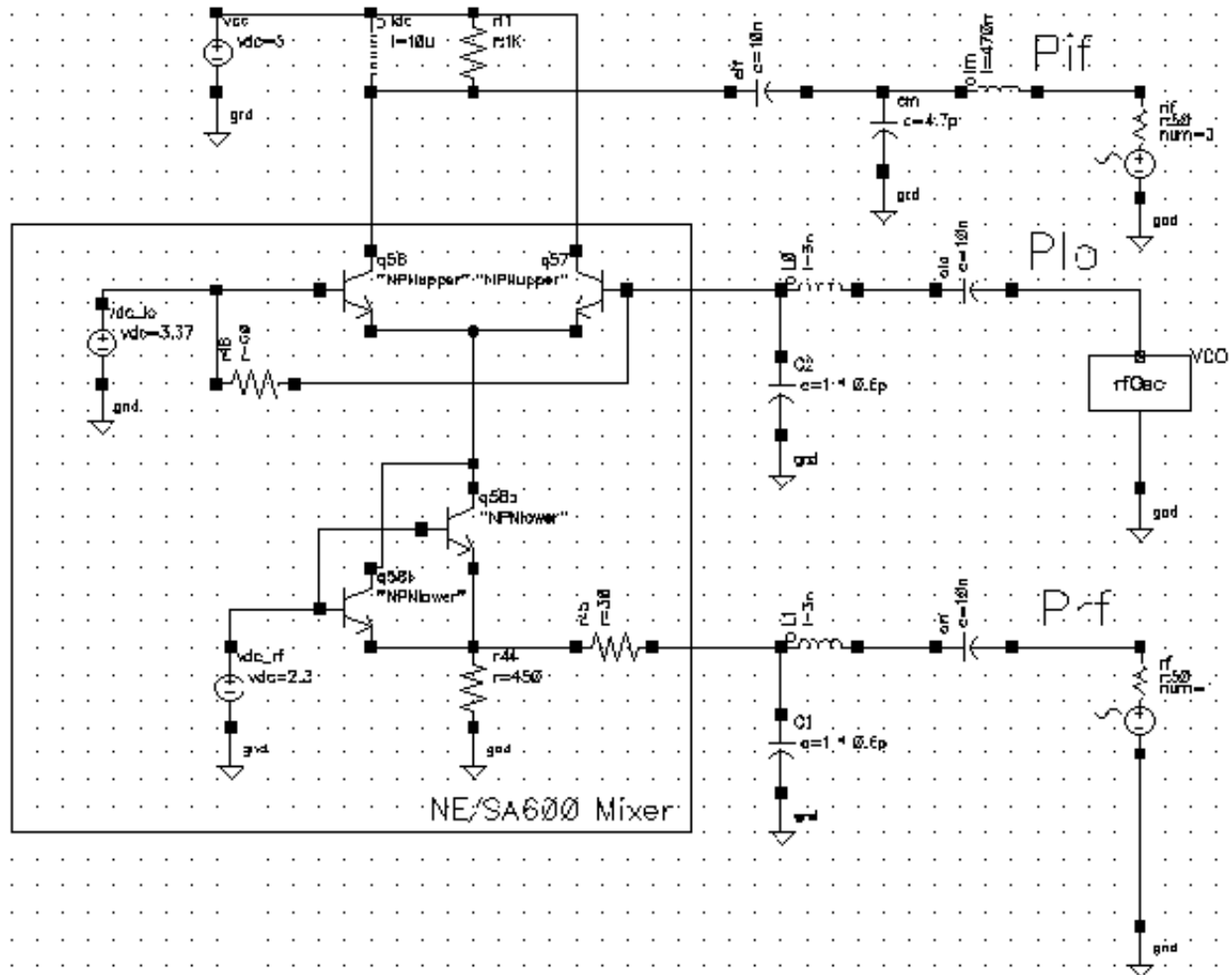
Figure 2-34 Results with Sinusoidal Data at Prf
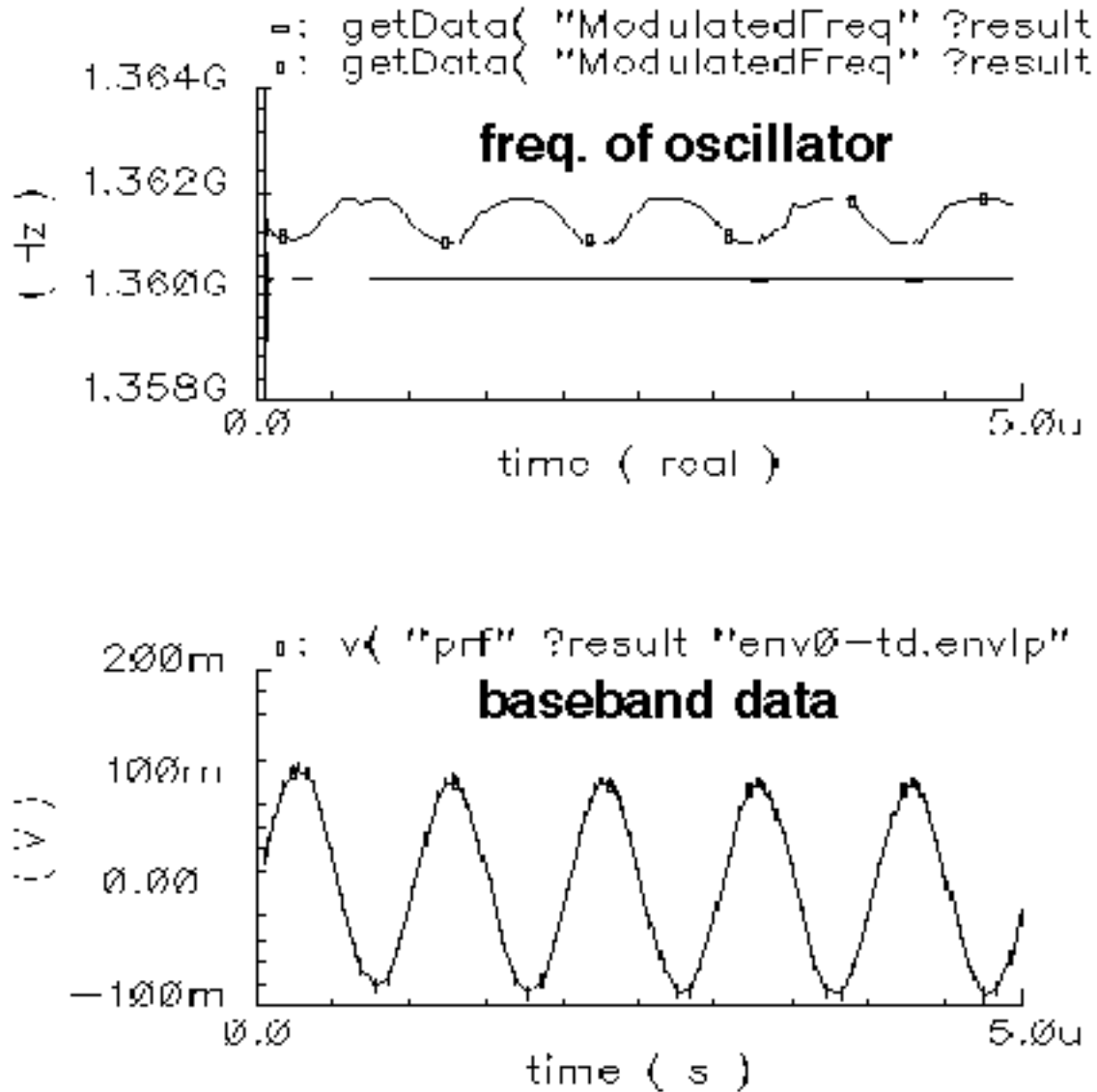
**Figure 2-35 Results with Digital Data at Prf**
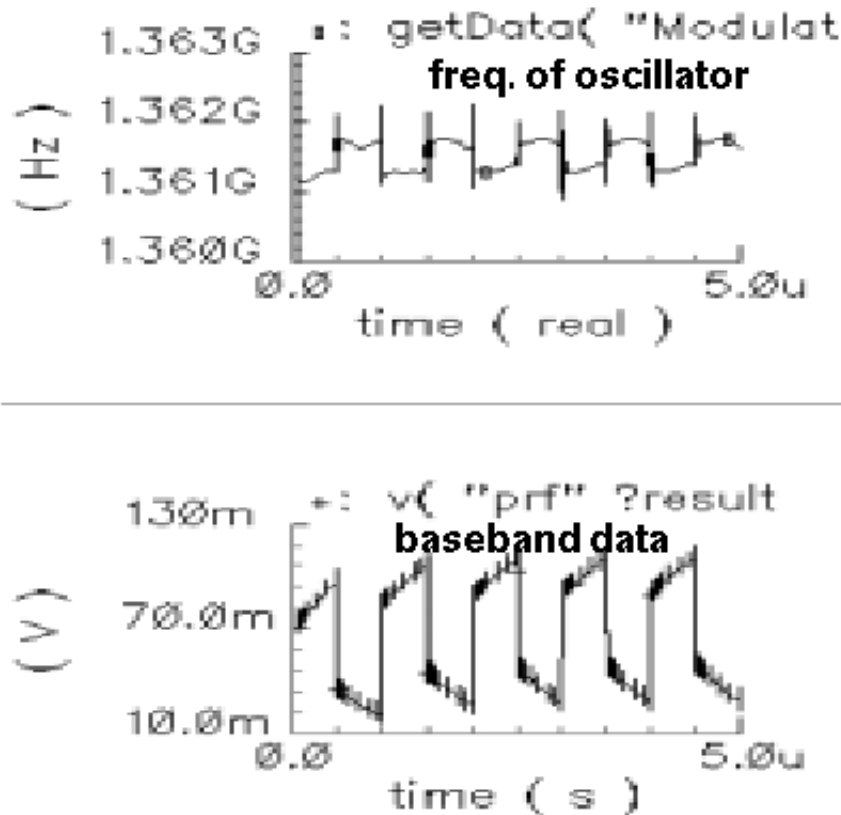
Simulating Oscillator-Pulling Effects for a Power Amplifier

In oscillator design, the oscillator-pulling effect is a common problem that can degrade performance. LO pulling is a phenomenon in which the LO frequency is influenced by a strong power amplifier signal when both the power amplifier output and the oscillator are at a high frequency.

This example simulates the system whose block diagram is shown in Figure 2-36. The system contains a local oscillator and a mixer followed by two power amplifier stages.

Figure 2-36 Circuit to Measure the Load Pulling Effect for an Oscillator
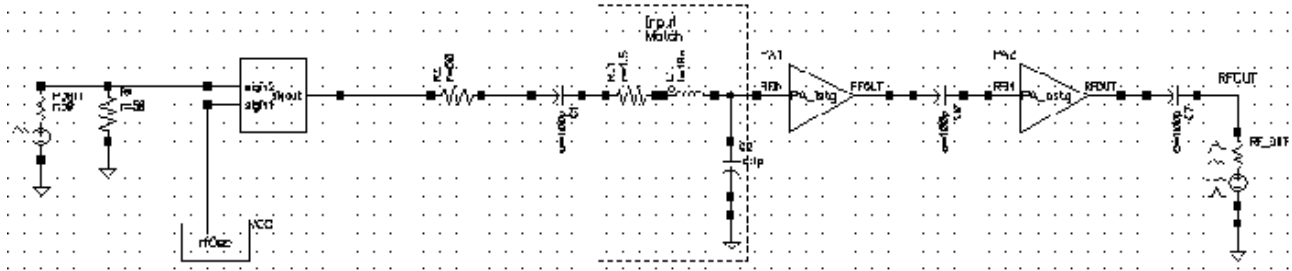
The simulation results are plotted in Figure 2-37. The upper plot is the instantaneous frequency from the VCO output. The lower plot is the baseband data signal. The figure shows that the oscillating frequency is affected by the power amplifier.

Figure 2-37 Simulation Results Showing Pulling Effect of the Oscillator
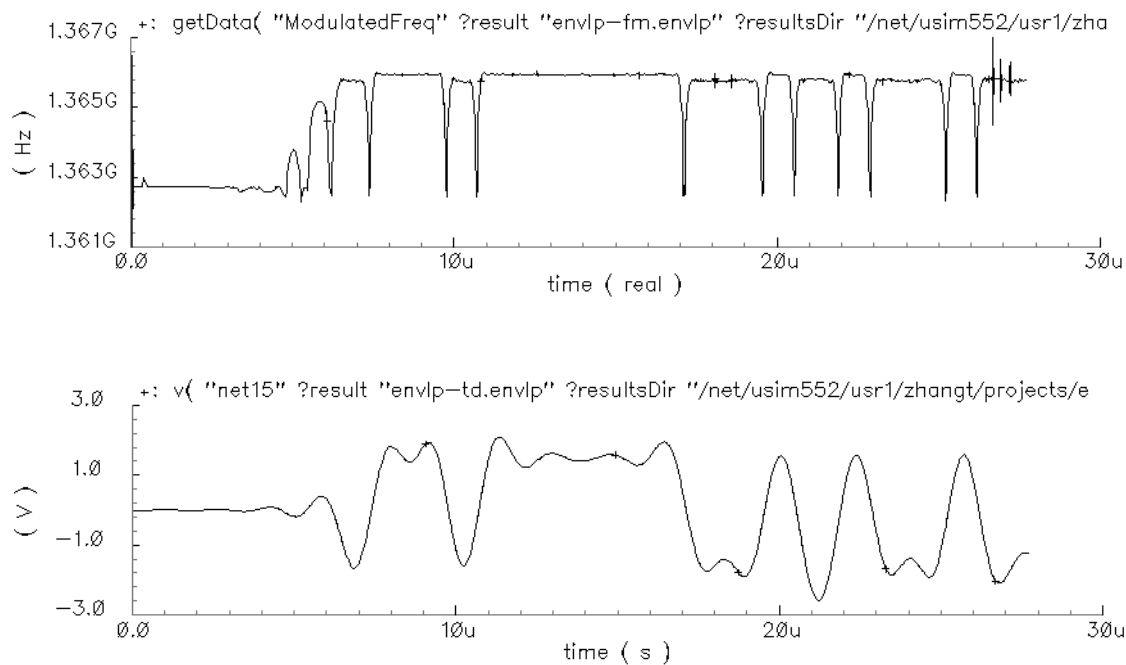
Simulating Circuits with Driven FM Sources

Another new feature of ENVLP analysis is faster simulation for circuits with driven FM sources.

Standard envelope analysis handles FM signals by sampling the signal at equal interval time points. The period is determined by the carrier frequency. For FM signals with large modulation indexes, standard envelope analysis is

■ Not efficient due to Newton iteration failures. The failures occurs because the polynomial condition is not satisfied. This can be seen in Figure 2-38 on page 317 and Figure 2-39 on page 317. Figure 2-38 indicates that standard envelope analysis is not working at all and the simulator is just doing a transient analysis.

■ Not accurate for computing the harmonic coefficients, which have been defined as the envelopes of the signal, because the real instantaneous frequency is different from the frequency used to do the Fourier transform. This can be seen by examining the output results of post-processing in Figure 2-40 on page 318 and Figure 2-41 on page 318, which display the derivative of phase for the first harmonic for the two methods respectively.

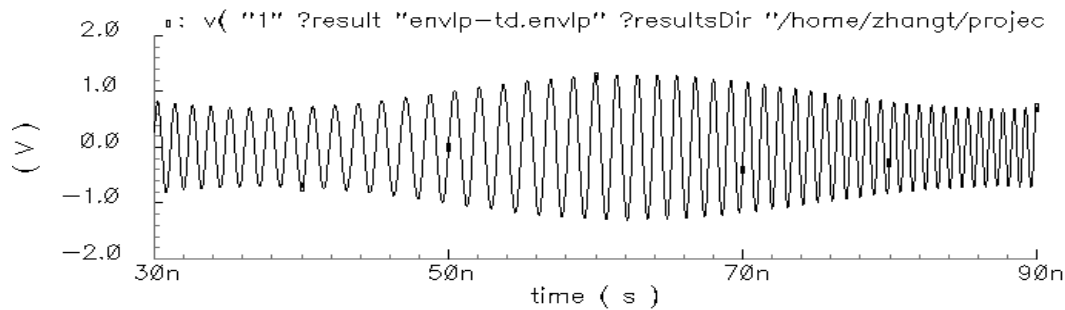**Figure 2-38 Standard ENVLP Analysis for an FM Signal**

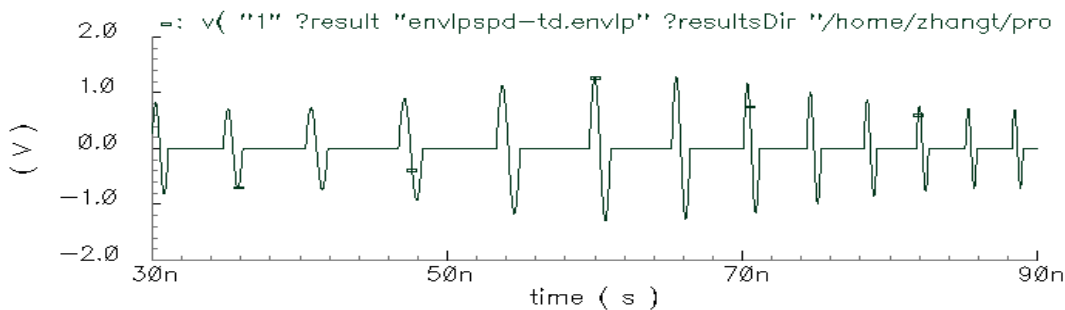**Figure 2-39 Faster ENVLP Analysis for an FM Signal**

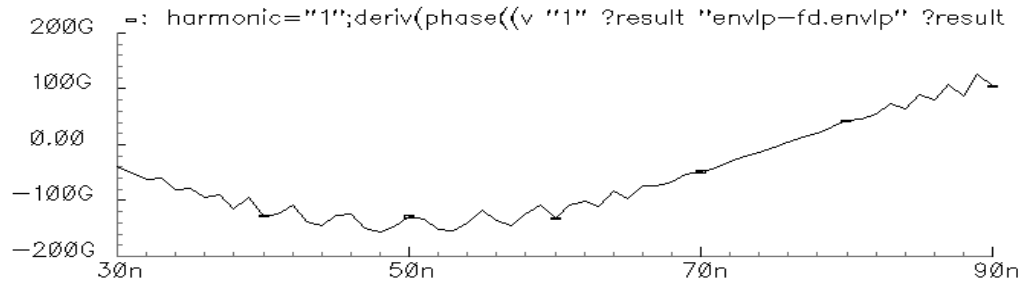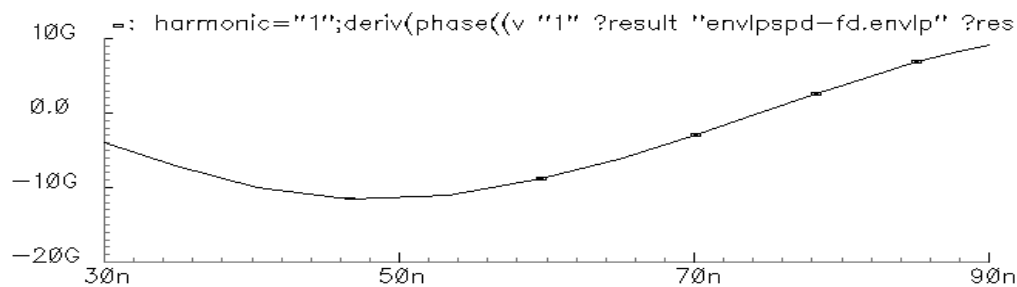**Figure 2-40 Frequency Profile Output From the Standard Method**

Figure 2-41 Frequency Profile Output From Fast ENVLP Analysis for FM Sources

To use the fast ENVLP analysis, you must represent the FM driven sources with Spectre internal source models such as the `port`, `vsource`, and `isource` components. Then, set the ENVLP analysis option `fmspeedup=yes`.

Arbitrary Frequency Modulated Sinusoidal Sources

FM sinusoidal sources can be specified in three different ways using `vsource`, `isource`, or `port`.

■   Specify modulated frequency (sinusoidal modulation)

```
V1 (1 2) vsource type=sine freq=ω₀ fmmodindex=k fmmodfreq=ω
```

For this case, write the FM signal from the source as

$$S(t) = A\sin(\omega_0 t + k\sin(\omega t))$$

■   Specify one file name, where the file contains the FM data

```
V1 (1 2) vsource type=sine freq=ω₀ fmmodindex=k fmmodfiles=["f_data"]
```

For this case, write the FM signal from the source as

$$S(t) = A\sin\left(\omega_0 \div k\int_0^t \delta\omega(\tau)d\tau\right)$$

where $\delta\omega(\tau)$ is the frequency modulation data from the `f_data` file.

■ Specify the names of two files that contain I and Q data for modulation. For example,

`V1 (1 2) vsource type=sine freq=`ω_0` fmmodindex=k fmmodfreq=["I" "Q"]`

For this case, write the FM signal from the source as

$$S(t) = A(t)\sin(\omega_0 + k\theta(t))$$

Here, *k* is the modulation index

$$A(t) = \sqrt{(I(t))^2 + (Q(t))^2}$$

$$\theta(t) = arc\tan\left(\frac{Q(t)}{I(t)}\right)$$

Where *I(t)* and *Q(t)* are the data from the `I` and `Q` files, respectively.

The file format for the `I` and `Q` files is the same as the PWL file format. The keywords related to PWL files (such as `pwldbm` and `pwlperiod`) also work for FM files in this case.

Running a Fast FM ENVLP Analysis

To speed up the FM ENVLP analysis, set the `fmspeedup=yes` option in the ENVLP analysis command. The default value of `fmspeedup=no`. Also set **clockname** to be the same as the **fundname** of the FM source.

Notice that the fast FM ENVLP analysis has the following limitations.

■ The algorithm that speeds up the ENVLP solver only works with FM sources

■ Frequency modulation can be applied only to sinusoidal sources (`type=sine`)

The following schematic, shown in Figure 2-42, is a circuit that includes an FM source followed by a PA, an LNA, and several other receiver components that perform demodulation.

### Figure 2-42 The Circuit with an FM Source
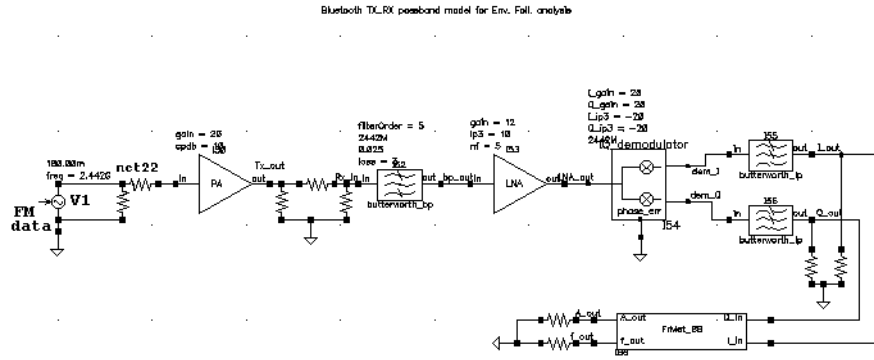
The FM source V_1 is defined as follows

```
V1 (net22 0) vsource type=sine ampl=180.00m freq=2.442G \
fmmodindex=24420000 fundname="f_carrier" fmmodfiles=[ "f_data" ]
```

With `fmspeedup=0`, ENVLP analysis runs for 140 s for this case. The result at `net22` is shown in Figure 2-43. With `fmspeedup=1`,ENVLP analysis runs for 27.8 s. The result at `net22` is shown in Figure 2-44.

Figure 2-43 Output at net22 From Standard ENVLP Analysis
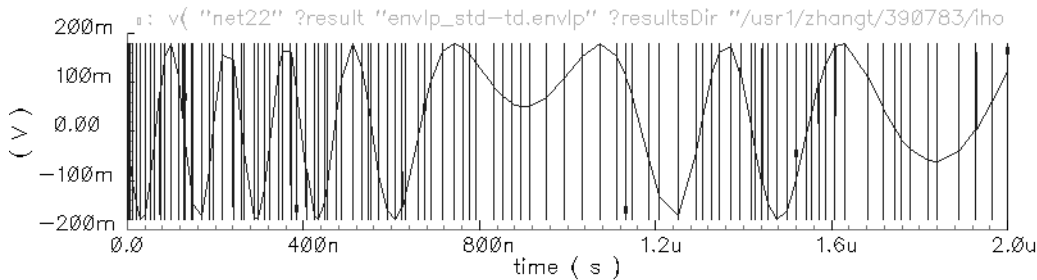
Because the clock frequency is adjusted while `fmspeedup=1`, the phase of each clock cycle is constant. Hence, a straight trace line is shown in Figure 2-44. However, with `fmspeedup=0`, the phase of each clock cycle is changing, which is shown in Figure 2-43.

Figure 2-44 Output at net22 From ENVLP Analysis with fmspeedup=1
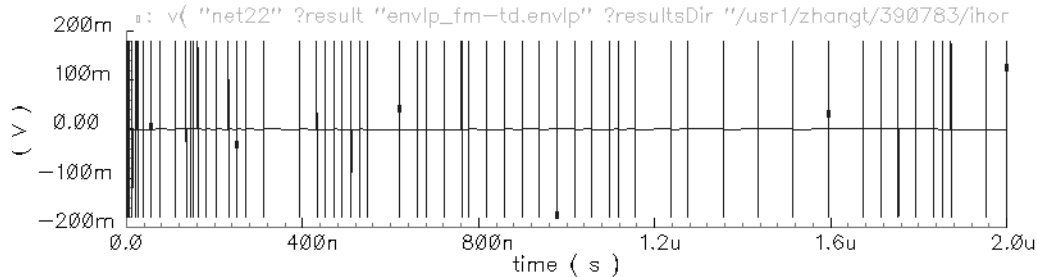
Frequently Asked Questions

Is ENVLP analysis as fast as transient analysis?

For circuits without hidden states, (which are usually introduced by Verilog-A modules), the answer is yes. Because a Newton method similar to that of PSS is used in Envelope analysis, circuits with hidden state cause difficulties, just as they do for PSS. However, most hidden states can be overcome [10].

How do I choose which ENVLP analysis to use?

The HB method is very efficient when simulating weakly nonlinear circuits because only a few harmonics are needed to represent the solution accurately. For highly nonlinear circuits with sharply raising or falling signals, time domain shooting is more suitable. However, when exploring design trade-offs using a few harmonics where accuracy is not of primary concerns, HB envelope might be the best choice.

How should I choose the number of harmonics for HB ENVLP analysis?

The default value is 3. The best value to use depends on the linearity in one cycle of the fast signal (LO or clock). For linear cases, a value of 1 is enough for accurate results. However, for nonlinear cases, such as a circuit with a square clock, even a value of 15 might not be large enough.

For what kind of circuits is multi-carrier HB Envelope suited?

As its name suggests, multi-carrier HB envelope analysis is designed for circuits with multi-carrier signals. For example, in a modulation system with a 1G LO and 10M IF where IF is a

modulated signal, both LO and IF can be looked on as carriers, so multi-carrier HB Envelope is useful.

Does multi-carrier HB Envelope support the function FM speed up?

No, it does not.

Does multi-carrier HB Envelope handle autonomous circuits?

No, it does not. Multi-carrier HB Envelope uses the multi-tone HB engine, which does not handle autonomous circuits.

References

[1]   R. Telichevesky, K. Kundert and J. White. "Efficient steady-state analysis based on matrix-free Krylov-subspace methods". *Proceedings of the 32nd Design Automation Conference*, June 1995.

[2]   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". *Proceedings of the 36th Design Automation Conference*, June 1999.

[3]   L. Petzold, "An efficient numerical method for highly oscillatory ordinary differential equations". *SIAM Journal of Numerical Analysis*, vol. 18, no. 3, pp 455-479, June 1981.

[4]   K. Kundert, J. White and A. Sangiovanni-Vincentelli. "An envelope-following method for the efficient transient simulation of switching power and filter circuits". *IEEE International Conference on Computer-Aided Design: Digest of Technical Papers*, November 1988.

[5]   J. Chen, D. Feng, J. Phillips, and K. Kundert, "Simulation and modeling of intermodulation distortion in communications circuits". *Proceedings of the Custom Integrated Circuit Conference*, San Diego, CA, May 1999.

[6]   K. C. Tsai and P. R. Gray, "A 1.9GHz 1W CMOS class E power amplifier for wireless communications", *IEEE Journal of Solid State Circuits*, pp. 962-970, Jul. 1999.

[7]   Behzad Razavi. *RF Microelectronics*, Prentice Hall, c1998. Series: Prentice-Hall Communications Engineering & Emerging Technologies Series.

[8]   E. Ngoya, J. Rousset, and D. Argollo. "Rigorous RF and microwave oscillator phase noise calculation by envelope transient technique". In *Microwave Symposium Digest*, 2000 IEEE MTT-S International, volume 1, pages 91-94 vol.1, 2000.

[9]   O. Narayan and Jaijeet Roychowdhury, *Analyzing Oscillators Using Multitime PDEs*.

[10]   Ken Kundert, *Hidden State in SpectreRF*, 2003.

3

Oscillators and Autonomous PSS Analysis

PSS analysis is capable of handling both autonomous (non-driven) and non-autonomous (driven) circuits. Autonomous circuits are time-invariant circuits that have time-varying responses. Thus, autonomous circuits generate non-constant waveforms even though they are not driven by a time-varying stimulus. Driven circuits require some time-varying stimulus to generate a time-varying response. The most common example of an autonomous circuit is an oscillator. Common driven circuits include amplifiers, filters, mixers, etc.

With driven circuits you specify the analysis period, which must be an integer multiple of the period of the drive signal or signals. Autonomous circuits have no drive signal and you do not know the actual period of oscillation precisely in advance. Instead, you specify an estimate of the oscillation period and PSS analysis computes the precise period along with the periodic solution waveforms.

When applied to autonomous circuits, PSS analysis requires you to specify a pair of nodes. 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, PSS assumes the circuit is autonomous; if not, the circuit is assumed to be driven. PSS analysis monitors the potential difference between these two nodes and uses it in the initial transient analysis portion of the autonomous PSS analysis; the signal is used to determine a better estimate of the period.

Phases of Autonomous PSS Analysis

A PSS analysis consists of two phases: an initial transient phase, which allows the circuit to be initialized, and the shooting phase, which is where the periodic steady-state solution is computed. The transient phase consists of three intervals. The first starts at t_{start}, which is normally 0, and continues through the onset of periodicity for the independent sources. The second is an optional stabilization interval, that you specify, whose length is t_{stab}. The final interval of the transient phase is peculiar to the autonomous PSS analysis. In this phase, the length of which is four times the specified estimate of the oscillation period, the PSS analysis monitors the waveforms in the circuit and develops a better estimate of the oscillation period.

During the first stage of the transient phase, PSS extracts the fundamental frequency from the set of nodes specified in the PSS statement. During the second stage, the PSS analysis examines all the nets in your design to verify the accuracy of the extracted PSS fundamental frequency. This enhancement improves PSS analysis of circuits with frequency dividers by re-evaluating the PSS fundamental to take into account the frequency division in your circuit.

After the initial transient phase is complete, the shooting interval begins. In this phase, the circuit is repeatedly simulated over one period while adjusting the length of the period and the initial condition to find the periodic steady-state solution.

In some cases when an autonomous PSS analysis does not converge after a few iterations, increasing the `tstab` interval makes convergence faster and easier. Adjusting the `tstab` interval might improve the starting point for the shooting interval by moving it such that signals are quiescent or changing slowly. This allows strongly nonlinear circuits to converge faster.

For details of the algorithm for autonomous PSS analysis, see "Autonomous PSS Analysis" on page 187.

Starting the Oscillator

When applying PSS analysis to oscillators, you need to start the oscillator, just as you would if you were simulating the turn-on transient of the oscillator using transient analysis. *The Designer's Guide to SPICE and Spectre* [kundert95] describes in some depth techniques for starting oscillators.

In summary, there are two techniques for starting oscillators, using initial conditions, or using a brief impulsive stimulus. Initial conditions would be provided for the components of the oscillator's resonator. If an impulsive stimulus is used, it should be applied so as to couple strongly into the oscillatory mode of the circuit, and poorly into any other long-lasting modes, such as those associated with bias circuitry.

Either way, after the trigger is applied to start the oscillator, it is important to allow the oscillator to run for a while before the shooting methods are applied to compute the steady-state result. To do so, specify an additional stabilization interval using the *tstab* parameter. In practice, an additional stabilization interval often improves convergence, especially when simulating high-Q oscillators.

Convergence Issues with Autonomous PSS Analysis

By their very nature, oscillators are quite nonlinear, which can cause convergence problems. If you experience convergence problems with autonomous PSS analysis, you should try one or more of the following suggestions.

■ As with non-autonomous PSS analysis, providing a larger value for t_{stab} generally improves convergence. While not always necessary, you occasionally need to set t_{stab} to a value equal to or greater than the time needed for the circuit to approximately reach steady-state.

■ Assure that whatever method you choose to start your oscillator is effective. It should "kick" the oscillator hard enough to start the oscillation and have it respond with a signal level that is between 25% and 100% 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.

■ Give a better estimate of the period. The simulator uses the estimate of the period you specify to determine how long the initial transient analysis interval is that is used to make a measurement of the oscillation period. If the period you specify is too short, it is not possible to get an accurate estimate of the oscillation period, and this might cause the PSS analysis to fail. Overestimation of the period is not a serious problem because the only penalty is a longer initial transient interval. However, significant overestimation might result in an excessively long simulation time.

■ While trapezoidal rule ringing is annoying in transient analysis, in PSS analysis it can cause the shooting iteration to stall before convergence is achieved. You can resolve this problem by changing the `method` from `traponly` to either `trap`, `gear2`, or `gear2only`.

Generally, using `method=traponly` is preferred with oscillators because it uses the trapezoidal rule exclusively, which does not exhibit artificial numerical damping like the other available methods [kundert95]. However, 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. With `method=trap`, Spectre® circuit simulator RF analysis (Spectre RF) occasionally takes a backward Euler step which acts to damp the ringing. The other two methods, `gear2` and `gear2only`, use Gear's backward difference method which is not subject to ringing. Each of these alternatives to *traponly* avoid trapezoidal rule ringing and the attendant convergence problems at the expense of adding a small amount of artificial numerical damping, which acts to reduce slightly the computed Q of the oscillator.

■ Increase the maximum iterations for shooting methods using the `maxperiods` parameter. Sometimes the analysis might need more than the default number of iterations to converge. This is more likely to occur with high-Q circuits. However, there are situations that prevent convergence regardless of how many iterations are taken. In this case, increasing the iteration limit simply results in the simulator taking longer to fail.

■ Sometimes loosening or tightening the tolerance for the linear solver (the `tolerance` parameter) can solve convergence problems. This tolerance does not affect the accuracy of the final solution.
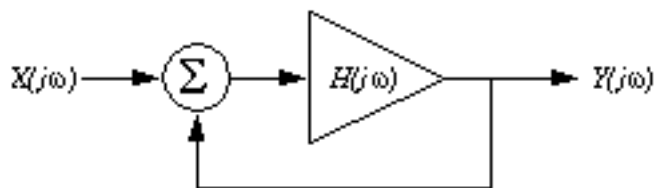
■   If the shooting iteration approaches convergence and then stalls, it might be because the tolerance on the shooting method is too tight. In this case, try loosening `steadyratio`, which controls how close to periodic the result must be before it is declared to be converged.

■   Finally, tightening the normal simulation tolerances (`maxstep`, `reltol`, `lteratio`, `errpreset`) can help resolve convergence problems in PSS analysis. Avoid using `errpreset=liberal`.

Be aware that in order to avoid the equilibrium solution (the one where all waveforms are constant valued) autonomous PSS analysis is constrained to find a result with nonzero amplitude. Thus, if the circuit cannot sustain an autonomous oscillation, and the equilibrium solution is the only solution that exists, PSS analysis fails to converge.

Phase Noise in Oscillators

Oscillators exhibit a natural tendency to amplify any noise present near the oscillation frequency. The closer the frequency of the noise is to the oscillation frequency, the greater the amplification. Noise amplified by the oscillator in this manner is referred to as *phase noise*. Phase noise is extremely important because it is the most significant source of noise in oscillators. In addition, because the phase noise is centered about the oscillation frequency, it can never completely be removed by filtering.

Phase noise can be understood by recognizing that the phase of an oscillator is arbitrary simply because there is no drive signal present 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 causes the phase to be disturbed, there is nothing that acts to restore the phase, so it is free to drift without bound. If the perturbation is random noise, the drift takes the form of a random walk. Furthermore, the closer the frequency of perturbation is to the oscillation frequency, the better it couples to the phase and the greater the magnitude of the drift. The perturbation need not come from random noise. It might also couple into the oscillator from other sources, such as the power supplies.

To better understand this phenomenon, consider the feedback oscillator shown. The loop gain of the oscillator is $H(j\omega)$. $X(jw)$ is taken to represent some perturbation stimulus and $Y(j\omega)$ is the response of the oscillator to X. The Barkhausen condition for oscillation states that the effective loop gain equals unity and the phase equals 360 degrees at the oscillation frequency ω_0 [clarke-hess]. The gain from the perturbation stimulus to the output is

$$Y(j\omega) \,/\, X(j\omega) = H(j\omega) \,/\, (1 - H(j\omega))$$

which goes to infinity at the oscillation frequency ω_0. When the perturbation stimulus is a noise source, the result is phase noise.

The amplification near the oscillation frequency that generates the phase noise is quantified by assuming the loop gain varies smoothly as a function of frequency in this region [razavi96]. If

$$\omega = \omega_0 + \Delta\omega$$

then

$$H(j\omega) \approx H(j\omega_0) + dH/d\omega \, \Delta\omega$$

and the transfer function becomes

$$Y(j(\omega_0 + \Delta\omega)) \,/\, X(j(\omega_0 + \Delta\omega)) \approx (H(j\,\omega_0) + dH/d\omega \, \Delta\omega) \,/\, (1 - H(j\omega_0) - dH/d\omega \, \Delta\omega).$$

Because $H(j\omega_0) = 1$ and $dH/d\omega \, \Delta\omega \ll 1$ in most practical situations, the transfer function reduces to

$$Y(j(\omega_0 + \Delta\omega)) \,/\, X(j(\omega_0 + \Delta\omega)) = \text{-1} \,/\, (dH/d\omega \, \Delta\omega)$$

Thus, for circuits that contain only white noise sources, the phase noise voltage (or current) is inversely proportional to $\Delta\omega$, while the phase noise power spectral density is proportional $1/\Delta\omega^2$ near the oscillation frequency.

Phase noise is created by a linear phenomenon, the amplification of noise near the carrier frequency that is a natural consequence of the oscillator's complex pole pair on the $j\omega$ axis at ω_0. In addition, practical oscillators are also strongly nonlinear because they operate in

compression. Recall that the loop gain of the oscillator must equal 1 at the oscillation frequency. If the loop gain is less than 1, the oscillation eventually dies out. To ensure the oscillator reliably oscillates even with normal variations in component values, the initial loop gain is always designed to be larger than 1, with the understanding that as the amplitude of the oscillation builds the amplifier goes into compression, which reduces the loop gain. The amplitude stabilizes at the point where the effective loop gain is 1. The nonlinear behavior inherent in practical oscillators results in noise folding. Noise mixes with the oscillation signal and its harmonics and get converted up or down in frequency by one or more multiples of the fundamental frequency. Besides the effect noise folding has on the noise level, it also introduces two noteworthy and easily identifiable artifacts.

First, the phase noise present near the oscillation frequency is mixed up and down, resulting in smaller, but still quite noticeable noise peaks at DC and each harmonic.
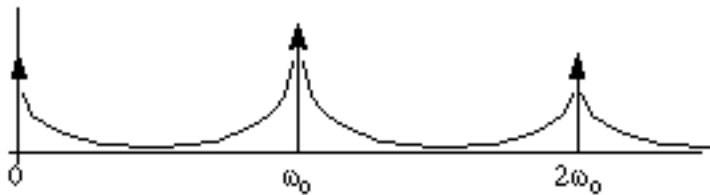
Second, if there are flicker noise sources present in the circuit, the flicker noise mixes up from low frequencies and results in additional noise near the oscillation frequency and its harmonics. Thus, for circuits that contain flicker noise sources, the noise power spectral density is proportional to $1/\Delta\omega^3$ near the oscillation frequency.
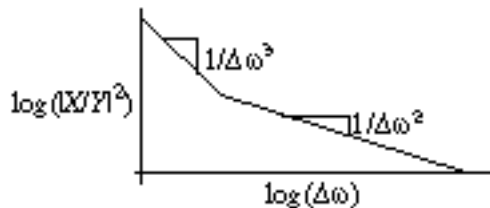
Phase noise is strongly affected by the Q of the oscillator. The higher the Q, the lower the phase noise. To see this, let $H(j\omega) = A(\omega) \exp(jF(\omega))$. Then

$$|Y/X|^2 = 1 / \{\Delta\omega^2[(dA/d\omega)^2 + (d\Phi/d\omega)^2]\}.$$

Razavi extends the traditional definition of Q to make it more insightful for ring and relaxation oscillators. Open loop Q is

$Q = \omega_0$ sqrt$\{(dA/d\omega)^2 + (d\Phi/d\omega)^2\}$ / 2.

In a resonant oscillator, $dA/d\omega = 0$ and so the open loop Q reduces to the traditional definition, $Q = \omega_0$ / $2(d\Phi/d\omega)$. For ring and relaxation oscillators, $dA/d\omega$ and $d\Phi/d\omega$ are of the same order and this new definition is more appropriate. The open loop Q is a measure of how much the closed-loop system opposes variations in the frequency, as seen with

$|Y/X|^2 = (\omega_0/\Delta\omega)$ / $(4Q^2)$

Sample Spectre RF Circuits

The following sections discuss two possible uses for Spectre RF, to simulate

■   High-Performance Receivers

■   Switched-Capacitor Filters

Important

> The example circuits discussed in this section are for illustrative purposes only. The circuits and netlists are not available.

Characterizing a High-Performance Receiver

This receiver was characterized using the sequence of analyses shown in the netlist in Example 3-1 [telichevesky96a]. Each analysis, along with its results, is described in the following sections. The analyses were run on an HP 735/125.

Example 3-1 Netlist Showing the Analyses Run on the Receiver

```
// compute steady state response to LO only
disableRF alter dev=Prf param=type value=dc
loAlone pss fund=780MHz

// compute transfer functions from all sources to output
freqResponse (out 0) pxf start=20kHz stop=200MHz lin=50
        maxsideband=3

// compute output & input-referred noise, & noise figure
noise pnoise start=2kHz stop=200MHz oprobe=Rl iprobe=Prf
        refsideband=1 maxsideband=15

// compute intermod distortion with LO and 2 input tones
enableRF alter dev=Prf param=type value=sine
interMod pss fund=10MHz swapfile="bigjake" outputtype=both

// compute harmonic distortion with LO and 1 input tone
disable2ndRFtone alter dev=Prf param=ampl2 value=0
harmDisto pss fund=60MHz harms=30

// apply second small signal tone to measure intermod
interModAC pac start=850MHz sidebands=[-15 -13]
```

The large RF circuit shown in Figure 3-1 on page 333 is an image rejection receiver that consists of a low-noise amplifier, a splitting network, two double-balanced mixers, and broadband Hilbert transform output filter combined with a summing network that is used to suppress the undesired sideband. A limiter in the LO path is used for controlling the amplitude of the LO.

**Figure 3-1 Block Diagram for a High-Performance Receiver**
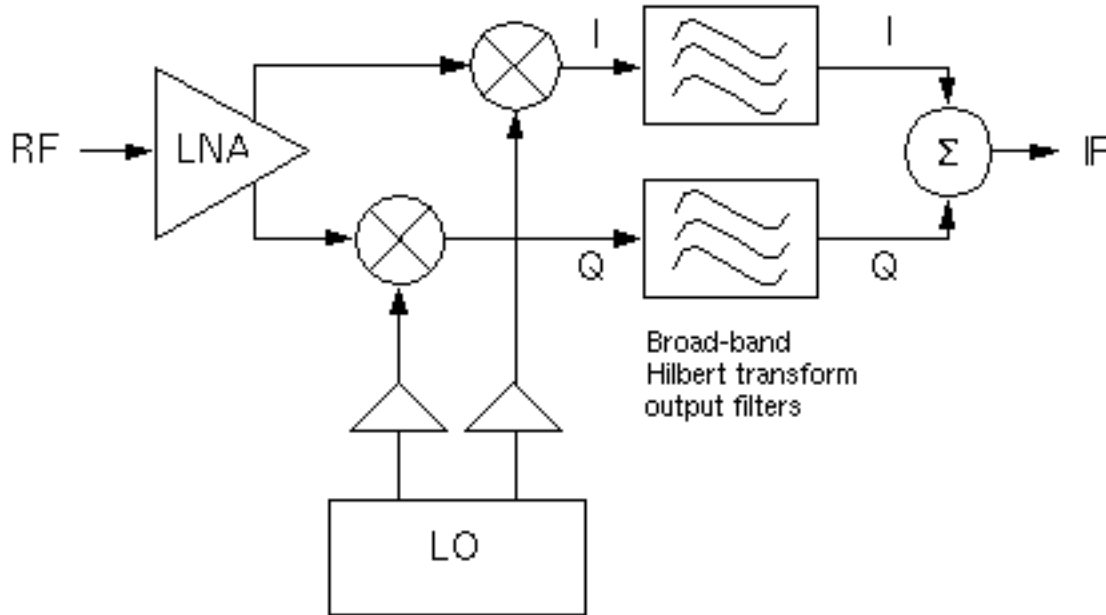
The receiver contains 167 bipolar transistors and uses 378 nodes. It is simulated at the circuit level using the Gummel-Poon model for each transistor. It generated 986 equations (largely because of the large number of inductors but also because the internal nodes of the BJTs needed to support their parasitic resistors).

PSS Analysis

This analysis computes the steady-state response of the circuit with only the LO applied. This is a necessary step that allows the use of the periodic small-signal analyses later. The *alter* statement that precedes the PSS analysis is used to disable the RF input.

Waveforms computed by this analysis are shown in <u>Figure 3-2</u> on page 334. The response of the receiver computed by PSS analysis is *loAlone* at the LO input and it appears in the top half Figure <u>3-2</u>. The response of the receiver at the output appears in the bottom half Figure <u>3-2</u>.

Figure 3-2 Response of the High-Performance Receiver for the PSS Analysis
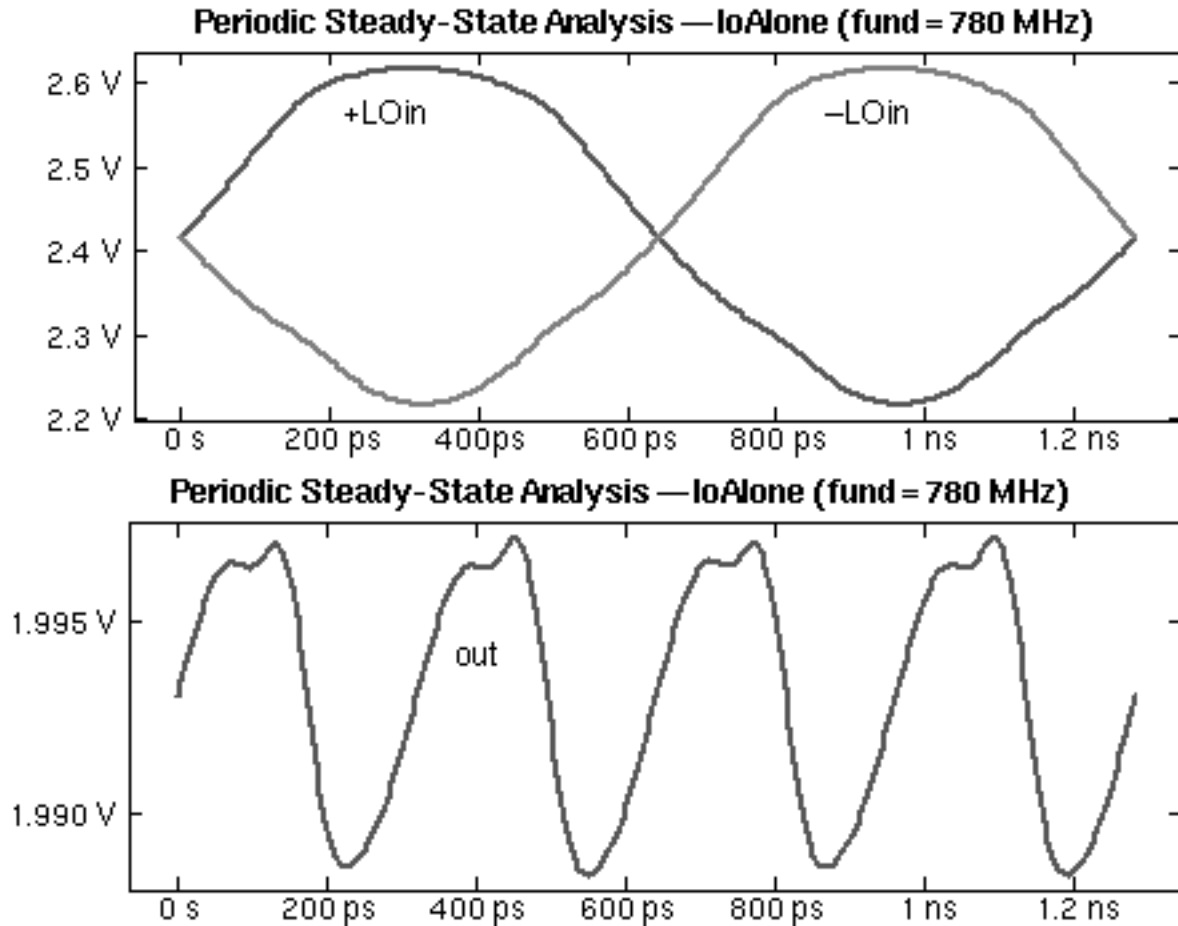
The Spectre RF simulation required about 30 seconds to complete this analysis. The PSS analysis used 200 time points and needed about 25 MB, or about 125 b per equation per time point.

PXF Analysis

The results from the PXF analysis *freqResponse* are shown in Figure 3-3 on page 335. Each transfer function from the RF input to the IF output represents a different sideband. Unlike traditional transfer functions as computed by AC analysis, these transfer functions can have inputs and outputs at different frequencies.

The transfer function from the RF input V_{rf} to the output voltageV(out) appears in the top half of Figure 3-3. The bottom half of Figure 3-3 is an expanded version of the top that clearly shows the conversion due to the ±1 harmonics of the LO.

Figure 3-3 Transfer Function From the Vrf to V(out)
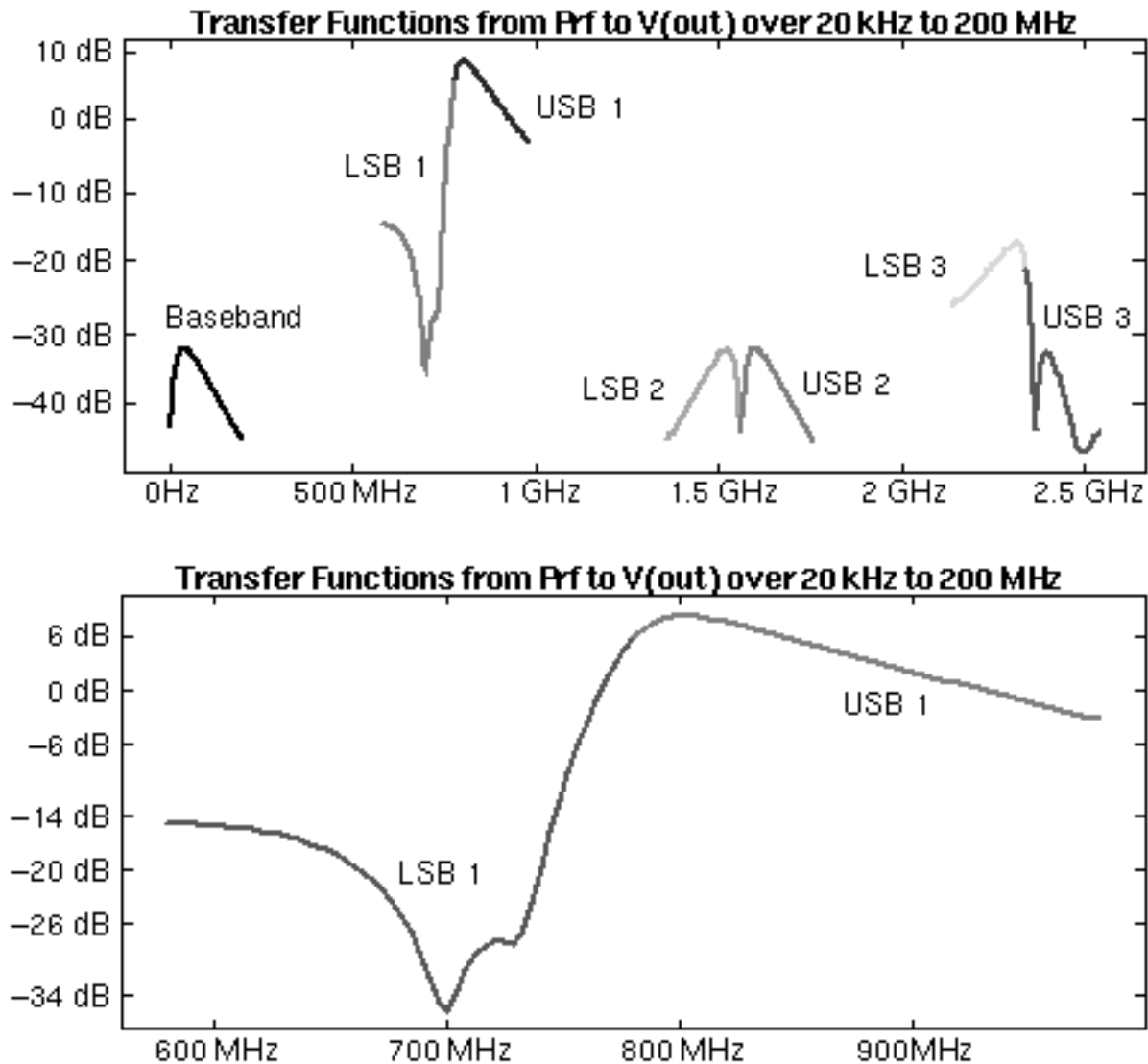
The X axis is labeled with the input frequency (by setting `freqaxis` to `in`), although you can also view the transfer functions versus the output frequency (set `freqaxis` to `out`). The output frequency range for all of the transfer functions shown is baseband, which is being swept from 20 kHz to 200 lHz. For example, $Vrf(2)$ is the transfer function from the second

upper sideband to baseband, and finally, $Vrf(0)$ is the transfer function from baseband to baseband.

Consider the dip at 700 MHz in $Vrf(-1)$, at which point the receiver exhibits 28 dB of attenuation. This implies that a signal from Vrf at -700 MHz mixes with the -1st harmonic of the LO and shows up attenuated at the output at

$f_{out} = -700\ MHz + 780\ MHz = 80$ MHz

Similarly, an 800 MHz signal at the input would mix with the 1st harmonic of the LO and would appear at the output at

$f_{out} = 800\ MHz - 780\ MHz = 20$ MHz

amplified by 15 dB. This strong asymmetry in the transfer function on either side of the LO is a clear indication that this receiver is designed to suppress the lower sideband.

<u>Figure 3-4</u> on page 337 shows the transfer functions to the output voltage, V(out), from V_{lo} and V_{cc}. The top half of the figure shows the transfer function to V(out) from the LO input, V_{lo}. The bottom half of the figure shows the transfer function to V(out) from the power supply, V_{cc}. This shows how sensitive the output is at the specified range of frequencies to noise on the LO or the power supply at various frequencies.

Figure 3-4 Transfer Functions From V_{lo} and V_{cc} to V(out)
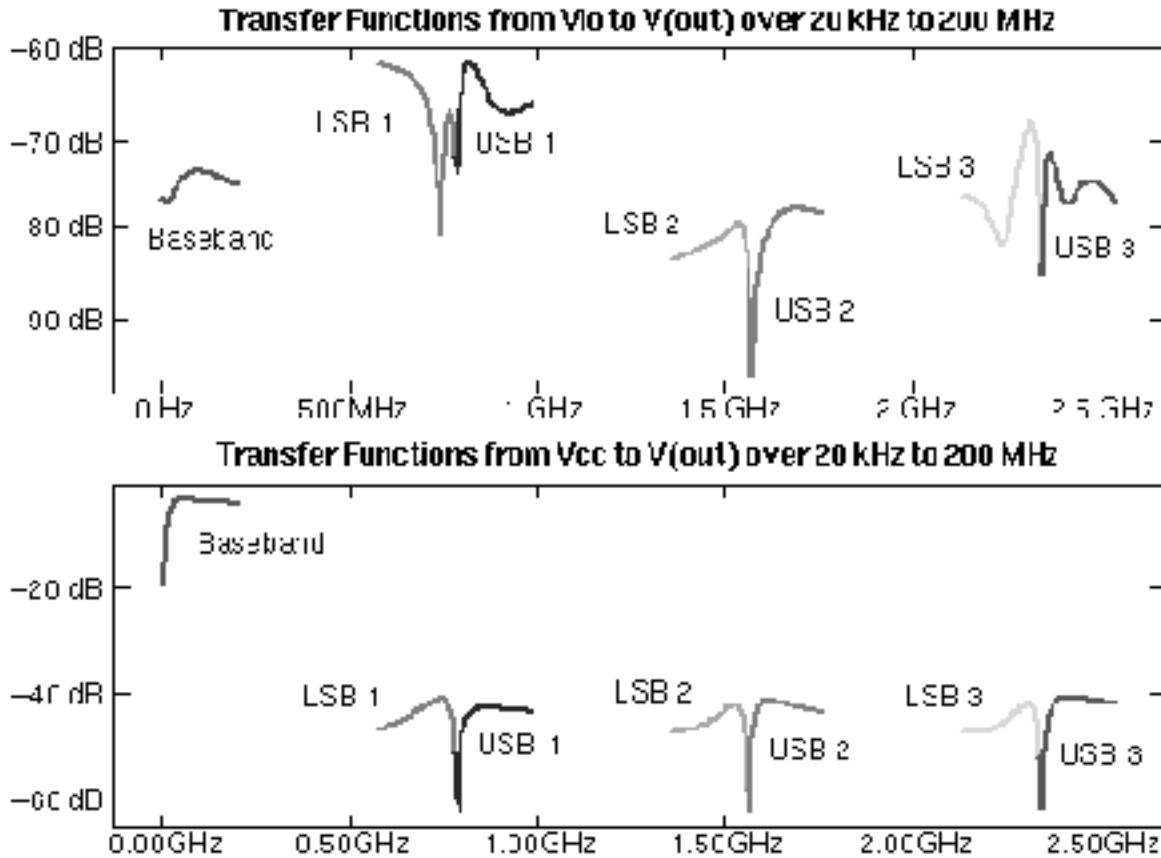
Spectre RF simulation required 6 minutes to complete this analysis, which contained 50 frequency points, or roughly 7 seconds per point. It reused the 25 MB of memory allocated in the PSS analysis l*oAlone*.

PNoise Analysis

The PNoise analysis is similar to the traditional noise analysis, except it includes the effect of noise moving from one frequency to another as it mixes with the LO and its harmonics. In this case, the noise measured at the output includes contributions from noise generated at frequencies $f_{out} \pm k\, f_{LO}$, where $k \leq 15$.

This noise analysis is performed on the detailed periodically varying linearized circuit, and so it includes subtle effects in the calculation. For example, at the points in time where the LO signal is near 0 V, the differential pairs that make up the mixer are balanced and so exhibit a

great deal of gain from the LO to the output. Any noise on the LO is greatly magnified at this time. After the LO moves away from 0, the differential pairs saturate and so exhibit a great deal of attenuation from the LO to the output. The faster the LO moves through zero, the less noise is transferred from the LO to the output. This is why switching mixers are preferred over multiplying mixers. This effect is accurately accounted for by the *PNoise* analysis.

The output noise, input-referred noise, and the noise figure as computed by the PNoise analysis are shown in Figure 3-5 on page 338. The outpupt noise at V(out) and the input-referred noise from V_{rf} appear in the top of the figure. The noise figure from V_{rf}, assuming 50 W source resistance appears in the bottom of the figure. The flicker or 1/f noise of the devices is clearly visible.

Figure 3-5 Output Noise, Input-Referred Noise, and Noise Figure
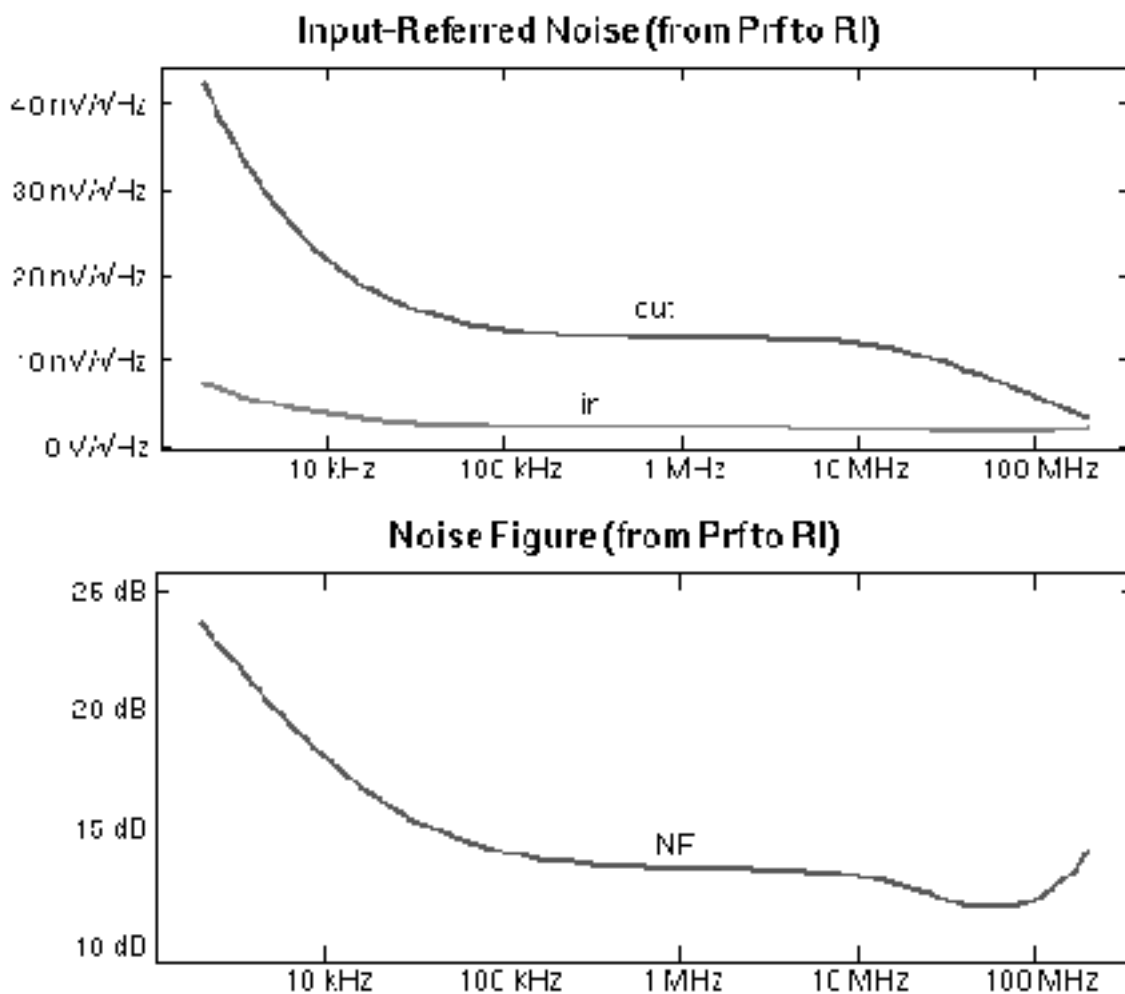
Spectre RF simulation required 8 minutes to complete this analysis, which contained 50 frequency points, or roughly 10 seconds per point. It reused the 25 MB of memory allocated in the PSS analysis l*oAlone*.

PSS Analysis interMod for Intermodulation Distortion

To measure the intermodulation distortion of a mixer, you can apply two tones to the input that are relatively close in frequency. The two tones must be co-periodic with each other and with the LO in order to perform a Fourier analysis. The steady-state response is then found, and the mixer is simulated for a time interval equal to the common period of the three input signals.

In order to compute the steady-state response, you can apply transient analysis until the initial transients decay to negligible levels, or you can apply the PSS analysis. If the transients decay quickly, transient analysis is generally preferred, because it is fast and does not require much memory. Otherwise, PSS analysis is usually preferred.

In this circuit, the LO frequency is 780 MHz and two 25 mV input signals were applied at 840 MHz and 850 MHz. The PSS analysis period was set to 100 ns. A Fourier analysis was performed and 12 harmonics were requested. The nonperiodicity reported by the Fourier analysis was only 61 nV, indicating that if any transients existed, they were very small. The results are shown in <u>Table 3-1</u> on page 339.

Table 3-1 Output Spectrum as Computed by PSS Analysis

Freq	Absolute Level	Relative Level
10 MHz	237.146 µV	–52.70 dB
20 MHz	51.9117 µV	–65.88 dB
30 MHz	37.7329 µV	–68.65 dB
40 MHz	53.1112 µV	–65.69 dB
50 MHz	223.043 µV	–53.22 dB
60 MHz	102.192 mV	0 dB
70 MHz	94.0778 mV	–0.72 dB
80 MHz	158.248 µV	–56.20 dB
90 MHz	23.4351 µV	–72.79 dB
100 MHz	27.342 µV	–71.45 dB
110 MHz	7.80743 µV	–82.34 dB

The third-order intercept point is computed using simple geometric reasoning [kundert95].

(3-1) $\quad IP_3 = \dfrac{3V_1 - V_3}{2}$

In Equation 3-1,

- V_1 is the magnitude of the first fundamental signal at 60 MHz

- V_3 is the magnitude of the first IM_3 term at 50 MHz

Both voltages are measured in dBV.

From Table 3-1, $V_1 = -19.81$ dBV and $V_3 = -73.03$ dBV, which indicates that the intercept point is 6.8 dBV. To confirm this number, the input amplitude was stepped over a range of values, and the amplitude of the fundamental (at 60 MHz) and the third-order intermodulation term (at 50 MHz) were plotted in Figure 3-6. The curves were extrapolated to compute

$IP_3 = 7.1$ dB

In Figure 3-6, the third-order intercept point of the receiver was computed by extrapolating actual measurements at small-signal levels. The measured result is either 6 or 7 dBV depending on whether the upper or lower sidebands are used. The IM_3 curves bend up at low levels due to small levels of numerical error generated by the simulator. This problem could be reduced at the expense of longer simulations and increased memory requirements by tightening simulator tolerances.

Figure 3-6 Third Order Intercept Point of the Receiver
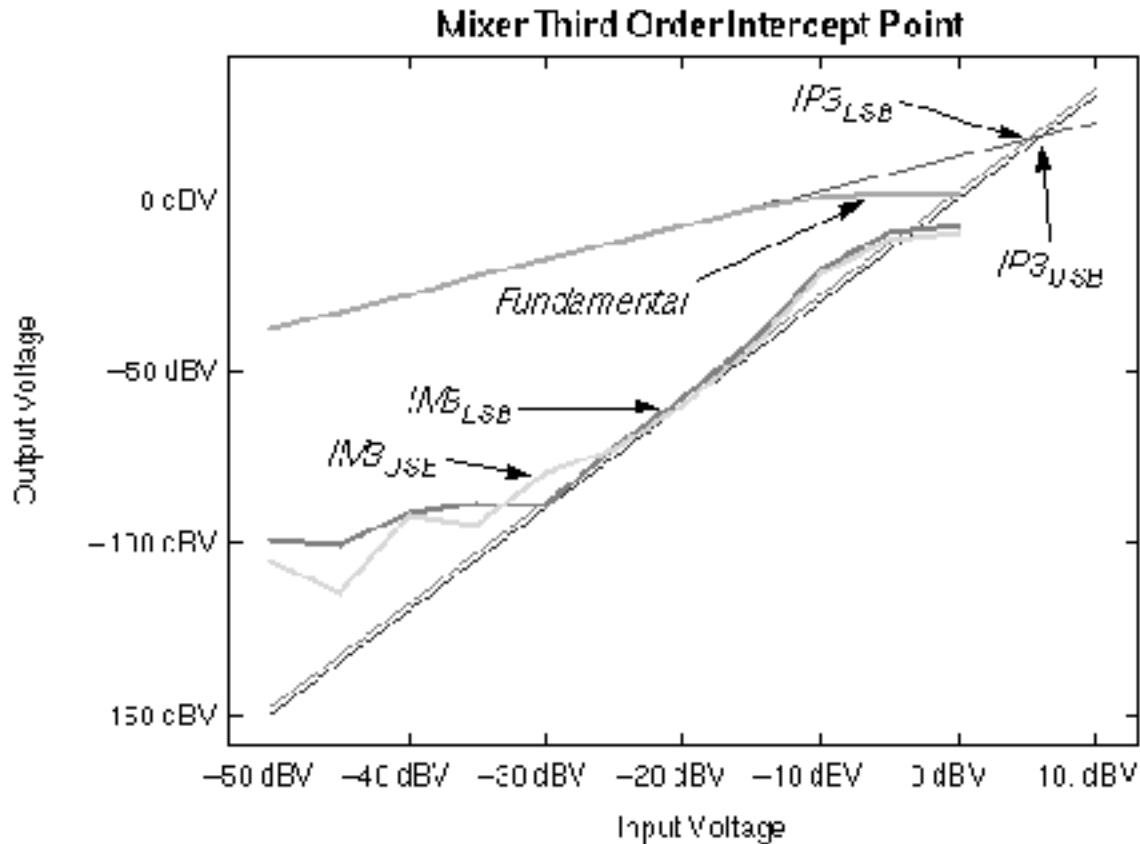
Spectre RF simulation required 15 minutes to complete the analysis. It used roughly 5,000 timesteps. At 125 b per equation per timestep, PSS would require over 600 MB of swap space to run. The computer used had only 1 GB of free disk space. The `swapfile` parameter of the PSS analysis was used to direct the simulator to use a conventional file rather than virtual memory to hold the periodically varying linear representation of the circuit. It used 430 MB for the swapfile and 40 MB of virtual memory.

An important thing to keep in mind, especially when reading "PSS harmDisto and PAC interModAC for Intermodulation Distortion" on page 342, is that with the approach used in this section, both the simulation time and memory increase in inverse proportion to the difference in frequencies of the two test tones. In this case, the difference was large, and so the problem was tractable. However, if the frequency difference dropped to 1 MHz or below, this approach would have been impractical.

PSS harmDisto and PAC interModAC for Intermodulation Distortion

An alterative approach to computing the intermodulation distortion of the receiver is to apply the LO and only one of the two input test tones during the PSS analysis. Apply the second input tone as a small-signal during a PAC analysis. The benefit to this approach is that the simulation interval used during PSS analysis can be much shorter. In this circuit, if only the 840 MHz RF input and the 780 MHz LO are assumed to be large, then the simulation interval is shortened from 100 ns on the previous analysis to 16.67 ns on this one. That is a factor of 6 improvement. Instead of 15 minutes, the analysis takes 2.5 minutes and 93 MB to compute 787 time points. However, so far only enough information is available to compute the receiver's harmonic distortion, which is useful in this case because this receiver is broadband. In many cases the receiver is narrow band and so the ability to efficiently compute the intermodulation distortion is very important.

As a second step, apply a small 850 MHz signal and run PAC analysis. Because only one frequency point is needed from the PAC analysis, it is very efficient, requiring only 47 seconds to compute the perturbation to the PSS solution due to the small 850 MHz RF test signal.

IP3 is computed using V_{L1}, V_{S1}, and V_{S3}, where $V_{L1} = -19.81$ dBV is the magnitude at the output of first fundamental-signal at 60 MHz (computed using *harmDisto*), $V_{S1} = 11.46$ dBV is the magnitude at the output of the second fundamental-signal at 70 MHz (computed as the -13th sideband from *interModAC*), and $V_{S3} = -42.98$ dBV is the IM_3 term at the output at 50 MHz (computed as the -15th sideband from *interModAC*).

As shown in Equation 3-2, the third-order intercept point is computed using Equation 3-1 on page 340.

In other words

(3-2) $$IP_3 = V_{L1} + \frac{V_{s1} - V_{s3}}{2} = 7.42 dBV$$

The 0.3 dB deviation results primarily because the transfer function changes slightly between 840 MHz and 850 MHz, creating some uncertainty in the previous measurement. This uncertainty can be reduced in this measurement by reducing the frequency difference. It can be eliminated by reducing the frequency difference to zero. There is also a small difference between this result and those computed previously that is caused because there is only one large input tone rather than two. This is usually not significant when computing intercept points because the test tones are necessarily rather small. In addition, the various approaches to computing *IP3* suffer from small errors in magnitude of the harmonics computed by Spectre RF simulation. These errors can be traced to errors in the waveforms as computed by transient analysis. The errors are quite small relative to the other signals

present in the circuit, such as the LO, and are within tolerance. The default tolerances are used. The errors can be reduced considerably by tightening the simulators normal tolerances. The *IP3* calculation of Equation 3-2 on page 342 should be the least sensitive to these errors.

Characterizing a Switched-Capacitor Filter

Spectre RF simulation can simulate switched-capacitor filters at the transistor level. Spectre RF simulation naturally includes second-order effects such as finite bandwidth effects, nonlinear switch resistance effects, charge redistribution effects, slew-rate limiting effects, and back-gate bias effects. It also allows you to include parasitic resistors and capacitors back annotated from the layout in the simulation. Finally, because it simulates at the circuit level, Spectre RF simulation can accurately simulate switched-current filters. These circuits cannot be adequately simulated using traditional switched-capacitor filter simulators, because the imperfections that are second order effects in switched-capacitor circuits play a much more important role in switched-current filters. Most of these effects cannot be included when using traditional switched-capacitor filter simulators.

The switched-capacitor filter being simulated is a 5-pole 2.2 kHz low-pass Elliptic filter. The circuit contains 71 MOSFETs and generates 58 equations. The filter was characterized using the sequence of analyses shown in the following netlist. Each analysis, along with its results, is described in the following sections. The analyses were run on an HP 735/125.

Example 3-2 The Analyses Run on the Switched-Capacitor Filter

```
// PSS analysis (sets periodic operating point)
clockAlone pss fund=CLOCK_FREQ saveinit=yes readic="%C:r.ic"
        writefinal="%C:r.ic"
// Measure transfer functions
TFin pac stop=10kHz lin=200
// Use PXF to get all transfer functions (including PSR)
unsmpldTFall (out gnd) pxf stop=10kHz lin=200 maxsideband=5
smpldTFall (sout gnd) pxf stop=10kHz lin=200 maxsideband=5
// Measure noise
unsmpldNoise (out gnd) pnoise start=100 stop=25kHz
        maxsideband=25
smpldNoise (sout gnd) pnoise start=100 stop=25kHz
        maxsideband=25
```

PSS Analysis clockAlone

This analysis computes the steady-state response of the circuit with only the clock applied. The results of this analysis are used to measure the offset voltage at the output of the filter that results from the offset voltage of the op-amps and charge injection from the switches into the integrators. This analysis is also a prerequisite to the periodic small-signal analyses that follow because it sets the periodic operating point.

The PSS analysis, like a conventional transient analysis, starts off from an initial condition. If you do not specify an initial condition, then Spectre RF simulation uses a DC analysis to determine the initial condition. During a DC analysis, the clocks are not operating, so the integrators have no feedback, keeping their outputs stuck at the rails. The initial condition computed during the DC analysis causes the filter to react wildly in transient analysis as shown in Figure 3-7, bouncing off the rails several times before it settles. This results in convergence difficulties during PSS analysis that you can resolve several ways.

One approach that works in this case is to simply specify zero initial conditions for all capacitors in the circuit. This avoids a bad starting point. Another approach that is widely applicable when confronting convergence problems in PSS analysis is to use the `tstab` parameter to delay the start of the PSS shooting loop. In this example, `tstab` = 1.5 ms, which implies that transient analysis is performed to at least t = 1.5 ms before the PSS analysis attempts to compute the steady-state solution. Waiting until t = 1.5 ms allows the simulator to get beyond the difficult time when the filter is bouncing off the rails, which results in PSS analysis converging easily.

As shown in Figure 3-7, the transient behavior of an internal node in the filter shows clipping that results from using the DC operation point as the initial condition. The clipping causes convergence problems for PSS analysis. Convergence is easily achieved either by setting `tstab` = 1.5 ms so that the PSS analysis gets beyond the clipping, or setting the initial conditions on the capacitors to zero.

Figure 3-7 Transient Behavior of an Internal Node
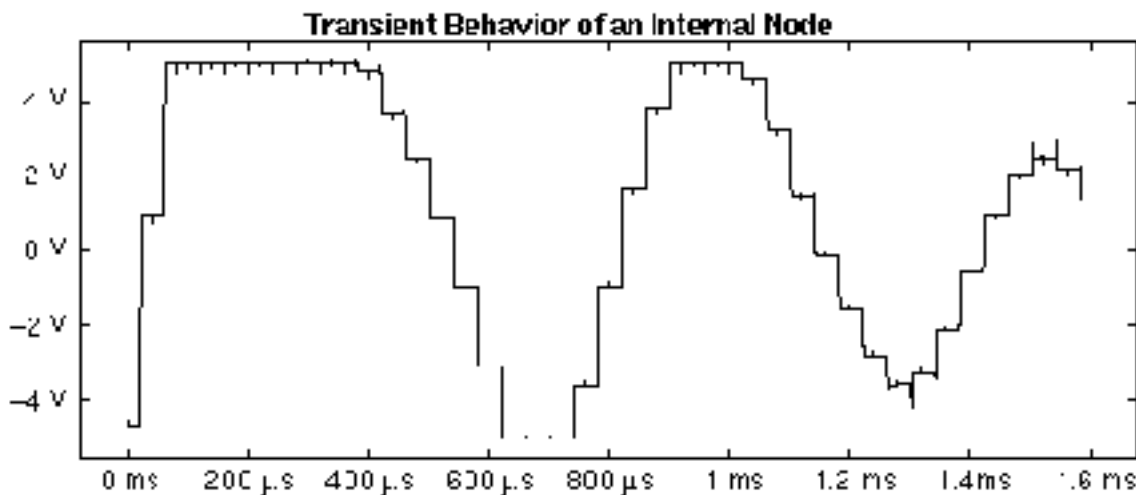
After the steady-state response has been computed, recomputing it is accelerated by saving the final point computed by the PSS analysis and using it as an initial condition for subsequent

PSS analyses. (Remember that because the steady-state response is periodic, the final point is the same as the initial point.)

In order to further improve the efficiency of the PSS analysis, the phasing of the clock signals relative to the simulation interval is carefully chosen. It is best to have the simulation interval begin and end at points where the signals are not changing abruptly. For example, the phasing shown in the top half of Figure 3-8 results in convergence in fewer iterations and less time than the phasing shown in the bottom half of the figure. In the top half of the figure, the signal is settled and unchanging immediately prior to the end of the analysis. Contrast this with the situation shown in the lower half of the figure, in which the waveform is changing abruptly at the end point.

**Figure 3-8 Choosing Begin and End Points for PSS Analysis**
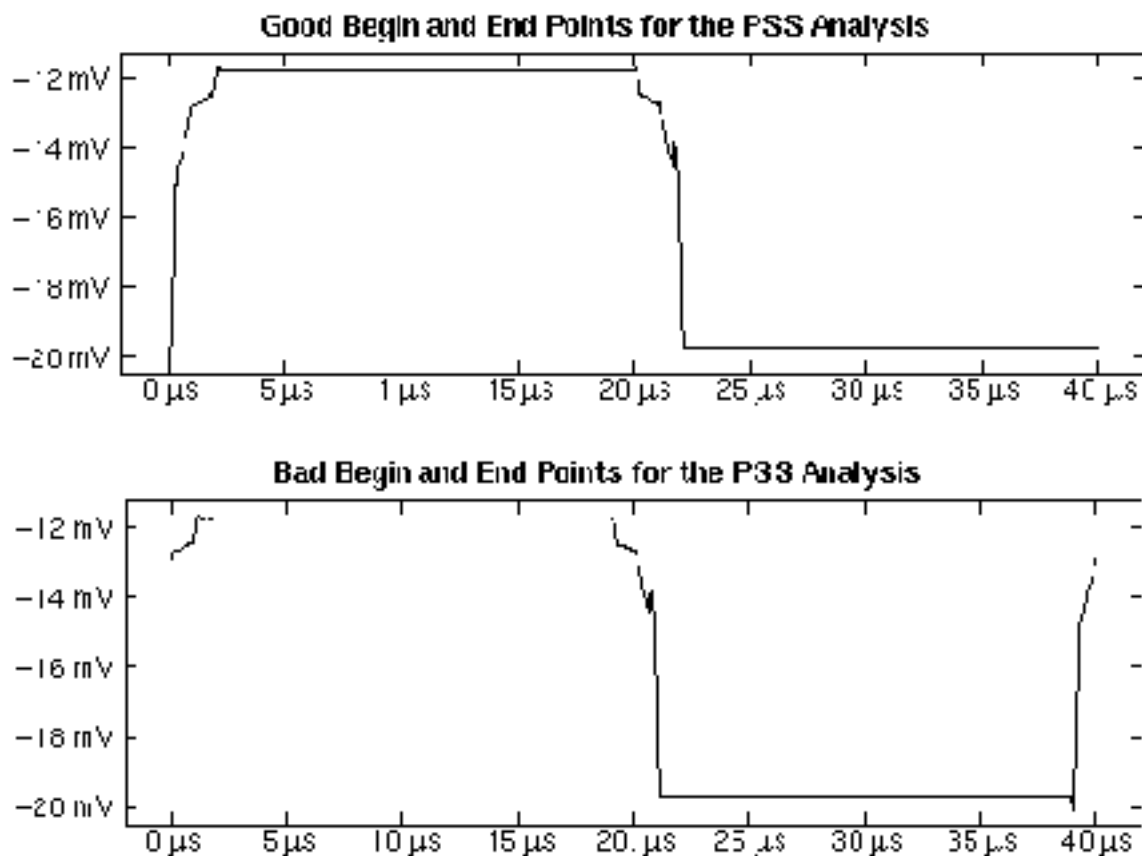
PAC Analysis TFin

The PAC analysis applies a small-signal at the input and computes the response at two outputs. The first output is the normal output of the filter. The signal at this output is continuous in time and includes various imperfections such as glitches and regions of slew-rate limiting and settling. It also contains output from both phases of filter. This output is interesting if the filter is followed with a continuous-time filter. The second output is the normal output after being passed through a sample-and-hold. This models the situation where the filter would be followed by an analog-to-digital converter (ADC) for further processing. In this case, most of the imperfections on the normal output are ignored by the sampling nature of the ADC. Taking into account the sampling nature of the ADC is important when trying to measure the transfer function, the noise, or the distortion of any clocked analog circuit such as a switched-capacitor filter.

With this circuit, only the magnitude and phase of the response at the fundamental frequency are interesting. The small input signal is considered to have unit magnitude and so the transfer functions are computed directly. The transfer functions are shown in <u>Figure 3-9</u> on page 347. Notice that the second null is missing from the transfer function of the normal (continuous-time) output. This is a consequence of blending the response from both phases of the filter.

Figure 3-9 Transfer Functions from the Input Voltage to the Outputs
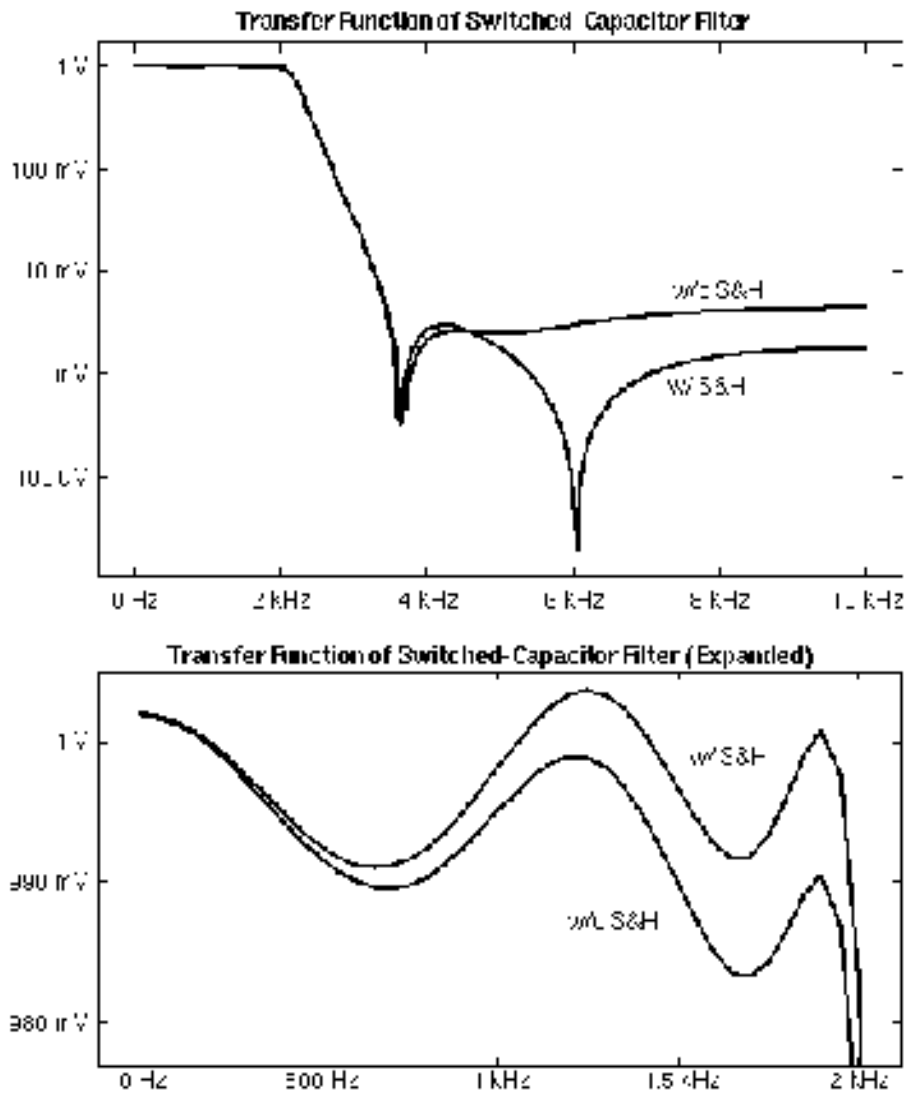
There are two possible outputs, each of which is used in a different application. The first is the normal output of the filter and the second is the normal output after being passed through a sample-and-hold.

The sample-and-hold was implemented in the Verilog-A language and is shown in the following netlist. This sample-and-hold is implemented in such a way that there is no hidden state that would prevent you from using PSS analysis. Both the delay and the aperture must be smaller than the period.

Example 3-3 Netlist Showing Sample-and-Hold Implemented Without Hidden State

```
module sh (Pout, Nout, Pin, Nin) (period, delay, aperture,
    tc)
node [V,I] Pin, Nin, Pout, Nout;
parameter real period=1 from (0:inf);
parameter real delay=0 from [0:inf);
parameter real aperture=1/100 from (0:inf);
parameter real tc=1/500 from (0:inf);
integer n; real start, stop;
node[V,I] hold;
    analog {

// Determine time aperture begins

        n = ($time() - delay + aperture) / period + 0.5;
        start = n*period + delay - aperture;
        $break_point( start );

// Determine time aperture ends

        n = ($time() - delay) / period + 0.5;
        stop = n*period + delay;
        stop = n*period + delay;

// Implement switch with effectively 1 Ohm series resistance

        if (($time() > start) && ($time() <= stop))
            I(hold) <- V(hold) - V(Pin,Nin);
        else
            I(hold) <- 1.0e-12 * (V(hold) - V(Pin,Nin));

// Implement capacitor with an effective capacitance of tc

        I(hold) <- tc * dot(V(hold));

// Buffer output

        V(Pout,Nout) <- V(hold);

// Control time step tightly during aperture

        if (($time() >= start) && ($time() <= stop))
            $bound_step( tc );
        else
            $bound_step( period / 5 );
    }
}
```

PXF Analyses unsmpldTFall and smpldTFall

These analyses directly compute the transfer function from all sources to either the unsampled or sampled output. They differ from the PAC analysis *TFin* in that they compute transfer functions from all sources to a single output rather than the transfer functions to all outputs from a single input. They are also about 50% slower than the PAC analysis. They are useful when measuring the ability of the filter to reject unwanted signals on the input, the clock, and on the supplies.

PNoise Analyses unsmpldNoise and smpldNoise

These analyses compute the total spot noise as a function of frequency at the sampled and unsampled outputs. The results are shown in Figure 3-10. They include noise folding (noise being converted down from sidebands of the clock by the sampling nature of the switched-capacitor filter).

Figure 3-10 Noise Response of Switched Capacitor Including Noise Folding
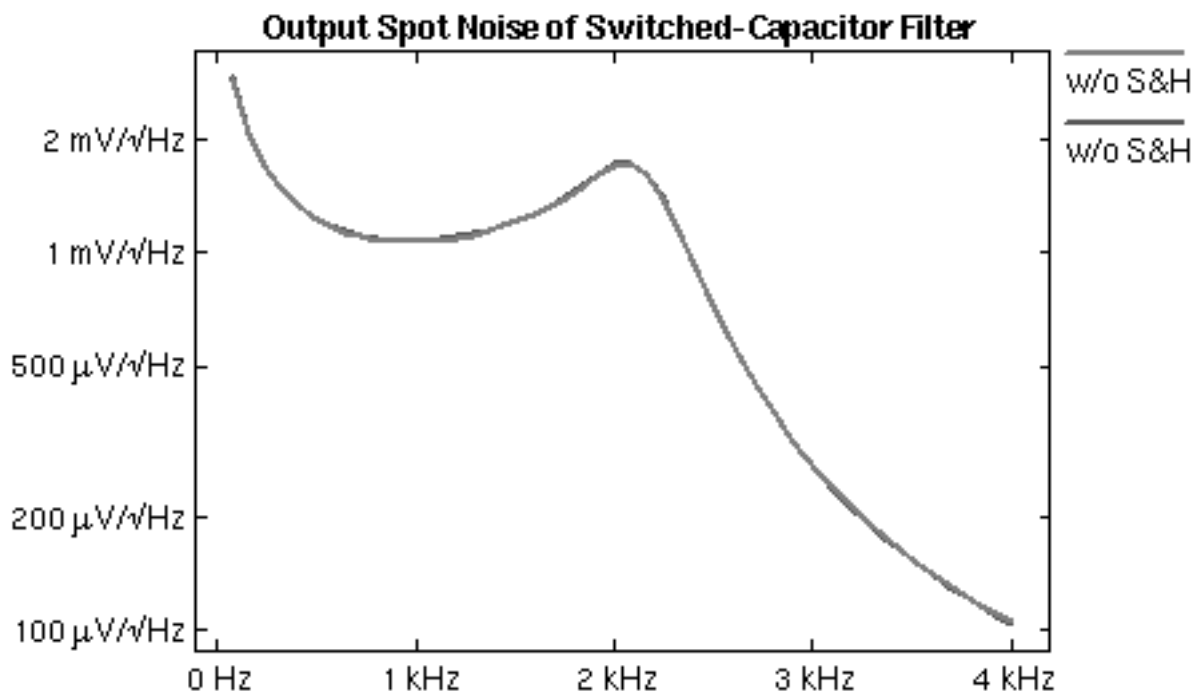
References

Clark-Hess 71   Kenneth K. Clarke, Donald T. Hess. *Communication Circuits: Analysis and Design.* Addison-Wesley Publishing Company, 1971.

Konrath 96   W. Konrath, H Brauns. "First fully CAE of a K-band sampling phase detector using periodic steady-state analysis and sophisticated SRD modeling." *Proceedings of the 1996 European Microwave Conference*, September 1996.

Kundert 90   Kenneth S. Kundert, Jacob K. White, Alberto Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog And Microwave Circuits.* Kluwer Academic Publishers, Boston 1990.

Kundert 94   Kenneth S. Kundert. "Accurate Fourier Analysis for Circuit Simulators." *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, May 1994.

Kundert 95   Kenneth S. Kundert. *The Designer's Guide to SPICE and Spectre.* Kluwer Academic Publishers, Boston 1995.

Razavi 96   Behzad Razavi. "A study of phase noise in CMOS oscillators." *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, March 1996.

Telichevesky 95   Ricardo Telichevesky, Kenneth S. Kundert, Jacob K. White. "Efficient Steady-State Analysis based on Matrix-Free Krylov-Subspace Methods." *Proceedings of the 32rd Design Automation Conference*, June 1995.

Telichevesky 96a   Ricardo Telichevesky, Kenneth S. Kundert, Jacob K. White. "Receiver Characterization using Periodic Small-Signal Analysis." *Proceedings of the IEEE 1996 Custom Integrated Circuits Conference*, May 1996.

Telichevesky 96b   Ricardo Telichevesky, Kenneth S. Kundert, Jacob K. White. "Efficient AC and Noise Analysis of Two-Tone RF Circuits." *Proceedings of the 33rd Design Automation Conference*, June 1996.

4

Semi-Autonomous Simulation and Small Signal Analysis

The semi-autonomous simulation is used to simulate driven oscillators with one or more periodic inputs. Now the algorithms can handle two typical driven oscillator scenarios:

Oscillator-Mixer co-simulation

Now we can simulate the oscillator and mixer together in the harmonic balance (HB) analysis. When a mixer is connected to an oscillator, it may introduce some effects to the oscillator, such as change of oscillating frequency and output amplitude drop due to loading effects, etc. Previous harmonic balance methods cannot accurately capture these effects because they cannot simulate the oscillator and mixer together. Using the semi-autonomous technique, we can simulate the oscillator and mixer together, and take the coupling effects between the oscillator and the mixer into consideration, resulting more accurate simulation. After the semi-autonomous simulation, we can perform corresponding semi-autonomous small signal analysis, like HBNOISE.

Driven oscillator simulation

Using the semi-autonomous simulation, we can simulate unlock driven oscillators (e.g., the oscillator with a power supply interference). When an oscillator is perturbed, but not locked by an injected periodic signal, its frequency and amplitude will change. The semi-autonomous simulation can handle the perturbed oscillator and give us the new oscillating frequency and waveforms. When the semi-autonomous simulation finishes, we can also perform small signal analysis (HBAC/HBXF/HBNOISE) on the semi-autonomous system, just like a normal multi-tone system.

The semi-autonomous simulation is a frequency domain simulation technique, and it is implemented in the Harmonic Balance (HB) analysis.

Initialization

Now we maintain good consistency between the autonomous PSS simulation and the semi-autonomous simulation. We can use all initialization methods for the autonomous PSS simulation in the semi-autonomous simulation.

tstab initialization method

In this method, Spectre runs couple oscillator periods of transient simulation and use the waveforms as the initial guess of the semi-autonomous simulation. The length of the transient simulation is determined by the "tstab" parameter in the HB statement. The user needs to make sure that the "tstab" is long enough to guarantee the oscillation can start properly.

The user can do one of the following to speedup the oscillation:

1. Specify an initial condition to one of the oscillator node (on the tank). For example, "ic outp=1.0".

2. Apply a pulse or PWL current/voltage source to the oscillator to kick start the oscillation.

It is highly recommended that the first method is used. The second method can prolong the transient simulation, and an over-strong pulse may cause transient convergence problem.

For a better semi-autonomous convergence, the "tstab" needs to be long enough to make the oscillator settle down the final oscillating mode. The integration method of the tstab simulation is specified by the HB option "tstabmethod".

Linear initial condition aided tstab

A linear Eigenvalue analysis can also be performed on the DC solution of the oscillator circuit to stimulus the tstab transient simulation. Using the linear IC method, the user doesn't need to specify the stimulus (you cannot use the PWL/Pulse kicker in linear IC, or linear IC will fail), and the tstab time can be reduced dramatically to save the simulation time. The linear IC works well for high Q oscillators, however, it may fail if the oscillator is not LC type, or the circuit is very large and ill-conditioned.

The linear initial condition method is turned on by the HB option "oscic=lin". By default, the linear method is not activated. In the semi-autonomous analysis, the "tstab" is always required, even though the "oscic=lin" is specified. "tstab=0" doesn't work for semi-autonomous simulation now (but we will improve the initialization code to make it work in the future Spectre version). It

is not recommended to use linear IC for oscillator-mixer simulation in this stage, because the circuit can be very large and the mixer may cause the linear IC method to fail.

Newton Methods

Just like the autonomous PSS simulation, the semi-autonomous simulation supports both the "onetier" and "twotier" Newton methods. The Newton method can be specified using the "oscmethod" option. Please refer to the autonomous simulation application notes for the detailed explanation of the different Newton methods. We suggest customers to try the "onetier" method first, because the "twotier" semi-autonomous simulation can be slow.

Use Model

The semi-autonomous simulation is for oscillators with periodic inputs. Users can use the method to simulate oscillators and mixers together, or simulate oscillators perturbed by periodic external interferences, such as power supply interference.

PSS simulation for semi-autonomous circuits

Now we can use the harmonic balance (HB) do semi-autonomous simulation. To activate the method, we need to add two oscillator output terminals in the HB statement. For example:

```
Hb1 (outp outn) hb funds=[1.1GHz  "frf"] maxharms=[5 5] tstab=1u
```

Compared to the driven multi-tone HB, the semi-autonomous statement has two differences:

1. The two terminals have to be defined as in autonomous PSS.

2. The first entry in "funds" is the initial guess of the oscillator frequency.

Small signal analysis for semi-autonomous circuits

A following small signal analysis statement will do the small signal analysis for semi-autonomous circuits. For example:

```
Hbnoise1  (oscoutp oscoutn) hbpnoise  start= 1  stop=1G  dec=5
sweeptype=relative relharmvec=[1 0]
```

Semi-Autonomous Simulation GUI

You can setup the semi-autonomous simulation in GUI, as shown in the figure below.

Figure 4-1 Semi-autonomous simulation form

To use the semi-autonomous simulation, you need to:

1. Choose 'hb' analysis.

2. Select "Names" option in the Tones section.

3. Select 'Oscillator'.

4. The tone name of the oscillator is 'osc!', fill in the initial frequency guess for this tone in the *Expr* field. Click *Change*.

5. Select Decide automatically from the *Run Transient?* drop-down list. This will cause Spectre to run a transient in the tstab interval until steady-state is reached. Then, a Fourier transform is calculated, and the frequency domain iterations begin.

6. Set the harmonics and the oversample factor as required. If *Run transient?* is set to *Decide automatically* or *Yes*, then *auto* is entered for the number of harmonics in the first tone only. This can be set manually, if desired. The number of harmonics for the rest of the tones must be set manually.

7. Click *Change*.

8. Specify the oscillator nodes in the circuit. This node can be any node in the circuit where the oscillations are present.

Trouble Shooting

1. If the oscillator doesn't oscillate in the semi-autonomous simulation, try to increase the "tstab" time. You can use the parameter "saveinit=yes" to save the tstab waveforms. If the oscillator doesn't oscillate in the tstab transient simulation, give it a kick using the IC method, or using the linear initial condition ("oscic=lin").

2. The semi-autonomous simulation may have convergence problem for unlock driven oscillators when the perturbation frequency is very low and perturbation amplitude is large. Decreasing the perturbation amplitude will improve the convergence. Increasing the harmonic number of the RF tone also improves the convergence.

3. All trouble shooting tricks for the autonomous PSS simulation are also applicable. Please refer to the autonomous PSS application notes.