Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Agilent Technologies
395 Page Mill Road
Palo Alto, CA 94304 U.S.A.

Copyright © 2001, Agilent Technologies. All Rights Reserved.
## Contents

### 1 Simulation Basics

Simulation Methods ................................................................. 1-1  
About Licensing .................................................................... 1-2  
Common Simulation Usage ..................................................... 1-3  
Working with the Examples Directory ................................. 1-4  
The Simulation Process ......................................................... 1-5  
Selecting Simulation Components ......................................... 1-6  
Using the Options Component .............................................. 1-7  
   Reusing DC Simulation Solutions ......................................... 1-7  
Sweeping Parameters ......................................................... 1-8  
Optimizing a Design ............................................................ 1-9  
Working with Expressions .................................................. 1-9  
   Expressions Examples ....................................................... 1-10  
Saving Simulation Data ....................................................... 1-10  
   Selectively Saving and Controlling Simulation Data ........... 1-10  
Controlling a Simulation ...................................................... 1-13  
Displaying Simulation Results ............................................. 1-14  
Viewing DC Solutions .......................................................... 1-15  
   Viewing Device Operating Point Data ................................. 1-15  
Reusing Simulation Solutions .............................................. 1-16  
Simulating from a Layout .................................................... 1-17  

### 2 Preparing a Circuit for Simulation

Using Current Probes ............................................................. 2-2  
Naming Nodes ....................................................................... 2-3  
Using Node Sets ................................................................. 2-4  
Highlighting Nodes ............................................................... 2-4  
Using Constants, Variables, and Functions ............................. 2-5  
   Pre-defined Constants ...................................................... 2-5  
   Pre-defined Variables .................................................... 2-6  
   Pre-defined Functions ..................................................... 2-6  
   The Effect of Expressions on Units ................................. 2-8  
Applying Measurements ...................................................... 2-9  
   Quantities Measurements Can Reference .......................... 2-10  
   Simulation Output Names ................................................. 2-10  
Using Simulation Templates ................................................. 2-12  
Using Simulation Instrument Components .......................... 2-19
6  S-parameter Simulation
Performing an S-parameter Simulation ................................................................. 6-2
Examples ............................................................................................................... 6-3
Simulating an Amplifier ....................................................................................... 6-3
Calculating Group Delay ..................................................................................... 6-6
Simulating Linear Noise ...................................................................................... 6-7
Analyzing a Frequency Translating Network ....................................................... 6-8
Eliminating Unwanted Effects ............................................................................. 6-9
S-parameter Simulation Overview .......................................................................... 6-10
S-parameter Definitions ...................................................................................... 6-10
Group Delay ....................................................................................................... 6-12
S-parameter Frequency Conversion .................................................................... 6-13
Noise Analysis ..................................................................................................... 6-14
Noise Figure ....................................................................................................... 6-14
Calculating Noise Figure .................................................................................... 6-15
Noisy 2-port Parameters ..................................................................................... 6-16

7  Harmonic Balance Simulation
Performing a Harmonic Balance Simulation .......................................................... 7-2
Examples ............................................................................................................... 7-3
Single Tone Harmonic Balance Simulation ........................................................ 7-4
Swept Harmonic Balance Simulation .................................................................. 7-6
Harmonic Balance Overview ................................................................................ 7-9
The Simulation Process ....................................................................................... 7-9
Comparing Harmonic Balance and Time Domain Simulators ......................... 7-10
Harmonics and Maximum Order ......................................................................... 7-10
Selecting a Solver .............................................................................................. 7-15
Reusing Simulation Solutions ............................................................................. 7-17
Troubleshooting a Simulation ............................................................................. 7-18
Selecting the Number of Harmonics .................................................................. 7-19
Reducing Simulation Time ................................................................................. 7-20
Solving Convergence Problems ......................................................................... 7-21
Oversampling to Prevent Aliasing ...................................................................... 7-23
Linearizing Nonlinear Devices .......................................................................... 7-26

8  Harmonic Balance for Nonlinear Noise Simulation
Performing a Nonlinear Noise Simulation ............................................................ 8-2
Performing a Noise Simulation with NoiseCons ................................................ 8-4
Nonlinear Noise Overview .................................................................................. 8-5
NoiseCon Overview ............................................................................................ 8-6
9 Harmonic Balance for Oscillator Simulation
Performing an Oscillator Simulation ................................................................. 9-2
Examples............................................................................................................ 9-3
Finding the Frequency of Oscillation ............................................................... 9-4
Calculating Large-Signal, Steady-State Oscillation Conditions ....................... 9-6
Calculating Oscillator Loop Gain .................................................................... 9-11
Using OscPort2 for Oscillator Analysis ......................................................... 9-14
Oscillator Simulation Overview ......................................................................... 9-21
Performing VCO Tuning .................................................................................. 9-22
Measuring Oscillator Loop Gain ..................................................................... 9-22
Troubleshooting a Simulation .......................................................................... 9-25
Solving Convergence and Speed Problems ..................................................... 9-25
Correcting for Search Failures ......................................................................... 9-26
When You Question the Accuracy of Frequency Results ............................... 9-26

10 Oscillator Noise Simulation
Simulating Phase Noise Using OscPort ............................................................. 10-2
Simulating Phase Noise with NoiseCons ......................................................... 10-4
Single Sideband Phase Noise ........................................................................ 10-5
Relative Noise Voltage Spectrum ................................................................. 10-7
Phase Noise Voltage Around a Harmonic ....................................................... 10-9
Absolute Noise Voltage Spectrum ................................................................. 10-11
Phase Noise Overview ...................................................................................... 10-13
How ADS Simulates Phase Noise .................................................................... 10-14
Possible Problems with Phase Noise Analysis ................................................. 10-15
Basic Phase Noise Theory ................................................................................ 10-17

11 Harmonic Balance for Mixers
Performing a Basic Mixer Simulation ............................................................... 11-2
Examples.......................................................................................................... 11-3
Finding Mixer Output Tones ............................................................................ 11-3
Performing a Small-Signal Simulation of a Mixer .......................................... 11-7
Determining Mixer Conversion Gain .............................................................. 11-10
Determining Mixer Intermodulation Distortion ................................................. 11-14
Small-Signal Overview .................................................................................... 11-17

12 Mixer Noise Simulation
Determining Mixer Noise .................................................................................. 12-2
Simulating Mixer Noise with NoiseCons ......................................................... 12-5
Mixer Noise Figure ......................................................................................... 12-6
Noise at Different Nodes and Frequencies ....................................................... 12-7
Effects of LO Phase Noise on Noise Figure ..................................................... 12-8
Small-Signal Noise Simulation ......................................................................... 12-12
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transient/Convolution Simulation Process</td>
<td>17-8</td>
</tr>
<tr>
<td>Integration Methods Used in Transient/Convolution Simulation</td>
<td>17-10</td>
</tr>
<tr>
<td>Reusing Transient Simulation Solutions</td>
<td>17-11</td>
</tr>
<tr>
<td>Monte-Carlo Noise in the Time Domain</td>
<td>17-12</td>
</tr>
<tr>
<td>Troubleshooting a Simulation</td>
<td>17-12</td>
</tr>
<tr>
<td>Avoiding Simulation Errors in Transient Analysis</td>
<td>17-12</td>
</tr>
<tr>
<td>Solving Convergence Problems</td>
<td>17-13</td>
</tr>
<tr>
<td>Comparing Time-Domain and Steady-State Results</td>
<td>17-14</td>
</tr>
<tr>
<td>Setting Max Frequency and Other Convolution Parameters</td>
<td>17-14</td>
</tr>
<tr>
<td>Dealing with Noncausal Frequency Responses</td>
<td>17-16</td>
</tr>
<tr>
<td>Using Measured and Simulated S-Parameter Data</td>
<td>17-17</td>
</tr>
</tbody>
</table>

18 Using Circuit Simulators for RF System Analysis

- Applicable Simulation Components                                      | 18-2 |
- Applicable Measurements                                                | 18-4 |
- Fundamentals of Using Circuit Simulation                               | 18-5 |
  - Components for System Analysis                                       | 18-5 |
  - Example                                                              | 18-6 |
- Budget Analysis                                                        | 18-8 |
  - Using the Perform Budget Simulation Option                           | 18-9 |
  - Using Budget Measurement Components                                 | 18-9 |
  - Adding a Budget Path                                                 | 18-9 |
  - Using Budget Measurement Functions                                  | 18-10|
  - Using the BudLinearization Component                                 | 18-10|
- Budget Examples                                                        | 18-11|
  - Using MixerIMT/MixerIMT2 Models in Spurious Signal Analysis          | 18-26|
- System Noise Analysis                                                  | 18-30|

19 Parameter Sweeps and Sweep Plans

- Conducting Sweeps                                                      | 19-2 |
- Basic Procedures                                                       | 19-4 |
  - Using ParamSweep to Sweep Two Parameters                             | 19-4 |
  - Using SweepPlans to Perform Fine and Coarse Sweeps                   | 19-8 |
  - Additional Examples                                                  | 19-13|
- Recommendations and Tips                                               | 19-14|
  - Ensuring that Sweep Results are Displayed Correctly                  | 19-14|
  - Controlling the Amount of Data Sent to the Dataset                   | 19-15|
  - Using Sweeps in Monte Carlo Analysis                                 | 19-15|
## 20 Working with Data Files

<table>
<thead>
<tr>
<th>Types of Data Files</th>
<th>20-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Making a Data File</td>
<td>20-4</td>
</tr>
<tr>
<td>Saving a Data File</td>
<td>20-4</td>
</tr>
<tr>
<td>Using Data Files, Datasets, and Data Access Components</td>
<td>20-5</td>
</tr>
<tr>
<td>Examples</td>
<td>20-5</td>
</tr>
<tr>
<td>SnP Format</td>
<td>20-6</td>
</tr>
<tr>
<td>Linking an .snp File to an SnP Component</td>
<td>20-6</td>
</tr>
<tr>
<td>Overview</td>
<td>20-7</td>
</tr>
<tr>
<td>Basic SnP File Format</td>
<td>20-8</td>
</tr>
<tr>
<td>Adding Comments to Data Files</td>
<td>20-11</td>
</tr>
<tr>
<td>Adding Noise Parameters to an SnP File</td>
<td>20-12</td>
</tr>
<tr>
<td>Applying the SnP Format, and Examples</td>
<td>20-13</td>
</tr>
<tr>
<td>Y- and Z-parameter Files</td>
<td>20-21</td>
</tr>
<tr>
<td>Y- (Z-) Parameter 5- to 99-port File Formats</td>
<td>20-24</td>
</tr>
<tr>
<td>S2D Format</td>
<td>20-26</td>
</tr>
<tr>
<td>Guidelines for .s2d</td>
<td>20-27</td>
</tr>
<tr>
<td>Complete .s2d File Example</td>
<td>20-42</td>
</tr>
<tr>
<td>P2D Format</td>
<td>20-46</td>
</tr>
<tr>
<td>Guidelines for .p2d</td>
<td>20-47</td>
</tr>
<tr>
<td>The ACDATA Block</td>
<td>20-49</td>
</tr>
<tr>
<td>The NDATA Block</td>
<td>20-56</td>
</tr>
<tr>
<td>The IMTDATA Block</td>
<td>20-58</td>
</tr>
<tr>
<td>Example .p2d File</td>
<td>20-61</td>
</tr>
<tr>
<td>Example .p2d File with Annotation</td>
<td>20-64</td>
</tr>
<tr>
<td>IMT Format</td>
<td>20-68</td>
</tr>
<tr>
<td>Example</td>
<td>20-68</td>
</tr>
<tr>
<td>TIM, BINTIM Formats</td>
<td>20-70</td>
</tr>
<tr>
<td>Guidelines for .tim</td>
<td>20-70</td>
</tr>
<tr>
<td>DSCR Format</td>
<td>20-74</td>
</tr>
<tr>
<td>Selecting a Row</td>
<td>20-74</td>
</tr>
<tr>
<td>Using the File with a DAC</td>
<td>20-75</td>
</tr>
<tr>
<td>Example</td>
<td>20-75</td>
</tr>
<tr>
<td>Generalized MDIF</td>
<td>20-76</td>
</tr>
<tr>
<td>PDF Format</td>
<td>20-79</td>
</tr>
<tr>
<td>Guidelines for .pdf</td>
<td>20-79</td>
</tr>
<tr>
<td>Interpretation of PDF data</td>
<td>20-80</td>
</tr>
<tr>
<td>ASCSIG and SIG Formats</td>
<td>20-81</td>
</tr>
<tr>
<td>Example .ascsig Files</td>
<td>20-81</td>
</tr>
<tr>
<td>CITfile Data Format</td>
<td>20-83</td>
</tr>
<tr>
<td>Overview</td>
<td>20-83</td>
</tr>
<tr>
<td>Data Formats</td>
<td>20-84</td>
</tr>
</tbody>
</table>
Chapter 1: Simulation Basics

This chapter provides general information that applies to all types of simulations. Before using this manual, you should be familiar with the information in the Quick Tour (online) and the User’s Guide. This manual contains information on how to use the circuit simulators that are available in Advanced Design System.

You should also refer to Chapter 2, Preparing a Circuit for Simulation and Chapter 3, A Simulation Example, then continue with the chapters in this manual that describe the different types of simulators.

Simulation Methods

Advanced Design System provides simulators that enable you to simulate circuits and RF systems designed for specific objectives. Table 1-1 provides brief descriptions of the available simulators; subsequent chapters describe each type of simulation in detail.

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>Fundamental to all simulations, it performs a topology check and an analysis of the DC operating point of a circuit.</td>
</tr>
<tr>
<td>AC</td>
<td>Obtains small-signal transfer parameters, such as voltage gain, current gain, and linear noise voltage and currents. This simulator is useful in designing passive circuits and small-signal active circuits such as low-noise amplifiers (LNAs).</td>
</tr>
<tr>
<td>S-parameter</td>
<td>Provides linear S-parameters, linear noise parameters, transimpedance ($Z_{ij}$), and transadmittance ($Y_{ij}$), by linearizing the circuit about the DC operating point and performing a linear small-signal analysis that treats the circuit as a multiport. Each port is turned on sequentially. S-parameters can be converted to Y- and Z-parameters. This simulator can be used to achieve many of the same design goals as the AC simulator.</td>
</tr>
<tr>
<td>Harmonic Balance</td>
<td>Uses nonlinear harmonic-balance techniques to find the steady-state solution in the frequency domain. This simulator is useful in designing RF amplifiers, mixers, and oscillators. A Krylov subspace technique is available to reduce memory requirements and increase the speed of solution. This option is useful in designing large RF integrated circuits or RF/IF subsystems, where a large number of devices or large numbers of harmonics and intermodulation products are involved.</td>
</tr>
</tbody>
</table>
Simulation Basics

Table 1-1. Simulator Descriptions

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large-signal S-parameter (LSSP)</td>
<td>A type of harmonic balance simulation, it performs large-signal S-parameter analyses to represent the nonlinear behavior of items such as power amplifiers.</td>
</tr>
<tr>
<td>P2D</td>
<td>Generates a .p2d file that can be used to describe the behavior of a file-based component (such as the AmplifierP2D component, available in the System-Amps &amp; Mixers library).</td>
</tr>
<tr>
<td>Gain Compression (X dB)</td>
<td>Seeks a user-defined gain-compression point at which an actual power curve deviates from an idealized linear power curve. This is useful in power amplifier design.</td>
</tr>
<tr>
<td>Circuit Envelope</td>
<td>Uses a combination of frequency- and time-domain analysis techniques to yield a fast and complete analysis of complex signals such as digitally modulated RF signals. It represents input waveforms as RF carriers with modulation envelopes that are described in the time domain. This is useful in designing circuits and systems involving modulators/demodulators or complex modulated signals.</td>
</tr>
<tr>
<td>Transient/Convolution</td>
<td>Solves a nonlinear circuit in the time domain, and linear components can be simulated by means of convolution or a simplified equivalent-circuit model.</td>
</tr>
</tbody>
</table>

About Licensing

These simulators require licenses to run a simulation. Confirm that the simulator of interest is included with your purchase. Advanced Design System will allow you to create a circuit, but if you do not have the correct license you will not be able to simulate it.
Common Simulation Usage

Table 1-2 describes some common design objectives and the simulators that would be appropriate to each, in the order in which they would generally be applied.

<table>
<thead>
<tr>
<th>Design</th>
<th>Simulator</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>S-parameter</td>
<td></td>
</tr>
<tr>
<td>Mixer</td>
<td>DC</td>
<td>Test for AC frequency conversion (also known as frequency-converting AC, or FCAC). Applies to system mixer models only.</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>Select nonlinear noise option to obtain noise figure.</td>
</tr>
<tr>
<td></td>
<td>Harmonic Balance</td>
<td>Find ACPR (adjacent-channel power ratio).</td>
</tr>
<tr>
<td>Transient</td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>Power amplifier</td>
<td>DC</td>
<td>Test for load-pull characteristics.</td>
</tr>
<tr>
<td></td>
<td>S-parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Harmonic Balance</td>
<td>Also use the P2D simulator to generate a .p2d file.</td>
</tr>
<tr>
<td></td>
<td>LSSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XDB</td>
<td>Find gain-compression point.</td>
</tr>
<tr>
<td>Transient</td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>Transceiver</td>
<td>AC</td>
<td>Test for AC frequency conversion (FCAC).</td>
</tr>
<tr>
<td></td>
<td>Harmonic Balance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>Oscillator</td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S-parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Harmonic Balance</td>
<td>Check for power spectra and phase noise.</td>
</tr>
<tr>
<td></td>
<td>Envelope</td>
<td>Check for startup switching.</td>
</tr>
<tr>
<td>Phase-locked loop</td>
<td>Envelope</td>
<td>Check for transient responses.</td>
</tr>
</tbody>
</table>
Working with the Examples Directory

Most of the designs referred to in this manual are in the Examples directory. The process for locating and opening example projects is described in the User's Guide.

There are example projects and templates that are supplied with the Advanced Design System. On UNIX, these projects are read-only directories. To work with an example project, you must first make a copy in a directory for which you have write permission. PC users should also copy these examples to preserve the integrity of the examples. For convenience in keeping track of designs, you may want to create directory names that mirror those in the examples directory.

Do not copy projects by using your operating system alone. Use these methods:

- Use the Copy Project command in the Main window to copy entire projects to your local directory.
- Copy projects as part of the software installation procedure.

This ensures that all files that are part of the project are copied.
The Simulation Process

The basic simulation process is:

- Create your schematic, then add current probes and identify the nodes from
  which you want to collect data. Details about common items that should be
  added to a schematic prior to simulation are described in Chapter 2, Preparing
  a Circuit for Simulation.

- Select a simulation method, specifying parameters as necessary. The
  parameters you specify are based on the type of simulation you choose.

- Select a name for the dataset. This is where the simulation data will be saved.

- Run the simulation.

- View DC data by annotating the schematic with DC solutions and by viewing
  brief or detailed device operating point data.

- Display additional results using the Data Display.

- Optimize and tune a design. For details, refer to the Tuning, Optimization and
  Statistical Design manual.

These are the basic steps. It is also possible to develop very detailed, complex
simulations, but the process, for the most part, remains the same. The remainder of
this chapter gives an overview of these steps.
Simulation Basics

Selecting Simulation Components

Simulation components are grouped in a number of simulation palettes accessed from the Component Palette List.

Each palette contains the specific simulation component, plus:
- The Options component
- Components for defining sweep plans and parameter sweeps
- Node set components
- Measurements
- Frequently-used components, such as ports and sources

To use a component, select it from the palette, position the pointer in the drawing areas of the Schematic window and click to place it.
Using the Options Component

The Options component includes general simulation options such as convergence tolerances, warnings, and global noise temperature. An options component can be used with any simulation; it is available from every simulation palette.

The Options component is commonly used in nonlinear noise analyses. The IEEE standard temperature ($T_0$) for noise figure measurement is 290 K (16.85 degrees Celsius). This is set via the Options component.

For details about the Options parameters, place an Options component on your schematic and double-click it. Click Help on the dialog box that appears.

Reusing DC Simulation Solutions

You can save the complete DC solution to a file and then reuse it as an initial guess in later simulations. For large circuits or ones with time-consuming DC simulations this can save a significant amount of CPU time by avoiding repeating of the same, or a very similar, simulation each time. This applies to any simulation that either performs or relies on a DC solution, which includes all simulations with nonlinear elements.

For example, once a DC solution is obtained by running an AC simulation, then future AC simulations at different frequencies or linear noise simulations do not have to re-simulate to get the same DC solution again. If the circuit is changed, either via a parameter change or even a topology change that will change the DC solution, then this saved DC solution can still be used an initial guess for the new DC solution. If the circuit change was not too significant, then having a reasonable initial guess will still usually reduce the total re-simulation time. If the circuit change is significant enough so that the simulation cannot converge using the supplied initial guess, then the simulator will proceed with its normal DC simulation algorithm. In this case, it would save CPU time to disable the Use Initial Guess.

To save a simulation for reuse:

1. From any simulation palette, select the Options component. Place it on the schematic and double-click to edit it.
2. Select the DC Solutions tab.
3. Enable Write Final Solution. Enter a filename and any extension, or use the default which is <$\text{project\_name}>.dcs. The file will be saved in the networks folder of the project.
4. Click **OK**.

To select a solution file to be used as the initial guess:

1. Place an Options component on the schematic if one is not present, then double-click to edit it.
2. Select the **DC Solutions** tab.
3. Select the option **Use Initial Guess**. Enter the filename and extension.
4. Optionally, to view any messages regarding how the initial guess affects the simulation, select the Annotate option.

If the circuit topology has changed between the time the solution file was created and when it is used as an initial guess, the simulator will still attempt to use as much of the data as possible. It will also output various messages, if desired, noting what has changed between the two versions. While this feature does not check for parameter changes, it can be a useful tool for comparing the topology of two circuits, or to identify what has changed since the solution was last saved. Items checked include the total number of equations (nodes and branches), the total number of instances and their names, and most connectivity changes.

**Sweeping Parameters**

Most simulations are performed over a range of values instead of just a single point. You can sweep over time or frequency (depending upon the type of simulation) or you can elect to sweep over another parameter. You can sweep one or more parameters using the following methods:

- You can set the sweep range of time, frequency, or a single other parameter (under the Sweep tab for a given simulator component.)
- You can use the ParamSweep and SweepPlan components for sweeping more than one parameter, or sweeping over more than one range of values. These components appear on all simulation palettes. For details on how to use these components, refer to Chapter 19, Parameter Sweeps and Sweep Plans.

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate >Simulation Setup). Individual sweep points are run on each machine and the results are combined into a single dataset on the local machine. For details on setting up remote and local machines for remote processing, refer to these appendices:
• Appendix E, Using Remote Simulation in the Installation on PC Systems manual

**Optimizing a Design**

You can set up nominal optimizations or statistical yields as part of a simulation. These features require a separate license. For the details of optimization, refer to Tuning, Optimization and Statistical Design.

**Working with Expressions**

You can add variables, functions, and conditional statements to a schematic, making your designs more flexible and versatile.

You can use these items:

• In component parameter definitions. From the component parameter editing dialog box, select the parameter and, if available, click Equation Editor. You can write an expression that defines this parameter.

• In variables. Variables can be added to a schematic using the VAR (VarEqn) component, which can be found in the Data Items palette. Once a variable is defined, it can be used in expressions within the design.

For more information on how to use variables and functions, refer to “Using Constants, Variables, and Functions” on page 2-5.

You can also add measurements to a schematic. Measurements are predefined functions that process data so that it can be presented in the Data Display. There are numerous predefined measurements under the simulation palettes, but you can also create your own using the MeasEqn component.

For more information on how to use measurements, refer to Expressions, Measurements, and Simulation Data Processing.
Expressions Examples

Many of the projects in the Examples directory use variables and measurements. One example that includes many variable definitions plus conditional statements is NADC_PA_Test.dsn in RF_Board/ NADC_PA_prj.

Saving Simulation Data

The results of the simulation are stored in a dataset. This dataset is then used by the Data Display for viewing results. You can select the name and location of the dataset you want to use for a simulation. The default name of the dataset is the name of the design.

To specify a dataset name:

From the Schematic window, choose Simulate > Simulation Setup. Enter the name of the dataset where you want simulation data to be saved. Click Apply or, if you are ready to run the simulation, click Simulate.

Note Running a simulation a second time will overwrite the contents of a dataset. Use a different dataset name if you want to save data from the previous simulation.

Selectively Saving and Controlling Simulation Data

The Output tab in all A/RF analysis components can be used to control the data that will be saved to the dataset. You can control output generated from named nodes, buses, measurement and VAR equation. By default, all named nodes up to two levels below the top level are saved. The data from all measurement equations and components with the parameter SaveCurrent set to yes is also saved. You can also control the saving of data using the hierarchy level for nodes and measurement equations or by explicitly specifying the output quantity.

To modify the default behavior of sending data to the dataset:

1. Edit the analysis controller item to select the Output tab.
• To output all named nodes/measurement equations in the top-level design only, select the Node Voltages and/or Measurement Equations options and use zero as the Maximum Depth.

• To output all named nodes/measurement equations in the top-level design and one or more levels in the hierarchy, select the Node Voltages and/or Measurement Equations options and set the desired Maximum Depth.
Note  When you use the Save by hierarchy options, all data from the specified levels is output—you cannot restrict it. However, you can add to it, selectively, from lower levels in the hierarchy using the Save by name section.

- To output named nodes/measurement equations selectively, irrespective of the hierarchy, disable the Node Voltages and/or Measurement Equations options in the Save by hierarchy section and use the Save by name section (and secondary Add/Remove dialog box) to select only those nodes/equations you want to output.

2. If using the Save by hierarchy method, select the desired level of hierarchy and click OK.

3. If using the Save by name method (alone or in conjunction with a specified level of hierarchy), click Add/Remove.
Controlling a Simulation

You can select how to start and end a simulation:

- To start a simulation, choose Simulate from the Simulate menu. You can also start one by clicking the Simulate button on the tool bar or by pressing F7.

- To end a simulation before it is finished, choose Stop and Release Simulator from the Simulate menu. This will release your simulation license. To end the simulation but keep the license, choose Simulation/ Synthesis > Stop Simulation from the Simulation Message window.

You can add more than one simulation component to a schematic, and specify which simulation to run (only one simulator can be active at a time):

Note: Buses appear in the list box as nodes. They do not contain any indices (as they do in the design environment). Individual components of a bus cannot be output.

4. Select each individual node you want to send to the dataset, or equation you want saved for each iteration (time, frequency) for the next simulation, click Add to move each to the Current Selection list box, and click OK.

The selected nodes/equations are displayed in the Save by name list box.

Note that a non-selected equation may still be output depending on the Maximum Depth setting, but all data will be processed at the end of all simulation iterations, requiring more memory.

If an equation is selected for any one simulation, it will not be output for any other simulation for which it is not explicitly selected.

5. Make any other desired changes for this analysis and click OK.
Simulation Basics

To disable simulations that are not desired, choose Edit > Component > Deactivate and click the appropriate simulation component.

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and the results are combined into a single dataset on the local machine. You can also use this utility to select the fastest available machine.

For details on setting up remote and local machines for remote processing, refer to these appendices:

• Appendix E, Using Remote Simulation in the Installation on PC Systems manual

Displaying Simulation Results

Most of the simulation results are viewed in the Data Display. You can set the Data Display so that it automatically opens when a simulation is finished:

1. Choose Simulate > Simulation Setup.
2. Select the option Open Data Display when simulation completes.
3. Specify a data display file (.dds) and it will be opened in the Data Display window when the simulation is finished.

The simulation templates include DisplayTemplate components. If your design includes a simulation template and the automatic display option is enabled, then the Data Display window will open with the corresponding display template.

If neither a data display file (.dds) is specified nor a template used, a blank Data Display window is opened.

Subsequent simulations of the same schematic will not open a new window, but rather bring the existing one to the foreground.

For information on how to work with the items in a Data Display window, refer to the Data Display manual.
There are also ways to view DC data directly from the schematic. You can view:

- DC node solutions
- DC operating point data

**Viewing DC Solutions**

After a simulation is finished, you can display DC node voltages and branch/pin currents on the schematic. Because a DC simulation is part of most other types of simulations as well, this feature is available for most simulations.

- To view DC solutions, from the Schematic window, choose Simulate > Annotate DC Solution. All node voltages and branch/pin currents of the last DC solution—which was obtained from either the last explicit or implicit DC analysis—are displayed on the schematic.
- To erase the solutions from the schematic, choose Simulate > Clear DC Annotation.

**Viewing Device Operating Point Data**

Any simulation that includes a DC analysis produces DC operating point information for most active and some passive devices in the circuit. This data includes currents, power, voltages, and linearized device parameters of the selected device. An explanation of the displayed parameters, if available, is under the model documentation in the Circuit Component manual.

To view device operating point data:

- From the Schematic window, choose Simulate > Detailed Device Operating Point. Crosshairs appear. Click the component of interest. The details appear in a separate window.
- To view a condensed list of details, choose Simulate > Brief Device Operating Point instead.
- You can save device operating point data to a dataset for viewing in the Data Display. Under the Parameters tab of most simulators, set the Device operating point level as desired.
Reusing Simulation Solutions

For some types of simulations, you can save a simulation solution and reuse it as an initial guess in a later simulation. This can save time by avoiding repeating the same, or very similar simulation, on a design. Using a simulation solution as an initial guess can help the subsequent simulation to reach a final answer faster. If you have made minor changes to the design or simulation setup, reusing solutions can reduce CPU time.

You can save and reuse these types of simulation data:

- **DC solutions**—You can save the complete DC solution and reuse it for any type of simulation that either performs or relies on a DC solution. For example, once a DC solution is obtained by running an AC simulation, then future AC simulations at different frequencies or linear noise simulations do not have to resimulate to get the same DC solution again. This feature is available from the DC Solutions tab of the Options component. For details, refer to “Using the Options Component” on page 1-7.

- **Harmonic Balance solutions**—You can save a harmonic balance solution and use it as an initial guess for another harmonic balance simulation, large-signal S-parameter, gain compression, or Circuit Envelope simulation. With the saved harmonic balance solution, you can later perform a nonlinear noise simulation and use the saved solution as the initial guess, removing the time required to recompute the nonlinear harmonic balance simulation. Another use would be to use the initial harmonic balance solution, then sweep a parameter to see the changes. For details, refer to “Reusing Simulation Solutions” on page 7-17.

- **Harmonic Balance guess from a Transient simulation**—Transient simulations can be set to generate a harmonic balance solution that can then be used as an initial guess for a harmonic balance simulation. For example, in circuits such as dividers, harmonic balance usually cannot directly converge on a solution since multiple mathematical, but useless, solutions exist. By first running a transient simulation and generating a harmonic balance solution file that can then be used as an initial guess, the harmonic balance simulation can converge to the desired solution. This feature is available from the Freq tab of the Transient component. For details, refer to “Reusing Transient Simulation Solutions” on page 17-11.
Simulating from a Layout

Simulating a layout cannot be done directly from the Layout window; it involves a few steps in the Schematic window. Essentially, you must treat the design as though it were a subnetwork and place it in a higher-level design. To do this, you must create a symbol for it (in the Schematic window).

To simulate a layout:

1. From the Layout window containing the design you want to simulate (in this example, DesignB), choose Window > Schematic.

2. In the Schematic window for DesignB, create a symbol (View > Create/ Edit Schematic Symbol). The number of pins must equal the number of ports on the design.

3. Switch back to the Schematic view (View > Create/ Edit Schematic) and choose File > Design Parameters and select the option labeled Simulate from Layout (SimLay).

4. Save the design.

5. From the Schematic window, create a new design (in this example, DesignA).

6. In the Schematic window for DesignA, open the Component Library and select and place an instance of DesignB.

7. Place the desired simulation control items, as well as any substrate or equation definitions, in DesignA and run the simulation from DesignA.
Chapter 2: Preparing a Circuit for Simulation

This chapter describes a variety of items that can be added to a schematic to prepare it for circuit simulation.

The process for creating a schematic—selecting and placing components, editing component parameters, and wiring—is described in the User’s Guide and in the online Quick Tour. You should be familiar with this process and with working in projects before continuing here.

Refer to the following topics for details on using simulation-specific items:

- “Using Current Probes” on page 2-2 describes how to specify the points in a circuit where you can measure and save current values.
- “Naming Nodes” on page 2-3 describes how to specify the circuit nodes where you can measure and save voltages.
- “Using Node Sets” on page 2-4 describes how to apply best guess voltage and resistance values at points in a circuit to set starting DC values.
- “Highlighting Nodes” on page 2-4 describes how to highlight nodes to quickly locate a point in a circuit.
- “Using Constants, Variables, and Functions” on page 2-5 shows how to use variables and equations to assign values to parameters.
- “Applying Measurements” on page 2-9 shows how to use pre-defined measurements in a schematic, which are evaluated during a simulation and whose results are saved to view in the Data Display.
- “Using Simulation Templates” on page 2-12 shows how to use predefined circuit and simulation setups to simplify creating your design.
- “Using Simulation Instrument Components” on page 2-19 shows how to simplify the simulation process by connecting your design to components that represent instruments and run a simulation.
Using Current Probes

Current probes are added to a schematic to collect current data at that point in the circuit. You can place as many probes as you want in a schematic. Current probes are found on the Probe Components palette.

Current probes have parameters that you may want to edit, but it is not necessary. You may want to rename the probe to something meaningful, since the name is used to name the data collected with the probe.

---

**Note**  
Current probes (I_Probe components) must be placed so that the arrow on the probe points in the direction of (positive) current flow. To flip a probe horizontally, choose **Edit > Rotate/Mirror > Mirror About Y**.
Naming Nodes

To collect voltage data at nodes of interest, you label the nodes on the schematic.

There are two types of nodes: global nodes and named nodes.

By placing a GlobalNode component on an upper-level schematic (Insert > Global Node) of a design, you can select and edit the name of a node that will maintain the same identity downward through all subnetworks of that design. This facilitates the interconnection of boards, IC chips, and connectors.

A named node can be applied to any schematic, but it is specific to that schematic only.

To specify a global node:

1. Choose Insert > Global Node and place the component on the schematic. Double-click to display the dialog box for entering a name.
2. Type a name in the Enter global node name field. Click Add. On this design and lower-level designs, it will be considered the same node.
3. To make an existing named node a global node, select it from Node Name List and click Add.

To name a node:

1. Choose Insert > Wire/Pin Label.
2. In the dialog box appears, type the desired name and click the node on the schematic that you want to associate with that name.

   Note In general, voltage and current data is in phasor representation, so the voltage values at named nodes are peak voltage.

3. You can repeat this for other nodes, or click Done to dismiss the dialog box.

   Note By placing an exclamation mark (!) at the end of node name, it becomes a global node for compatibility with Cadence formats. This should be used only if Cadence compatibility is required.
Using Node Sets

By placing a NodeSet component (available in all simulation palettes) at a strategic place in a circuit, you can instruct the DC simulator to begin its analysis at a given best guess voltage. It is also possible to enter values for connection resistance.

Node sets can be used in any analysis, but are especially useful for:

- Circuits that are bi-stable, such as a flip-flop or ring oscillator, to force it to a known high or low state rather than letting the DC solver find the meta-stable state halfway between high and low.
- Circuits that are isolated from DC by blocking capacitors

NodeSets work in a two-stage process. In the first stage, these elements attach the specified voltage source with a series resistor to the specified node(s) to force a value. A DC solution for the entire circuit is then calculated. In the second stage, the forcing source and resistor are removed and the DC solution is refined, using the previous DC solution as an initial guess.

If you choose to use a NodeSetByName component, you can specify a name to facilitate the retrieval of voltage data in the dataset.

Highlighting Nodes

Highlighting nodes can help you identify specific points in a schematic or subnetwork. To do this, choose Simulate > Highlight Node. This opens a window that lists all nodes (such as named connections, wires, pins, and ports) in a circuit and in all of its subcircuits. Click a node in the list and it will be highlighted on the schematic.

Highlighting nodes can help in troubleshooting a simulation problem. If problems are encountered at a node during simulation, the error and node name will appear in the Simulation/Synthesis Messages window. By using the highlight node feature you can quickly zoom in on the problem area.

Clearing a Highlight

To clear the highlight, select the node name in the Highlight Node window and click Clear.
Using Constants, Variables, and Functions

Advanced Design System contains built-in global constants, variables, and functions that can be used in a schematic. You can use them:

- With the VarEqn component
- With components whose parameters can be defined using equations. (For a selected parameter, the Equation Editor button will appear in the component editing dialog box.)

These can simplify schematic design. For example, you can set a variable named Frequency to a specific value, then use the variable wherever the frequency needs to be specified in the schematic. If you want to change the frequency, you do so in one place.

For more information on how to use VarEqn, refer to the VarEqn component help.

---

**Note**  You can use the conditional statement `if/ then/ else/ endif` in variable definitions and component equations. Be sure to include the `endif`.

---

Many of the projects in the Examples directory use variables. One example that includes many variable definitions plus conditional statements is `NADC_PA_.dsn` in `RF_Board/NADC_PA_prj`.

Lists of constants, variables, and functions are next.

## Pre-defined Constants

The pre-defined built-in constants available for use in an equation are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>e</code></td>
<td><code>2.718 282 ...</code></td>
<td><code>e</code></td>
</tr>
<tr>
<td><code>ln10</code></td>
<td><code>2.302 585 ...</code></td>
<td><code>ln(10)</code></td>
</tr>
<tr>
<td><code>c0</code></td>
<td><code>2.997 924 58 e+08 m/s</code></td>
<td>speed of light</td>
</tr>
<tr>
<td><code>e0</code></td>
<td><code>8.854 188 ... e-12 F/m</code></td>
<td>vacuum permittivity <code>(1/(u0*c0*c0))</code></td>
</tr>
<tr>
<td><code>u0</code></td>
<td><code>1.256 637 ... e-06 H/m</code></td>
<td>vacuum permeability <code>(4*pi*1e-7)</code></td>
</tr>
<tr>
<td><code>boltzmann</code></td>
<td><code>1.380 658 e-23 J/K</code></td>
<td>Boltzmann's constant</td>
</tr>
<tr>
<td><code>qelectron</code></td>
<td><code>1.602 177 33 e-19 C</code></td>
<td>charge of an electron</td>
</tr>
</tbody>
</table>
Preparing a Circuit for Simulation

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>planck</td>
<td>6.6260755e-34 J*s</td>
<td>Planck's constant</td>
</tr>
<tr>
<td>pi</td>
<td>3.141593...</td>
<td>pi</td>
</tr>
</tbody>
</table>

Pre-defined Variables

The pre-defined, built-in variables for use in an equation are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>0 s</td>
<td>analysis time</td>
</tr>
<tr>
<td>timestep</td>
<td>1 s</td>
<td>analysis time step</td>
</tr>
<tr>
<td>freq</td>
<td>1 e+006 Hz</td>
<td>analysis frequency</td>
</tr>
<tr>
<td>temp</td>
<td>25 C</td>
<td>analysis temperature</td>
</tr>
<tr>
<td>_freq1</td>
<td>1 e+006 Hz</td>
<td>fundamental frequencies defined for multi-tone simulations such as Harmonic Balance and Circuit Envelope</td>
</tr>
<tr>
<td>_freq12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pre-defined Functions

Function arguments have the following designations.

- **Complex**: \( x, y \), \( r, r0, r1, rx, ry, s, s1, s2, lower\_bound, upper\_bound \)

In general, the functions return a complex number, unless it is a string operator as noted. A function that returns a real value effectively has a zero value imaginary term.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos(x)</td>
<td>cosine function, ( x ) is in radians</td>
</tr>
<tr>
<td>cot(x)</td>
<td>cotangent function, ( x ) is in radians</td>
</tr>
<tr>
<td>conj(x)</td>
<td>complex-conjugate function</td>
</tr>
<tr>
<td>cosh(x)</td>
<td>hyperbolic cosine function</td>
</tr>
<tr>
<td>coth(x)</td>
<td>hyperbolic cotangent function</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>exp(x)</td>
<td>exponential function</td>
</tr>
<tr>
<td>imag(x)</td>
<td>imaginary-part function</td>
</tr>
<tr>
<td>log(x)</td>
<td>log base 10 function</td>
</tr>
<tr>
<td>ln(x)</td>
<td>natural log function</td>
</tr>
<tr>
<td>mag(x)</td>
<td>magnitude function</td>
</tr>
<tr>
<td>phase(x)</td>
<td>phase (in degrees) function</td>
</tr>
<tr>
<td>phasedeg(x)</td>
<td>phase (in degrees) function</td>
</tr>
<tr>
<td>phaserad(x)</td>
<td>phase (in radians) function</td>
</tr>
<tr>
<td>real(x)</td>
<td>real-part function</td>
</tr>
<tr>
<td>sin(x)</td>
<td>sine function, x is in radians</td>
</tr>
<tr>
<td>sinh(x)</td>
<td>hyperbolic sine function</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>square root function</td>
</tr>
<tr>
<td>tan(x)</td>
<td>tangent function, x is in radians</td>
</tr>
<tr>
<td>tanh(x)</td>
<td>hyperbolic tangent function</td>
</tr>
<tr>
<td>abs(rx)</td>
<td>absolute value function</td>
</tr>
<tr>
<td>arcsinh(rx)</td>
<td>arcsinh function</td>
</tr>
<tr>
<td>arctan(rx)</td>
<td>arctan function, returns radians</td>
</tr>
<tr>
<td>atan2(rx, ry)</td>
<td>arctangent function (two real arguments), returns radians</td>
</tr>
<tr>
<td>complex(rx, ry)</td>
<td>real-to-complex conversion function</td>
</tr>
<tr>
<td>db(rx)</td>
<td>decibel function, 20 log10(x)</td>
</tr>
<tr>
<td>dbpolar(rx, ry)</td>
<td>(dB,angle)-to-rectangular conversion function, rx=mag in dB, ry=angle, degrees</td>
</tr>
<tr>
<td>dbmtoW(rx)</td>
<td>convert dBm to watts</td>
</tr>
<tr>
<td>deg(rx)</td>
<td>radian-to-degree conversion function</td>
</tr>
<tr>
<td>int(rx)</td>
<td>convert-to-integer function</td>
</tr>
<tr>
<td>jn(r0, r1)</td>
<td>bessel function</td>
</tr>
<tr>
<td>max(rx, ry)</td>
<td>maximum function</td>
</tr>
<tr>
<td>min(rx, ry)</td>
<td>minimum function</td>
</tr>
<tr>
<td>polar(rx, ry)</td>
<td>polar-to-rectangular conversion function, rx=magnitude, ry=angle, degrees</td>
</tr>
</tbody>
</table>
Preparing a Circuit for Simulation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rad(rx)</td>
<td>degree-to-radian conversion function</td>
</tr>
<tr>
<td>sgn(rx)</td>
<td>signum function</td>
</tr>
<tr>
<td>sinc(rx)</td>
<td>sin(x)/x function</td>
</tr>
<tr>
<td>sprintf(...)</td>
<td>formatted print utility; returns a string</td>
</tr>
</tbody>
</table>

example:
- \( x = 2 \)
- \( y = 14 \)
- \( z = \text{sprintf}( "%i.%i", x, y) \)
  - results in the string "2.14"
  - \( \text{sprintf} \) follows standard C programming syntax

<table>
<thead>
<tr>
<th>strcat(...)</th>
<th>string concatenation utility; returns a string</th>
</tr>
</thead>
</table>

example:
- \( s1 = "my cat" \)
- \( s2 = " is frisky" \)
- \( s3 = \text{strcat}( s1, s2) \)
  - results in the string "my cat is frisky"

The Effect of Expressions on Units

Due to the way that the simulator processes expressions, the following expression is considered valid by the ADS simulator: \( F = 1.0 \text{ M} \text{ M} \). This value is interpreted by the simulator as: \( F = 1.0 * 1.0e6 * 1.0e6 \). This situation can occur when a variable is defined with units and the variable is then used as a component parameter that also has a units field. Although valid, such an expression usually does not specify the intended value.

The behavior of the Edit Component Parameters dialog is designed so that a parameter value, initially specified as a number followed by a scale factor, is changed to a non-numeric value, and the scale factor setting is automatically set to None. This scale factor setting can be changed manually, if desired.
Applying Measurements

Measurements are pre-defined expressions that make it easy to make common calculations such as VSWR or signal-to-noise ratio. Measurements are available from the simulation palettes and have two purposes:

- They can be used on the schematic, in conjunction with simulations, to process the results of a simulation.
- They can be used in Data Display equations to process the results of a simulation and display various relationships graphically.

To create your own measurement, use the MeasEqn component. For details about measurements, refer to the manual Expressions, Measurements, and Simulation Data Processing.

To add a measurement to a schematic:

Select a measurement from the simulation palette and place it on the schematic.

- You can modify the measurement to customize it or change the name. Click the Help button in the dialog box for details about the measurement.
- You can select the measurement for output for a specific analysis. This has the effect of restricting evaluation to that analysis only (if more exist), as well as saving the result after each analysis iteration (e.g. each time point, or frequency point), instead of after all iterations, thus using less memory for intermediary data.

To view the results after running the simulation:

1. Open a Data Display window and select a plot and place it in the window.
2. The name of the measurement will appear in the list of variables. Select it to add it to the plot and click OK.

Measurements can also be used in Data Display equations to perform additional processing after a simulation. For information on how to use measurements in Data Display equations, refer to the Data Display manual.
Quantities Measurements Can Reference

Measurements can reference:

- Any simulation outputs (voltages, currents, S-parameters) from the current circuit level and levels below using full hierarchical names (refer to “Simulation Output Names” on page 2-10).

- Other measurements and variable equations. Measurement equations and variable equations follow the same nested scoping rules: measurement equations can reference other measurement and variable equations at the current or higher levels. Note that measurement and variable equations cannot share the same name (refer to Naming Conventions in the User’s Guide).

- Existing data in datasets produced by previous simulations or imported via the instrument server. The full circuit path of the saved simulation output is always required. Using the same syntax used for data displays, To reference an existing dataset entry, preface the measurement name with the dataset name.

Example.

<table>
<thead>
<tr>
<th>MeasEqn1 = Vout</th>
<th>accesses node Vout in the current circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeasEqn2 = saved_dataset.DC1.DC.Vout</td>
<td>accesses node Vout, generated by analysis DC1 in the dataset saved_dataset.ds</td>
</tr>
</tbody>
</table>

Simulation Output Names

To successfully use measurement equations, you must understand the full names associated with simulation outputs. Each simulation output has a unique name. A measurement may refer to such an output by using its unique name, or a condensed version of it. The full unique name of a simulation output is described in the following illustration.

```
analysis_path.circuit_path.name
    └── name of the simulation output
        └── path of the simulation output with respect to the circuit level of the measurement that references it
            └── a concatenated string of the full circuit names of all the simulations driving the analysis
```
where:

- **analysis_path** is a concatenated string of the full circuit names of all the simulations driving the analysis. For example, a single top-level DC analysis called DC1 on the design results in the analysis path DC1.DC. If a top-level sweep analysis called Sweep1 drives that DC analysis, then the circuit path is Sweep1.DC1.DC. The .DC suffix is specific to the DC analysis. Major suffixes are as follows:

  - **DC**
  - **AC**
  - **AC noise**
  - Harmonic Balance, P2D, XDB, Envelope, LSSP
  - Harmonic Balance noise
  - Transient

  The .DC suffix is specific to the DC analysis. Major suffixes are as follows:

  - **DC**
  - **AC**
  - **NC**
  - **HB**
  - **HB_NOISE**
  - **TRAN**

- **circuit_path** is the path of the simulation output (node voltage, current, etc.) with respect to the circuit level of the measurement that references it. For example:

  - **MeasEqn1 = Vout**
  - **MeasEqn2 = X1.Vout**

  may reference a node voltage Vout at the current level (hence no path), while

  may reference a node voltage Vout in the subcircuit X1 of the current level (hence the circuit path is X1).

  Unlike node voltages and currents, S-, Y-, Z-parameters and the corresponding delays require no circuit path.

- **name** is the name of the simulation output (e.g. Vout for a node voltage, I_Probe1.i for the current through current probe I_Probe1, S for scattering parameters).

The circuit path and the name are required for proper reference. The analysis path is optional, and may be used in the case where a design contains multiple analyses to differentiate between same-name outputs. The analysis path need not be complete. For example, the node voltage Vout generated through the DC analysis DC1 may be referenced by a same-level measurement as follows:
Preparing a Circuit for Simulation

\[ \text{MeasEqn1} = \text{Vout} \]
\[ \text{MeasEqn2} = \text{DC.Vout} \]
\[ \text{MeasEqn3} = \text{DC1.DC.Vout} \]
but not \[ \text{MeasEqn4} = \text{DC1.Vout} \]

The same resolution rules used for a data display apply here to analysis outputs.

**Using Simulation Templates**

A number of templates are available to facilitate setting up common simulations. Copy these to a directory where you have write permission.

To use a simulation template:

1. Choose **Insert > Template**.
2. From the dialog box that appears, select the desired simulation type and click **OK**. Place the template in the Schematic window and modify it as required.

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BJT_curve_tracer</td>
<td>This simulation uses a swept current source for the base current and a swept voltage source for the collector voltage, to simulate the DC collector current versus collector-emitter voltage curves of a BJT.</td>
</tr>
<tr>
<td>ConvPulseRespT</td>
<td>This simulation uses nonlinear, time-domain analysis to simulate the pulse response of a network. The pulse response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the pulse signal from one line to another can be simulated. If the circuit contains distributed elements, then convolution will be used during the simulation. The reflected and transmitted signals may be shown. Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.</td>
</tr>
</tbody>
</table>

Table 2-1. Simulation Template Descriptions

2-12  Using Simulation Templates
### Table 2-1. Simulation Template Descriptions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvStepRespT</td>
<td>This simulation uses nonlinear, time-domain analysis to simulate the step response of a network. The step response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the step signal from one line to another can be simulated. If the circuit contains distributed elements, then convolution will be used during the simulation. The reflected and transmitted signals may be shown. Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.</td>
</tr>
<tr>
<td>DC_BJT_T</td>
<td>This generates the same I-V curves as the BJT_curve_tracer, except that the sources and simulation controllers are packaged up into a subcircuit.</td>
</tr>
<tr>
<td>DC_FET_T</td>
<td>This generates the same I-V curves as the FET_curve_tracer, except that the sources and simulation controllers are packaged up into a subcircuit.</td>
</tr>
<tr>
<td>FET_curve_tracer</td>
<td>This uses swept voltage sources for the gate and drain voltages, to simulate the DC drain current versus drain-source voltage curves of a FET.</td>
</tr>
<tr>
<td>HB1Tone</td>
<td>This simulation generates the output power, power gain, harmonic distortion, and the output spectrum, when the test signal is a sinusoid at one power and frequency.</td>
</tr>
<tr>
<td>HB1ToneSwptFreq</td>
<td>This simulation generates the frequency-dependent output power, power gain, harmonic distortion, and the output spectrum, when the test signal is a sinusoid at one power and is swept over frequency.</td>
</tr>
</tbody>
</table>
Preparing a Circuit for Simulation

Table 2-1. Simulation Template Descriptions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HB1ToneSwptPwr</td>
<td>This simulation generates the output power, power gain, harmonic distortion, output spectrum, and gain compression, when the test signal is a swept-power sinusoid at one frequency.</td>
</tr>
<tr>
<td>HB2Tone</td>
<td>This simulation generates the output power, power gain, output spectrum, and third- and fifth-order intermodulation distortion points (input- and output-referred) when the test signals are two sinusoids of the same power.</td>
</tr>
<tr>
<td>HB2ToneSwptPwr</td>
<td>This simulation generates the output power, power gain, output spectrum, and third- and fifth-order intermodulation distortion points (input- and output-referred), as well as the intermodulation distortion levels when the test signals are two sinusoids and their power is swept.</td>
</tr>
<tr>
<td>LinearPulseRespT</td>
<td>This simulation uses linear, swept-frequency AC analysis to simulate the time-domain pulse response of a network. The pulse response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the pulse signal from one line to another can be simulated. The reflected and transmitted signals may be shown. Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.</td>
</tr>
</tbody>
</table>
**Table 2-1. Simulation Template Descriptions (continued)**

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinearStepRespT</td>
<td>This simulation uses linear, swept-frequency AC analysis to simulate the time-domain step response of a network. The step response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the step signal from one line to another can be simulated. The reflected and transmitted signals may be shown. Refer to the example file examples/RF_Board/TDRcrosstalk_prj to see this template in use.</td>
</tr>
<tr>
<td>MixConvGainNF</td>
<td>This simulates the conversion gain and noise figure of a mixer.</td>
</tr>
<tr>
<td>MixTOI</td>
<td>This simulates the output spectrum, output power, conversion gain, and third-order intercept points of a mixer.</td>
</tr>
<tr>
<td>S_Params</td>
<td>This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and polar plots for S21 and S12. The Smith charts include a circle of constant VSWR, whose value you may set.</td>
</tr>
<tr>
<td>S_Params_DC</td>
<td>This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and polar plots for S21 and S12. The Smith charts include a circle of constant VSWR, whose value you may set. In addition, it generates zoomed plots of S11 and S21, over a reduced frequency range. A DC simulation is also run.</td>
</tr>
<tr>
<td>SP_BJT_T_T</td>
<td>This is a two-port vector network analyzer equivalent with biasing for a BJ T. It sweeps the base current and collector-emitter voltage, and simulates the S-parameters of the device at each bias point, at one analysis frequency</td>
</tr>
</tbody>
</table>
## Table 2-1. Simulation Template Descriptions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP_DiffT</td>
<td>This simulates the S-parameters of any two-port network, but the test ports are ungrounded. This allows the simulation of differential-mode S-parameters.</td>
</tr>
<tr>
<td>SP_FET_T</td>
<td>This is a two-port vector network analyzer equivalent with biasing for a FET. It sweeps the gate-source and drain-source voltages, and simulates the S-parameters of the device at each bias point, at one analysis frequency.</td>
</tr>
<tr>
<td>SP_NWA_4PortBiasLogT</td>
<td>This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR. It is identical to SP_NWA_4PortT, except that a log frequency sweep is used, and a single DC bias may be set for each of the test ports.</td>
</tr>
<tr>
<td>SP_NWA_4PortBiasT</td>
<td>This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR. It is identical to SP_NWA_4PortT, except that a single DC bias may be set for each of the test ports.</td>
</tr>
</tbody>
</table>
Using Simulation Templates  2-17

Table 2-1. Simulation Template Descriptions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP_NWA_4PortLogT</td>
<td>This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR. It is identical to SP_NWA_4PortT, except that a log frequency sweep is used.</td>
</tr>
<tr>
<td>SP_NWA_4PortT</td>
<td>This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR.</td>
</tr>
<tr>
<td>SP_NWA_LogT</td>
<td>This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and rectangular plots for dB(S21) and dB(S12). It is identical to the SP_NWA_T template, except that a log frequency sweep is used.</td>
</tr>
</tbody>
</table>
Preparing a Circuit for Simulation

### Table 2-1. Simulation Template Descriptions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP_NWA_T</td>
<td>This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and rectangular plots for dB(S21) and dB(S12). A two-port vector network analyzer equivalent instrument is used. In addition, it generates zoomed plots of S11 and S21, over a reduced frequency range. Also, plots showing available gain and stability circles may be created.</td>
</tr>
<tr>
<td>Sparams_wNoise</td>
<td>This simulates the S-parameters and noise figure of any two-port network, and generates Smith chart plots for S11 and S22, and rectangular plots for dB(S21) and dB(S12). The Smith charts include a circle of constant VSWR, whose value you may set. In addition, it generates zoomed plots of S11 and S21, over a reduced frequency range. Also, plots showing available gain, noise figure, and stability circles are created.</td>
</tr>
</tbody>
</table>
Using Simulation Instrument Components

Simulation instrument components provide a method for symbolically connecting your circuit to an instrument. You connect your design to components that represent various instruments and run the simulation.

The instruments are set up as curve tracers, TDRs, and network analyzers. There are two or more of each type of instrument—each one is designed for a particular simulation or measurement. They are located on the Component Palette, under Simulation-Instrument.

To use a simulation instrument:

1. Create your design.
2. From the Component Palette, choose Simulation-Instrument. Select the appropriate instrument and place it on your schematic.
3. Connect the ports of your design to the instrument connectors.
4. Set the instrument parameters.
5. Run the simulation.
Preparing a Circuit for Simulation
Chapter 3: A Simulation Example

This chapter is a detailed simulation example. The circuit is a simple BJT and the simulation is set up to calculate the DC operating point. The simulation process is described here in detail, and this process can be applied to more complex circuits and simulations.

Before continuing with this chapter, you should be familiar with the previous sections of this book, the Quick Tour (online), and the User’s Guide.

The DC_OP_POINT.dsn design used here is located in the Examples directory under MW_Ckts/ LNA_prj. Details on working with example projects can be found in the User’s Guide.

Working through this example consists of these tasks:

- “Placing Circuit Sources” on page 3-2
- “Specifying Points for Collecting Data” on page 3-4
- “Selecting a Simulation Type” on page 3-5
A Simulation Example

• “Selecting a Sweep Type and Plan” on page 3-7
• “Setting Simulation Options” on page 3-8
• “Modifying the Simulation Setup” on page 3-8
• “Starting the Simulation” on page 3-9
• “Displaying Simulation Data” on page 3-10

Placing Circuit Sources

In many cases your circuit will contain sources. In this example, you need to select a voltage source that is appropriate for a DC simulation:

1. From the Component Palette List, choose Sources-Time Domain.
2. Select V_DC (DC Voltage Source) and place this component in the Schematic window.
3. Set the DC voltage (Vdc) to 3.0 V.

Ensuring Sources are Connected Properly

You must ensure that the sources are connected properly to the circuit. You can do this two ways:

• Wire the source directly to your circuit.
• Use node names (Insert > Wire/ Pin Label) to define connections. By labeling the end of the source and a point in the circuit, the source then behaves as if connected physically to that point of the circuit. For example, named connections can be used for Vcc, Vb, and Ve.

If you are wiring components, the color of the pin changes when a successful connection is established. Wires that are simply overlaid on intervening nodes will not be connected to those nodes, and clicking on the intervening nodes after the end node is wired does not ensure that the intervening nodes are connected properly.

In addition, the endpoints of dangling wires (wires that are connected at one end only) can be moved but not edited. To move an endpoint, choose Edit > Move > Move Wire Endpoint. When crosshairs appear, select the open endpoint and move it; the wire will move with it. It is also possible, using standard copy and move commands, to move and copy dangling wires and networks of wires.
To label and share a source:

1. Choose Insert > Wire/Pin Label and a dialog box appears.
2. Enter a name for the node (for example, Vcc).
3. Click the pin of the source you want to label (for example, the positive side of \text{V\_DC}). The label Vcc appears at that node.
4. Next click the pin of a node to which you want Vcc to apply (for example, one end of a resistor). The resistor node is also labeled Vcc. (Note that you must click the component pin, not a wire between components.)

\textbf{Note} \quad \text{To delete a named connection, choose Edit > Wire/ Pin Label > Remove Wire/ Pin Label. When crosshairs appear, click the pin whose connection name you want to remove.}

5. Edit any parameters of the source as needed.
Specifying Points for Collecting Data

You must specify the points in the circuit where voltage and current values will be collected and saved in the dataset:

- To identify points where you want voltages to be taken, use node names (Insert > Wire/ Pin Label) or global nodes (Insert > Global Node). For circuit nodes that are labeled this way, voltages will be calculated during the simulation and saved to the dataset.

- For collecting current values, insert current probes (I_Probe) in your circuit at points of interest. Current probes are found on the Probe Components palette.

- Some components, such as the DC voltage source (V_DC) used in this example, include a SaveCurrent parameter. If this is set to yes, the current flowing through this component will be saved to the data set.

To help identify the data collected by a probe or source, you might want to change the default Instance Name to something more meaningful. For example, you can name a DC voltage source Vsupply. The current through that source will appear in the dataset as Vsupply.i.

Note: Current probes (I_Probe components) must be placed so that the arrow on the probe points in the direction of (positive) current flow. To flip a probe horizontally, choose Edit > Rotate/ Mirror > Mirror About Y.

In general, voltage and current data is in phasor representation, so the voltage values at named nodes are peak voltage.

The process for labeling nodes is described in “Ensuring Sources are Connected Properly” on page 3-2.
To insert a current probe:

1. Select the **Probe Components** palette.

2. Select the **I_Probe** component and place it at the desired point in your schematic. (You may need to move components or rewire to do this.) Edit the probe component and change the Instance Name to something that suggests its purpose in the circuit.

For details on selectively sending data to the dataset, refer to “Selectively Saving and Controlling Simulation Data” on page 1-10.

**Selecting a Simulation Type**

There are a variety of simulation methods to choose from. To use a particular simulator, you must have purchased a license for it. If you do not have a license, a message will appear when you attempt to run the simulator. This may also happen if you share licenses that are already in use.

Each simulator has its own palette, which can be selected from the Component Palette List. Details for using each simulator are described in later chapters.

To add a DC simulation component (for this example):

1. Select the **Simulation-DC** palette.

2. Select and place the **DC** component on your schematic.

**Editing Simulation Parameters**

There are two ways to edit simulation parameters:

- You can edit parameters directly on the schematic. Click somewhere within the parameter value to invoke the on-screen editor, and change the value as desired. Press Return to enter data and go to the next entry. Note that by default, not all parameters are displayed on the schematic; to view/edit additional parameters, use the dialog box, as described next.

- You can edit parameters within the dialog box. To open the dialog box, either double-click the simulation component, or select it and choose **Edit > Component > Edit Component Parameters**. You can also bring up the dialog box using the **Edit Component Parameters** icon on the toolbar.
A Simulation Example

**Note**  For many simulation options, the default parameters should provide satisfactory results and will not need editing.

You may find it useful to display (on the schematic) the options you have selected, to remind you of the parameters governing a simulation.

To display additional parameters on the schematic:

1. Select the **Display** tab.
2. Select each parameter you want to appear and click **OK**.
To exercise certain simulation options, you must first display them on the schematic, then set them equal to the desired value. For example, to send a defined power parameter RF_power to the dataset following a simulation, enable the display of the Other parameter, and then edit it in the Schematic window as follows:

\[ \text{Other}=\text{OutVar}="RF\_power" \]

OutVar is used to define an output variable that is sent to the dataset. To output multiple variables, use multiple OutVar statements, separating them by spaces.

**Selecting a Sweep Type and Plan**

The Sweep tab enables you to identify the parameter that you want to sweep, and then specify the sweep type. You can sweep over:

- A single point
- A linear range
- A logarithmic range

After you select the sweep type, you can then specify the sweep range.

You can also select a sweep plan. A sweep plan enables you to specify a sweep once and then use these settings in other places. For more information, refer to Chapter 19, Parameter Sweeps and Sweep Plans. Note that for some simulators, these settings are found on the Freq tab.

To set the sweep values for this example:

Double-click the DC component to edit it. Set the following parameters as shown:

- Parameter to sweep = VBE
  - Start = 0.6
  - Stop = 0.85
  - Step size = .002
  - Num. of pts. = 126

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and results combined into a single dataset on the local machine. For
A Simulation Example

details on setting up remote and local machines for remote processing, refer to these appendices:

- Appendix E, Using Remote Simulation in the Installation on PC Systems manual

Setting Simulation Options

An Options component can be used with any simulation. The most common use of the Options component is to set the simulation temperature, but it also enables you to specify settings for convergence tolerances, warnings and other advanced options. (For this example, an Options component is not used.)

To add an Options component:

1. Select any simulation palette.
2. Select and place a simulation Options component and double-click to edit it. You can set general simulation options such as temperature, DC convergence tolerances, warnings, and other settings.

For more information about each field, click the Help button at the bottom of the Options Component dialog box.

Modifying the Simulation Setup

Additional simulation setup options include:

- Specifying a name for the dataset to which simulation data will be saved
- Automatically displaying data when the simulation is finished
- Selecting a different machine on which to run the simulation

To specify simulation setup options:

1. From the Schematic window, choose Simulate > Setup. A Simulation Setup dialog box appears:
2. The data calculated during a simulation is saved in a dataset. The default dataset name is the same as the project. Change the name as desired.

3. By default, the Data Display will automatically launch when the simulation is complete. If the simulation results were displayed before, the same window will open if you specify its name in the Data Display field.

4. If you use remote simulation hosts, you can specify a machine other than the local one for running the simulation. For more information on how to set up remote machines, refer to the installation manual for your platform.

Starting the Simulation

There are several ways to launch a simulation:

• Press F7 on the keyboard.
• Click the Simulate icon on the toolbar.
• Choose **Simulate** > **Simulate** from the Schematic window.
• Choose **Simulate** from the Setup dialog box while it is open.

When the simulation begins, a status and error message window appears. When the simulation is complete, the line `Simulation finished` at the bottom of the window indicates that the simulation has run successfully. The location of the dataset where the simulation data is saved is also noted.
A Simulation Example

Displaying Simulation Data

You will typically want to view most of your results in a Data Display, however you also have the option to view DC results on the schematic and view lists of device operating point details. While it is not very useful in this example—given that the DC results are reported only for the last point in a sweep—it illustrates an important feature.

Viewing the DC Solution

1. Choose Simulate > Annotate DC Solution. DC voltages and currents appear at the pins of all the active devices and lumped elements, as shown below.

**Note**  Current is defined as positive if flowing into a device, so +13.5 mA is flowing from the emitter to ground.
**Note** To clear the annotations, choose Simulate > Clear DC Annotation from the main menu. You will have to resimulate to annotate again.

**Viewing the Detailed Device Operating Point**

1. In the same Schematic window, choose Simulate > Detailed Device Operating Point. Crosshairs appear.

2. Place the crosshairs over a transistor and click. A detailed DC operating point listing appears.

3. As you select other devices in a circuit, the DC operating point data for additional devices is added to the list.

For more information about the parameters that are displayed in the list, refer to the documentation for the specified component.
**A Simulation Example**

**Viewing the Brief Device Operating Point**

To view a subset of the above information that covers the most common parameters, choose Simulate > Brief Device Operating Point and select a device. The details are similar to those in the detailed list, but this list contains fewer parameters.

For details about the data that appears in the list, refer to the selected component's documentation.

**Viewing More Results in the Data Display**

The remainder of the simulations results can be viewed using the Data Display. For complete details on how to use the Data Display, refer to the Data Display manual and to the Quick Tour (online). The results of this simulation are in DC_op_point.dds. In this example, collector current versus $V_{be}$ at Probe1 is displayed.

There are various options for plotting and scaling data. To edit general plot characteristics, select the Plot Options tab. This allows you to enter a title and axis label. It also allows you to deselect Auto Scale and enter a scale that zooms in on a range of interest. You can also choose between linear and log scales here, and select grid characteristics.

To edit a trace before placing it (unless you have selected the List plot type), select the parameter under Traces, then click Trace Options. Select the Trace Type, Trace Options, and Trace Expressions tabs to select, for example, trace patterns, trace colors, and fonts for labels, as well as to edit mathematical expressions for display.
Chapter 4: DC Simulation

The DC Simulation component, in the Simulation-DC palette, calculates the DC operating characteristics of a design under test (DUT), including how much power the circuit consumes.

You can also sweep one or more parameters, enabling you to perform tasks such as verifying model parameters by comparing the simulated DC transfer characteristics (I-V curves) of the model with actual measurements.

Refer to the following topics for details on DC simulation:

- “Performing a DC Simulation” on page 4-2 has the minimum setup requirements for a DC simulation.
- “Examples” on page 4-3 contains detailed DC simulation setups, one for calculating a single DC point and another that sweeps variables to generate a set of I-V curves.
- “DC Simulation Overview” on page 4-8 is a brief explanation of the DC simulation process.
- “Troubleshooting a Simulation” on page 4-9 offers suggestions on how to improve a simulation.
Performing a DC Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis:

• Add the DC simulation component to the schematic. If you do not edit the simulation component default settings, values will be calculated for the current probes and nodes that you have applied to the circuit, based on the settings of the other components in the schematic.

• To sweep over a range, such as varying input voltage or changing a resistor value, double-click the simulation component and select the sweep tab. Enter the name of the parameter to sweep. Select the sweep type and enter the range.

For details about each field, click Help from the open dialog box.
Examples

There are two examples that describe how to set up and run a DC simulation:

- “Simulating a BJT” on page 4-3 shows how to calculate a single-point DC bias of a BJT.
- “Sweeping Parameters” on page 4-5 shows how to sweep one or more circuit parameters over a range of values during a simulation.

Simulating a BJT

Figure 4-1 illustrates the setup for a DC simulation of a BJT. This simulation calculates a single DC operating point.

Note This design, DC1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in DC1.dds

Figure 4-1. Setup for a Single-Point DC Bias Simulation
To perform a single-point DC simulation:

1. From the **Sources-Freq Domain** palette, select **V_DC**. Place the DC voltage source on the schematic to provide collector voltage and modify the voltage as needed.

   **Note**  
   Ensure that sources are connected either by wires or by means of a named connection (Insert > Node Name).

2. From the **Sources-Freq Domain** palette, select **I_DC**. Place the source on the schematic to provide base current and modify the current as needed.

3. From the **Probe Components** palette, select **I_Probe**. Place this current probe in an appropriate location in the circuit.

4. From the **Simulation-DC** palette, select and place the **DC** simulation component on the schematic and edit the parameters and options as needed.

5. Click **OK** to accept changes and close the dialog box.

6. **Simulate**. When the simulation is finished, you can immediately exercise the following DC simulation options:
   - For a direct representation of currents and voltages at all nodes on the schematic, choose **Simulate > Annotate DC Solution**.
   - For the details of the device alone, choose **Simulate > Detailed Device Operating Point** or **Brief Device Operating Point**, then click the device when the crosshairs appear.
   - Choose **Simulate > Clear DC Annotation** to remove the annotation.

7. In the Data Display, the data from the simulation is presented as a list of DC current and voltages:

<table>
<thead>
<tr>
<th>freq</th>
<th>DC1.Vc</th>
<th>DC1.Vo</th>
<th>..._Probe1.I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000 Hz</td>
<td>3.000 V</td>
<td>678.8 mV</td>
<td>61.02 uA</td>
</tr>
</tbody>
</table>
Sweeping Parameters

By performing a DC swept bias or a swept variable simulation, you can check the operating point of the circuit against a swept parameter such as temperature or bias supply voltage. The resulting data can be plotted to resemble the appearance of a curve tracer.

Figure 4-2 illustrates a setup for performing an idealized, swept DC bias simulation of a MOSFET. Vdd and Vgg are swept across a range of voltages. The result is a family of curves representing drain current versus drain voltage, for varying values of gate voltage.

Note  This design, DC2.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in DC2.dds.

---

Figure 4-2. Example Setup for a Swept DC Simulation
**Note**  This example uses a referenced model. The parameter `cells` refers to the extraction parameter known as binning and reflects the electrical characteristics of this particular model. The number of cells has been referenced in a VarEqn component.

To perform an idealized, swept DC bias simulation:

1. From the **Sources-Freq Domain** palette, select `V_DC`. Place this V_DC component (SRC1) between the base of the device and ground. Edit this component and define Vdc = Vgg1, rather than a numerical value.

2. Place another V_DC component (SRC2) between the drain of the device and ground. Open this component and define Vdc = Vdd1, rather than a numerical value.

   This makes it possible to sweep these values. Any swept parameter must be initialized by means of an equation, as will be demonstrated later.

3. From the **Probe Components** palette, select `I_Probe`. Place the probe between the drain and SRC2, and rename it if you like (in this case, it has been called `Probe1`).

4. Edit the DC component by selecting the **Sweep** tab and modifying the sweep parameters as follows:
   - Parameter to sweep = `Vdd1` (This appears as SweepVar on the schematic if the display for this parameter is turned on.)

   **Note**  Variables entered in this field will appear in quotes on the schematic. If you enter a variable directly on the schematic (in this case, as the right-hand side of the SweepVar statement of the DC component), you must surround the variable with double quotes. This applies to the other user-defined variables shown in subsequent steps.

   - Start = 0
   - Stop = 6
   - Step = 0.1

5. Click **OK** to accept changes and close the dialog box.
6. Choose Simulation-DC palette, select PSWP (ParamSweep). Place this component on the schematic and edit it as follows:

On the Sweep tab:
• Parameter to sweep = \textit{Vgg1}
• Start = 2
• Stop = 3
• Step = 0.25

On the Simulations tab:
• Simulation 1 = DC1 (This appears as SimInstanceName[1] on the schematic if the display for this parameter is turned on.)

7. Click OK to accept changes and close the dialog box.

8. Choose Data Items palette, select Var eqn (Variables and equations). Place this component on the schematic and enter the following equations:
• \textit{Vdd1} = 4
• \textit{Vgg1} = 2.5

9. Click OK to accept changes and close the dialog box.

10. Simulate. When the simulation is finished, plot probe current (Probe1.i). The resulting family of curves representing drain current versus drain voltage, for varying values of gate voltage, is shown next:
DC Simulation

DC Simulation Overview

A DC simulation is run with inductors shorted, capacitors opened, and all time-varying sources set to their quiescent values.

The DC simulation technique relies on an iterative process of mathematical convergence toward a solution. If a sweep is performed, the convergence test is performed at the first point in the sweep and the remainder of the sweep is disabled.

Simulation Assumptions

DC voltages and currents are considered signals of zero frequency. The simulator uses this concept when performing a DC simulation, and the following conditions apply:

- Linear elements are replaced by their (real) conductances at zero frequency.
- Capacitors, microstrip gaps, AC coupled lines, and similar items are treated as open circuits.
- Inductors, conductive discontinuities, and similar items are virtual short circuits.
- Transmission lines are replaced by DC conductance values calculated from their length, cross-sectional area, and conductivity.
- Scattering parameter (S-parameter) files must include zero frequency data to operate properly at DC (this is also required for harmonic balance analysis). Otherwise, the simulator extrapolates each S-parameter for the zero-frequency case, and uses the real part as the DC response.
- The simulator has built-in safeguards against nodes that are DC-isolated (that have no DC path to ground), as well as against DC source-inductor loops. Nevertheless, try to avoid these conditions.

To improve the convergence process during swept DC bias simulation, the simulator uses the results from the previous bias point as an initial guess for the next bias point. The results of the swept DC bias simulation are saved in a binary form in a temporary file. This file is used to postprocess selected measurements. Because this file contains all of the results, new measurements can be specified and displayed without the need for a subsequent simulation.
Effect of DC Simulator on Other Simulations

A single-point DC bias simulation automatically precedes every transient, harmonic balance, and bias-dependent linear simulation. This provides starting points for those simulations. In the case of an AC simulation, it determines linearized models for the nonlinear components and for harmonic balance it determines an initial estimate. If the bias simulation fails or is incorrect in some way (because of, for example, incorrect nodal connections), the harmonic balance or bias-dependent linear simulations will fail or show incorrect results.

Troubleshooting a Simulation

This section presents suggestions for using this simulator and improving the accuracy of results.

Circuit Does Not Behave as Expected

In the case of unusual circuit behavior, try any or all of the following:

- Check that the circuit is biased correctly.
- Move the point at which the signal is applied through the circuit.
- Break the circuit into sections and verify the performance of each section.

Simulation Time

Simulation time is affected by the type of circuit, as well as by the number of the fundamental frequencies and their orders. It is also affected by the convergence criteria, which can be set in the Options component.
Type of Circuit

It is possible to simulate both linear and nonlinear circuits. However, when analyzing nonlinear circuits, it is important to keep in mind the following points:

- The system can simulate amplifiers, filters, frequency multipliers, mixers, detectors, and oscillators that have lumped and distributed components. These components can be ideal or measured.

- Except for circuit envelope and transient simulations, the simulator uses sinusoidal waveforms. Therefore, simulation times will be shorter for circuits whose waveforms more closely resemble a sinusoid. Circuits whose waveforms are not smooth require more simulation time.

- The more nonlinear a circuit is, the longer the simulation time, because the waveforms are not sinusoidal.

When the Simulator Fails to Converge

The DC operating point of active devices such as transistors and diodes must be determined before a high-frequency simulation can be performed. Consequently, a DC simulation is performed as the first step of any nonlinear simulation. When the simulator cannot reach a solution within a given tolerance, after a given number of numerical iterations, it is said to have failed to converge. There is no one specific solution for solving convergence problems, but the following suggestions may provide assistance.

One approach is to modify the convergence criteria used by the simulator. This will also affect the speed and accuracy of the solution. You may also increase the number of iterations used in the simulation. If you have a single-point DC or coarsely swept DC bias problem, try using a refined sweep with small step sizes.

Ideally, convergence occurs when, at each node, the voltage change between iterations is zero, and the sum of the currents at each node is zero (Kirchoff’s Current Law). In practice, less-exact solutions are acceptable; the Options component allows you to set the level of tolerance.
When Resulting I-V Curves are Not as Expected

If your DC I-V curves appear different than expected, this is probably due to one or more of the following reasons:

- Your independent swept parameters are swapped on the grid display. For example, if SRC1 sweeps a transistor gate voltage and SRC2 sweeps a drain voltage, and if the current is displayed with SRC1 (gate voltage) on the x-axis, these traces will not look like typical DC I-V traces. Exchange the order of the swept variables SRC1 and SRC2 on the grid.

- You are sweeping one source with large variations and another with small variations. For example, say that you are sweeping the base current and the collector voltage of a transistor by connecting a DC current source at the base and a DC voltage source at the collector. Furthermore, say that you are using SRC1 to sweep the base current and SRC2 to sweep the collector voltage. A DC bias-current sweep typically involves very small increments—on the order of microamperes, for example. Inside the program, the SRC1 sweep always occurs inside the SRC2 sweep, so the net effect is that the base current is swept faster.

Because of the small increments in the current sweep, proceeding from one current value to the next may not change the circuit state (operating point) in the simulator. This causes the program to converge to the same solution as the previous current value, resulting in jagged I-V curves.

- You are measuring the DC current in the wrong direction. By default, positive current through a two-terminal element, such as a resistor or ammeter, is measured as flowing from pin 1 to pin 2.

To measure current in the opposite direction, you can edit the schematic by changing the element's orientation, or specify the current at a different pin, or change the orientation of the current probe.
Eliminating DC Open-Circuit Errors

If the DC circuit simulation is to succeed, all circuit nodes must have a DC path to ground. This eliminates nodes that float at an undefined DC voltage value and allows the DC simulation to find the bias point of all nodes in the circuit. This error is commonly caused by two capacitors in series.

- To eliminate this error, insert a large resistor (10 Mohm is a good value) from the offending node to ground. This will add a DC path to ground without affecting the circuit’s performance.

It is possible to create a circuit containing short-circuited DC loops that cannot be handled by the DC simulation. For instance, if two ideal inductors are connected in parallel, they create a DC short-circuit loop. The DC current in this loop is undefined; it can have any value without affecting the circuit solution.

- To eliminate this error, add a small DC resistance (say, 0.001 ohm) in series within the loop. Or, if the loop is made up of ideal inductors, simply change the ideal inductors to lossy inductors with a small amount of loss.

Finally, as a last resort, turn off the topology checker. In the Options component, make sure Perform topology check is not selected. This will turn off all topology checking, and simulations will proceed directly to the DC simulation stage. This will speed up simulations slightly, and may be helpful in rare cases where it is certain that the DC simulation will succeed despite the topology.

Note: Errors found by the topology checker almost always lead to DC simulation failures, so turning off the topology checker is not recommended.
Topology Check Failures

In general, a topology check is performed before a simulation is run. The topology check finds common circuit errors that would lead to more serious errors later in the simulation and reports these in the Simulation/Synthesis Messages window. The check verifies the existence of the following conditions:

- At least two components are connected to each node.
- There is a DC path from every node to ground.
- There are no loops of DC shorts (for example, inductors) or voltage or current sources.

The topology checker is not able to check these conditions through transformers or transmission lines.

Checking Nodes and Pins

If the topology check fails, the offending nodes are highlighted in the Simulation/Synthesis Messages window. Choose Simulate > Highlight Node to identify these nodes.

Unconnected Wire Errors

In the case of a circuit with a wire that is missing, a message such as the following can appear in the Simulation/Synthesis Messages window during a simulation:

Warning detected by HPEESOF SIM during DC analysis ‘DC1’.
Circuit as given has no unique solution.
A virtual resistance of 1 TOhms was added to each node.

In the case of unconnected wires or pins, choose Options > Check Representation > Unconnected pins and click OK. All unconnected wires and components are highlighted, making them easily visible on the screen. Connect these to the correct circuit node.
DC Simulation

**Port/Pin Mismatches**

To check for mismatches between ports and named pins on an underlying schematic, use the Check Representation feature. Choose Options > Check Representation, then select Port vs Pin mismatch (symbol/schematic). In a circuit without a topology problem, the simulator reports the following:

Unconnected pins: 0  
Pin vs. Port (schematic/symbol) mismatches: 0

Where a wire is missing between two pins, for example, the components and the disconnected pins will be highlighted in red on the schematic, and a message such as the following will appear, indicating the coordinates of the connectors:

Unconnected pins: 2  
I_DC SRC1, pin 2 (2.500,–0.125)  
BJT_NPN BJT1, pin 2 (3.250,0.375)

To clear the highlighting, choose View > Clear Highlighting.

---

**Note**  Always ensure that there is no duplication of port numbers.

---

**No Solution Found**

If the DC simulation cannot find a solution, then the simulator will set all DC sources to zero and attempt to sweep their value up slowly to the desired level, using the results of each simulation as the initial guess for the next. Normally, the circuit response follows the DC source values in a smooth curve up to the desired value. Very rarely, circuits can exhibit a bizarre behavior resulting in a circuit response that diverges as the source values are increased. The simulator tolerates a certain amount of this behavior; however, if the circuit response diverges beyond specific boundaries, the simulator will give up.
Increasing Iterations

Occasionally the DC simulation cannot converge on a solution within a given number of iterations.

To increase the number of iterations:

1. In the DC Simulation dialog box, select the Parameters tab.
2. Set Max. Iterations to a value high enough to allow convergence.

Impossible Circuits

A circuit can fail to have a unique solution, as in the case of impossible circuits—such as circuits with two DC current sources (or, perhaps, current-mirror circuits) in series. The simulator attempts to analyze these circuits by adding a large resistor in parallel with every node in the circuit. If the DC simulation then succeeds, the resistor value is increased as high as possible (up to 1 TOhm). This allows the DC simulation to proceed, and usually to succeed. However, the offending circuit node or nodes often cause problems with a subsequent harmonic balance simulation, because the harmonic balance simulator does not add these resistors. It is best to find the problem node and add a DC path to ground (such as a large resistor).

If virtual resistors were added to each node during DC simulation but the simulation still failed, look for circuit problems such as loops of inductors and voltage sources.

Highly Nonlinear Circuits

Occasionally the default DC simulation algorithm fails in the case of highly nonlinear circuits. Select from the different convergence algorithms under the Parameters tab to find the one that works the best.

If the DC simulation still fails, try adjusting tolerance parameters, described next. Otherwise, the only recourse is to attempt to modify the circuit so that nonlinearities have less of an impact on the DC simulation.
DC Simulation

**Adjusting Tolerance Parameters**

Several tolerance parameters can be changed so that DC simulations are more likely to converge, although at the cost of a somewhat less accurate result. These parameters include:

- Current relative tolerance
- Voltage relative tolerance
- Current absolute tolerance
- Voltage absolute tolerance

To change any of these parameters, edit them in an Options component.

The tolerance parameters are described next.

**Current Tolerance, Relative and Absolute**

These tolerances are used to satisfy Kirchoff’s Current Law (KCL) in solving for the currents at each node in the circuit. The simulator attempts to find a solution that satisfies KCL, so that the sum of the currents entering (or leaving) all circuit nodes is zero. At each iteration, it uses Current relative tolerance and Current absolute tolerance as a tolerance for the node currents. For convergence to be achieved, the currents must satisfy the following at each circuit node:

\[
|\sum_i i_n| \leq \varepsilon_{rel} \sum_i |i_n| + \varepsilon_{abs}
\]

where

- \(i_n\) = Current in each branch connected to the node
- \(\varepsilon_{rel}\) = Current relative tolerance
- \(\varepsilon_{abs}\) = Current absolute tolerance

The default value for Current relative tolerance is \(10^{-6}\) (0.0001 percent), and the default value for Current absolute tolerance is \(10^{-12}\) (1 pA). For many problems, these tolerances are much tighter than they need to be. (The default value of Current relative tolerance in Berkeley SPICE 3e1 is \(10^{-3}\).) Relaxing these tolerances not only allows problem circuits to be solved, but it also allows them to be solved in less time.
Voltage relative tolerance, Voltage absolute tolerance

Once Kirchoff’s law is satisfied for all nodes, the simulator checks for unique solutions by calculating all node voltages. Sometimes, large changes in node voltages cause very little change in node currents. For instance, if two S-parameter blocks (that is, any 2-port, such as an amplifier or filter, for which there are measured S-parameters) are cascaded, and the reference node between the two components is not grounded, then the differential voltage between the two S-parameter blocks can have any value at all without changing the currents. The circuit then has multiple possible solutions. To find the correct solution for all node voltages, the simulator will use the Voltage relative tolerance and Voltage absolute tolerance parameters in a manner similar to the way it uses Current absolute tolerance and Current relative tolerance. For convergence, the following relationship must be satisfied for every node voltage in the circuit:

$$|\Delta V_n| \leq \varepsilon_{rel}|V_n| + \varepsilon_{abs}$$

where

- $\Delta V_n$ = Change in the node voltage solution from the previous iteration
- $V_n$ = Node voltage found in this iteration of the solution
- $\varepsilon_{rel}$ = Voltage relative tolerance
- $\varepsilon_{abs}$ = Voltage absolute tolerance

The default value for both Voltage relative tolerance and Voltage absolute tolerance is $10^{-6}$. Like Current absolute tolerance and Current relative tolerance, these tolerances can be loosened to help with simulation convergence and speed.
DC Simulation
Chapter 5: Linear AC and Noise Simulation

The AC simulation component, in the Simulation-AC palette, performs a small-signal, linear AC analysis. As part of the analysis, the DC operating point is calculated and any nonlinear devices are linearized around that operating point. This analysis does not generate harmonics or exhibit compression. An AC simulation enables you to obtain small-signal transfer parameters, such as voltage gain, current gain, transimpedance, transadmittance, and linear noise.

Refer to the following topics for details on Linear AC and Noise simulation:

- “Performing an AC Simulation” on page 5-2 has the minimum setup requirements for an AC simulation.
- “Examples” on page 5-3 describes in detail how to set up a basic AC simulation and how to calculate noise.
- “AC Simulation Overview” on page 5-6 is a brief description of the AC simulator.
- “Linear Noise Overview” on page 5-8 describes how noise is calculated as part of an AC simulation.
Performing an AC Simulation

An AC simulation is performed in the frequency domain. You can simulate a single frequency point, or across a frequency span in a linear or logarithmic sweep.

To perform an AC simulation, create your circuit, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis, be sure to:

• Add the AC simulation component to the schematic. Double-click to edit it. Fill in the fields under the Frequency tab:
  • Select the sweep type. For a single point, enter the frequency. For a linear or logarithmic sweep, elect to define the sweep with start/stop or center/span values.

• To calculate noise, select the Noise tab and enable Calculate noise. You select a node for noise calculations from the Edit list, then click Add. Use the Mode list to sort the noise contributed by individual noise sources by name or value.

• You can enable frequency conversion, which is useful when analyzing circuits with standard (not user-defined) behavioral mixer models. For more information on this option, refer to “Enabling Frequency Conversion” on page 5-6.

• You can perform budget calculations as part of the simulation. For more information on budget analyses, refer to Chapter 18, Using Circuit Simulators for RF System Analysis.

For details about each field, click Help from the open dialog box.
Examples

This section gives detailed setups to perform an AC simulation for:

- “Analyzing a Power Amplifier” on page 5-3
- “Calculating Linear Noise” on page 5-5
- “Calculating Port Thermal Noise” on page 5-5

The examples show how to simulate a power amplifier and display the amplifier output, how to calculate linear noise, and how to calculate port thermal noise.

Analyzing a Power Amplifier

Figure 5-1 illustrates the setup for an AC simulation of a power amplifier.

---

**Note** This design, AC1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in AC1.dds.

---

In this example, no variables are swept. The output load resistor has been labeled Vout, to represent the AC output voltage.

To perform a basic AC simulation:

1. From the Sources-Freq Domain palette, select V_DC. Place this component on the schematic and edit the component so that \( V_{dc} = 5 \text{ V} \).
2. From the Sources-Freq Domain palette, select V_1Tone. Place this component on the schematic and edit the component so that \( V_{ac} = 1 \text{ V} \). You can optionally use a V_AC component.
3. From the Simulation-AC palette, select AC. Place this simulation component on the schematic and edit it to select the Frequency tab. Ensure that Start/Stop is selected, then set the following values:
   - **Sweep Type** = Linear
   - **Start** = 1 kHz
   - **Stop** = 100 MHz
   - **Step** = 10 MHz
4. Click **OK** to accept changes and close the dialog box.

5. **Simulate.** When the simulation is finished, a Data Display window opens. Plot **Vout.** The following illustration shows a plot of AC output voltage (Vout), in dB, versus frequency:
Calculating Linear Noise

You can simulate linear noise by setting the appropriate options via the Noise tab of the AC Simulation component.

To perform a linear noise analysis:

1. In the circuit to be simulated, ensure that you have named the nodes at which noise data are to be reported (for example, Input and Output).
2. Edit the AC Simulation component and select Noise, then select Calculate Noise.
3. In the Edit field, enter the names of nodes at which you want noise data to be reported.
4. Use the Mode popup menu to obtain the noise contributed by individual noise sources and sort these contributions by name or value. To obtain just the total noise value at the output only, set Mode to Off.
5. Either accept the default for Bandwidth, or edit it as required. The default is often sufficient.
6. Enter a value in the Dynamic range to display field, in dB. The default is often sufficient if Mode is set to On.
7. Click OK to accept changes and close the dialog box.
8. Launch the simulation. To display and plot resulting data, refer to the Data Display manual.

Calculating Port Thermal Noise

To simulate port thermal noise:

Proceed as in “Calculating Linear Noise” on page 5-5, with the following additional steps prior to launching the simulation:

• Ensure that you have placed input and output ports in the circuit.
• Select the Include port noise option

For information on how noise is calculated, refer to “Linear Noise Overview” on page 5-8.
AC Simulation Overview

When an AC small-signal simulation is run, the system first computes the DC operating point of the circuit. Whenever a linear simulation such as a linear AC simulation requires a single-point DC bias simulation to be run first, it is referred to as a bias-dependent linear simulation. The most common example is the case of a linear amplifier that uses a biased transistor as the active element. The DC bias simulation is executed automatically and transparently (unless an error causes the DC simulation to fail to converge). Following the DC bias simulation, the simulator linearizes all nonlinear devices about their bias point. The resulting linear circuit is then simulated over the specified frequency range. Small-signal AC simulation is also performed before a harmonic-balance (spectral) simulation to generate an initial guess at the final solution.

Simulation can be performed repeatedly while sweeping some parameter. If changing these parameters affects the DC operating point, the DC operating point and linearized circuit will be recomputed at each step.

**Note**  If the circuit has only one AC source, it is often convenient to set its magnitude to one and its phase to zero. In this way, the small-signal transfer function is computed directly.

Enabling Frequency Conversion

Traditional small-signal AC analysis is truly linear in the sense that frequency conversion effects do not occur. In RF system simulation, however, it is common to have frequency-translating mixer components that have approximately linear RF-to-IF conversion characteristics under small-signal RF drive. By enabling frequency conversion (also known as FCAC), you can perform system-level small-signal analyses on such systems. As is the case in standard AC analysis, a noise option is also available.

At the beginning of each FCAC simulation, a so-called frequency map is established. This map specifies the frequencies present at the various circuit nodes, and is based on the frequencies of the sources and the types of behavioral mixer components present in the network. Each node in the network can have only one frequency associated with it. Consequently, each behavioral mixer component can model
frequency conversion to either the upper or the lower sideband, but not to both simultaneously.

Sources most often used for FCAC analysis include the V_1Tone, I_1Tone, and P_1Tone components. The frequency used by the source is given by the Freq parameter. If a multitone source is used, the frequency is specified by the Freq[1] parameter. When no frequency is explicitly specified, voltage and current sources default to the global value of the freq variable, while port sources simply become passive. Small-signal amplitudes used for FCAC analysis are given by Vac, Iac, and Pac parameters for voltage sources, current source, and ports, respectively.

**Note** It is not possible to use FCAC analysis accurately with user-constructed circuit-level mixer blocks (such as diode mixers, Gilbert cell mixers, and the like).
Linear Noise Overview

Linear noise simulation is an option available with the AC and S-parameter simulators. The frequency at which the noise is analyzed is the same as the AC simulation frequency. Noise voltages and currents are saved in the dataset with the keyword Noise included in the parameter name to identify the type of simulation.

The simulator performs an arbitrary-topology, multiport, network noise simulation. The following noise contributions are included in this simulation:

- Temperature-dependent thermal noise from lossy passive elements, including those specified by data files
- Temperature and bias-dependent noise from nonlinear devices
- Noise from linear active devices specified by 2-port data files that include noise parameters
- Noise from noise source elements

The noise simulation computes the noise generated by each element, and then determines how that noise affects the noise properties of the network. In most cases, the noise generated by circuit elements is calculated automatically. Lossy passive elements, for example, contribute noise according to their ability to deliver thermal noise power. The noise contributions from nonlinear devices are computed by models that include temperature and bias dependence; those models are similar to those used by SPICE. The computation of network-level noise properties from the component elements is performed by means of noise-correlation matrices. Most noise measurements are based on either noise figure or noise parameter calculations, which are defined for 2-port networks only.

Note: The temperature of lossy passive elements is used to calculate their noise contributions. Since a lossy passive element at a physical temperature of 0 K does not generate any thermal noise, you may want to disable the noise contribution of any such element by setting its physical temperature to –273.15 degrees Celsius (0 K). Use the Noise parameter in resistors to enable (Noise = YES) or disable (Noise = NO) noise generation. Do not use this method for nonlinear devices such as transistors. There is no way to disable nonlinear device noise.
The program's nonlinear device models include one or more of the following noise effects:

- Thermal noise generated by the resistances that exist within the nonlinear device models. This noise is proportional to the device temperature and is independent of bias.

- Channel noise for JFET, MESFET, HEMT and similar devices. This noise may be due to thermal noise, high-field diffusion noise, or other effects. This noise is generally a function of device temperature and bias.

- Shot noise is caused by the quantized and random nature of current flow across junctions and is modeled for diodes and BJTs. This noise is proportional to the device bias current and is independent of temperature.

- Flicker (1/f) noise is modeled in most nonlinear devices.

- Burst (or popcorn) noise is another low-frequency, bias-dependent noise effect modeled in bipolar transistors.

**Noise Parameter Definitions**

Noise parameters are used to define the noise electrical properties of an n-port electrical element at a given frequency. The noise parameters over a range of frequencies define the element's performance for all noise-power spectral density and tones that define an incident noise source.

Definitions of noise parameters can be found in standard textbooks covering electrical circuit theory. Noise parameters are used by the program to define the noise properties of any electrical element. The following discussion is for a 2-port element, but may be generalized for any n-port element.

A 2-port element noise-wave representation may use two noise waves at the element input. Otherwise, it may use one noise wave at the element input and one at the element output. A multiport-element noise-wave representation has one noise wave at each element port.

In the following noise discussions, the noise is assumed to be spot noise with a bandwidth of 1 Hz.
The spot noise figure is the ratio of the output noise power per unit bandwidth to the portion of output noise power that is attributable to the thermal noise in the input termination per unit bandwidth. The noise temperature of the input termination is assumed to be 290 K.

The noise correlation matrix, \([N]\), is defined as follows:

\[
[N] = \begin{bmatrix}
N_{11} & N_{12} \\
N_{21} & N_{22}
\end{bmatrix} = \begin{bmatrix}
\langle b_n b_n^* \rangle & \langle b_n a_n^* \rangle \\
\langle b_n^* a_n \rangle & \langle a_n a_n^* \rangle
\end{bmatrix}
\]

where * represents the complex conjugate.
Chapter 6: S-parameter Simulation

Using the S-parameter simulation component (S_Param) from the Simulation-S_Param palette enables you to:

- Obtain the scattering parameters (S-parameters) of a component, circuit, or subnetwork and convert those parameters to Y- or Z-parameters.
- Simulate group delay or linear noise.
- Simulate the effects of frequency conversion on small-signal S-parameters in a circuit employing a mixer. (This is also known as analyzing a frequency-translating network.)

The Simulation-S_Param palette also contains components for general simulation options and sweeps, as well as a variety of measurement components for calculating relevant measurements.

Refer to the following topics for details on S-parameter simulation:

- “Performing an S-parameter Simulation” on page 6-2 has the minimum setup requirements for an S-parameter simulation.
- “Examples” on page 6-3 give detailed setups for running a basic S-parameter simulation on an amplifier, as well as examples for calculating group delay, linear noise, and frequency translation.
- “S-parameter Simulation Overview” on page 6-10 is a brief description of the S-parameter simulator and some of its methods, such as group delay and frequency conversion.
- “Noise Analysis” on page 6-14 gives some of the equations and techniques that are the basis of noise calculations.
Performing an S-parameter Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis, be sure to:

- Apply ports to all inputs and outputs. Use P_1Tone or P_nTone power sources to drive inputs. Terminate all other ports using port-impedance terminations (Term). Verify impedance. The Term component is found on the Simulation-S_Param palette. Power sources are on the Sources-Freq Domain palette.

- Check the Num field for each port. The S-parameter port numbers are derived from these fields. For a 2-port circuit, you would want the input labeled as Num=1 and the output as Num=2.

- Add the S-parameter component to the schematic and double-click to edit it. For a basic simulation, fill in the fields under the Frequency tab:
  - Select the Sweep type, single point, logarithmic, or linear. For a linear or logarithmic sweep, elect to define the sweep with start/stop or center/span values.
  - To calculate admittance or impedance parameters, enable the options under the Parameters tab.
  - Group delay calculations can be enabled from the Parameters tab.
  - You can enable frequency conversion, which is useful when analyzing circuits with standard (not user-defined) behavioral mixer models. This option is under the Parameters tab.
  - To calculate noise, select the Noise tab and enable Calculate noise. You select a node for noise calculations from the Edit list, then click Add. Use the Mode list to sort the noise contributed by individual noise sources by name or value.

For details about each field, click Help from the dialog box.

For more detailed descriptions of simulation setups, refer to “Examples” on page 6-3.
Examples

This section contains examples for:

- “Simulating an Amplifier” on page 6-3
- “Calculating Group Delay” on page 6-6
- “Simulating Linear Noise” on page 6-7
- “Analyzing a Frequency Translating Network” on page 6-8
- “Eliminating Unwanted Effects” on page 6-9

These examples give detailed descriptions for setting up and running S-parameter simulations.

Simulating an Amplifier

Figure 6-1 illustrates an example setup for performing a basic S-parameter simulation of an amplifier.

---

Note  This design, SP1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in SP1.dds.
To perform a basic S-parameter simulation:

1. From the Simulation-S_Param palette, select a Term component for each port of the component or circuit to be simulated. You can edit the impedances as required, although the default value of 50 ohms is generally sufficient. Ensure that the terminations are properly connected to the component or circuit under test.

2. Ensure that the number of the input Term component is set to Num = 1, and that of the output Term component to Num = 2.

**Note** By default, the Term component provides a noise contribution (Noise = yes), but is inactive unless noise contributions are requested. Also, ensure that the number of each Term component (as defined by the component’s Num parameter) is appropriate to the location of the component in the circuit, to ensure that the S-parameter data is meaningful.
3. From the Simulation-S_Param palette, select SP. Place this component on the schematic and select the Frequency tab. Ensure that Start/Stop is selected, then set the following parameters:
   - Sweep Type = Linear
   - Start = 800 MHz
   - Stop = 900 MHz
   - Step-size = 1 MHz

4. To obtain S-parameters, select the Parameters tab and ensure that S-parameters is selected. For a description of the options on the Parameters tab, click Help. To obtain Y- (admittance) and Z- (impedance) parameters, select the corresponding options.

5. Click OK to accept changes and close the dialog box.

6. Simulate. When the simulation is finished, plot S(2,1) in the Data Display. The following is a plot of the gain (S_{21}) versus frequency.
Calculating Group Delay

By measuring the transit time, with respect to frequency, of a signal through the device under test, group delay is a useful measure of phase distortion in components such as amplifiers and filters.

To calculate group delay, you enable the Group delay option and, if desired, set the group delay aperture. These options are under the Parameters tab. The results appear in the Data Display variables list under delay.

For more information, refer to “Group Delay” on page 6-12.

To calculate group delay:

1. Proceed as in “Simulating an Amplifier” on page 6-3, setting frequencies and sweep parameters as needed.
2. Edit the S-Param component, select the Parameters tab, and enable Group delay.
3. Group delay aperture is an option that is found on network analyzers and behaves similarly here. The simulator sets the frequency aperture to 0.01% of the current frequency. To override the default frequency aperture, enable Group delay aperture and edit the value as needed.
4. Click OK to accept changes and close the dialog box.
5. **Simulate**. When the simulation is finished, plot the group delay data items, identified by the prefix delay. This is the absolute group delay, in seconds.

**Hint**  If the group delay data appears noisy, increase the value in the Group delay aperture field. If the results appear inaccurate, decrease the value. Generally, adjusting this value by a factor of 10 (in the appropriate direction) improves noisy or inaccurate results.

For an example of group delay data, refer to “Obtaining Group Delay Data” on page 18-23.
Simulating Linear Noise

Options for simulating linear noise are available from the Noise tab of the S_Param simulation component. For more information about how noise is calculated, refer to “Noise Analysis” on page 6-14.

To simulate linear noise:

1. Proceed as in “Simulating an Amplifier” on page 6-3, setting frequencies and sweep parameters as needed.

2. Edit the S-parameter Simulation component and select the Noise tab. Then select the Calculate noise option.

3. In the Edit field, enter the names of the nodes at which you want noise data to be reported.

   Note: It is not necessary to name nodes if only noise figure is desired.

4. Use the Mode popup menu to sort the noise contributors (nodes) that are reported.

5. Either accept the default values for Dynamic range to display and Bandwidth, or edit these as required. The defaults are generally sufficient.

6. Click OK to accept changes and close the dialog box.

7. Simulate. When the simulation is finished, plot the noise data items. These are noise figure, identified as nf[port_number], and the equivalent input noise temperature, identified as te[port_number].

Adjusting Noise Temperature

The IEEE definition of noise figure states that it should be measured at the standard noise temperature of 290 K (16.85°C). Advanced Design System uses this definition and value of the standard noise temperature in its calculation of noise figure. For a passive circuit, if the simulation temperature is not equal to this value, the noise figure will not be the same as the loss in decibels. The simulation temperature defaults to 25°C. It can be changed by adding an Options item to the design and changing the simulation temperature there to 16.85°C.
Analyzing a Frequency Translating Network

To simulate the effects of frequency translation (also known as frequency conversion) in circuits employing mixers, the S-parameter simulator uses the same algorithm as the AC simulation component. This option causes the simulator to consider the frequency not only of the input fundamental, but also the frequency of the resulting translations. A simple model is used to calculate the reference frequencies at each node.

Selecting the Calculate noise option (under the Noise tab) will result in frequency conversion data for nonlinear noise.

For more conversion information, refer to “S-parameter Frequency Conversion” on page 6-13.

To analyze a frequency translating network:

1. Proceed as in “Simulating an Amplifier” on page 6-3.
2. Insert passive ports at locations where you want to obtain S-parameters.
3. Set frequencies and sweep parameters as needed.
4. Use a large-signal voltage or current source, such as V_1Tone or I_1Tone as the driving signal that causes the frequency translation (not a large-signal port source, such as a P_1Tone).
5. Select the Parameters tab, then select Enable AC frequency conversion.
6. In the field labeled S-parameter freq. conv. port, enter 1.

| Note | The frequency conversion port must be the number of the input port. |

7. To calculate frequency conversion data for nonlinear noise, select the Noise tab and enable Calculate noise.
Eliminating Unwanted Effects

It is sometimes helpful to reduce the contribution of other components in an analysis of a circuit involving, for example, amplifiers. The DC_Block component functions as an open during the DC part of the simulation (which is conducted automatically), while the DC_Feed component functions as an open during the S-parameter simulation. This eliminates the loss that would otherwise be experienced with the Term and the bias resistors in the circuit. Figure 6-2 illustrates the use of the DC_Block component in an example circuit.

Note This design, Amp_wBothMatches.dsn, is in the Examples directory under MW_Ckts/ LNA_prj.

Figure 6-2. DC_Block and DC_Feed Components in a Circuit

To eliminate the effects of port and bias resistances:

1. From the Lumped Elements palette, select the DC_Block and DC_Feed components (as appropriate) and place them in the circuit as follows:
   • Place the DC_Block component (represented by a capacitor) between the ports and the device under test.
   • Place the DC_Feed components (represented by an inductor, not shown in this design) between the pins of the device under test and any bias resistors.
S-parameter Simulation

S-parameter Simulation Overview

S-parameters are used to define the signal-wave response of an n-port electrical element at a given frequency. Detailed discussions of S-parameters can be found in standard textbooks on electrical circuit theory.

Note: You may find it helpful to review the publication S-Parameter Techniques for Faster, More Accurate Network Design (AN 95–1), 5952–1130 (available through the Agilent Web site).

S-parameter simulation is a type of small-signal AC simulation. It is most commonly used to characterize a passive RF component and establish the small-signal characteristics of a device at a specific bias and temperature.

All nonlinear components are linearized and the linear circuit that results is analyzed as a multiport device. Each port is excited in sequence, a linear small-signal simulation is performed, and the response is measured at all ports in the circuit. That response is then converted into S-parameter data, which are in turn sent to the dataset.

Note: By selecting the appropriate option under the Parameters tab, it is possible to convert S-parameter data to Y- and Z-parameters. The S-parameter results are retained.

S-parameter Definitions

The following is a representation of a signal wave in a two-port electrical-element.
where

- $a_1$ is the wave into port 1
- $b_1$ is the wave out of port 1
- $a_2$ is the wave into port 2
- $b_2$ is the wave out of port 2

The S-parameters for this conventional element are defined in standard microwave textbooks as follows:

$$
\begin{align*}
b_1 &= a_1 s_{11} + a_2 s_{12} \\
b_2 &= a_1 s_{21} + a_2 s_{22}
\end{align*}
$$

where

- $s_{11}$ is the port-1 reflection coefficient: $s_{11} = b_1/a_1; a_2 = 0$
- $s_{22}$ is the port-2 reflection coefficient: $s_{22} = b_2/a_2; a_1 = 0$
- $s_{21}$ is the forward transmission coefficient: $s_{21} = b_2/a_1; a_2 = 0$
- $s_{12}$ is the reverse transmission coefficient: $s_{12} = b_1/a_2; a_1 = 0$

These equations can be solved for $b_1$ and $a_1$ in terms of $a_2$ and $b_2$ to yield the transmission (T) parameters as follows:

$$
\begin{align*}
b_1 &= a_2 t_{11} + b_2 t_{12} \\
a_1 &= a_2 t_{21} + b_2 t_{22}
\end{align*}
$$

The T-parameters are related to the S-parameters as follows:

$$
\begin{bmatrix}
t_{11} & t_{12} \\
t_{21} & t_{22}
\end{bmatrix} =
\begin{bmatrix}
s_{12} - s_{11} & s_{22}/s_{21} & s_{11}/s_{21} \\
-s_{22}/s_{21} & 1/s_{21}
\end{bmatrix}
$$

S-parameters are defined with respect to a reference impedance that is typically 50 ohms. For 50-ohm S-parameters—with the 2-port element terminated with 50 ohms at each port—the $s_{21}$ parameter represents the voltage gain of the element from port 1 to port 2.
Group Delay

Group delay is a useful measure of phase distortion in components such as amplifiers and filters. It measures the transit time, with respect to frequency, of a signal through the device under test.

The simulator calculates group delay by performing a finite difference of the phase response to obtain $d\Phi/d\omega$.

$$\text{group\_delay} = \text{delay}(2,1) = -\text{diff}(\text{unwrap}(\text{phaserrad}(S21),\pi))/(2\pi)$$

The simulator sets the frequency aperture to 0.01% of the current frequency. You can override this value by modifying the value in the Group delay aperture field, under the Parameters tab. This function is similar to that found on network analyzers, like the 8710. Refer also to these functions:

- delay
- dev\_lin\_phase
- phaserrad
- unwrap
- diff
- phasedeg
- ripple (and GpDelRip measurement equation)
- volt\_gain

Descriptions are in the Expressions, Measurements, and Simulation Data Processing manual.

Group delay results are considered with respect to the input and output ports only. Results of group delay calculations include delay(2,1) and delay(1,2), which can be viewed in the Data Display. These are absolute group delay, in seconds. For additional results data, add the measurement equations dev\_lin\_phase and GpDelRip to the schematic. Calculations from these equations will also be available in the Data Display.
S-parameter Frequency Conversion

S-parameter simulation normally allows only one frequency to be considered in a noise analysis—that of the source. This can be a disadvantage in obtaining simulation results for circuits involving mixers, which are inherently frequency-translating devices involving multiple frequencies. (Refer to Chapter 11, Harmonic Balance for Mixers.)

As an aid in the simulation of frequencies involving mixer subnetworks, the option Enable AC Frequency Conversion (under the Parameters tab) causes the simulator to consider not only the frequency of the source but also that of one of the mixer’s sidebands (which are defined by the user). Only the upper or lower sideband is considered, not both. The frequency-conversion results will appear in the dataset as for any nodes or probes placed to capture voltage or current data.

The S-parameter simulator uses the same conversion algorithm used by the AC Simulation component. For more information on this algorithm, refer to “Enabling Frequency Conversion” on page 5-6.
Noise Analysis

During an S-parameter simulation, you can calculate these noise characteristics:

- Noise figure
- Noisy 2-port parameters

Each are discussed in the sections that follow.

**Hint**
To aid in noise figure measurements, use the noise circle measurement component, NsCircle or the function ns_circle. Refer to ns_circle in the manual Expressions, Measurements, and Simulation Data Processing.

**Noise Figure**

The parameter nf(k) in the dataset is the noise figure at output port k. You can view results using the Data Display. The noise figure for each port will appear in the variables list as nf(1), nf(2), and so on. nf displays noise figures for all ports.

When noise figure is calculated at a port, the other ports in the network are terminated in their respective impedances.

For a 2-port circuit, the noise figure is the signal-to-noise ratio at the input, divided by the signal-to-noise ratio at the output. It has default units of dB. For a circuit with more than two ports, the noise figure is the ratio of the total noise at the output port to the transmitted input noise. The total noise is the transmitted input noise plus the noise contributed by the network. The transmitted input noise represents the portion of the incident thermal noise (kTB, where k=Boltzmann's constant, 1.381...10^-23 J/K, T=290 K, B=1 Hz) which passes through the system.
Calculating Noise Figure

The common definition of noise factor is signal to noise at the input divided by the signal to noise at the output or

\[ F = \frac{SNR_i}{SNR_o} \]

Another definition is provided by Van Der Ziel:

\[ F = \frac{V_o^2}{\text{contribution of the source to } V_o^2} \]

This definition describes the way that noise is computed in Advanced Design System. This noise of the network without the ports is computed and denoted by \( V_N^2 \).

The port noise is computed separately, and Van Der Ziel’s equation is written as:

\[ F = \frac{KTB|S_{21}|^2 + \frac{V_N^2}{R}}{KTB|S_{21}|^2} \]

If there are multiple ports, the noise figure at output port \( k \) is generalized as:

\[ F = \frac{\sum_{j \neq k}^{N} KT j \Delta F |S_{kj}|^2 + \frac{V_N^2}{R_k}}{\sum_{j \neq k}^{N} KT j \Delta F |S_{kj}|^2} \]

Note that the output port noise is never included in the summation of noise sources. This definition makes it possible to generalize the noise figure calculation to the case of n-port networks, and in the limiting case of \( N=2 \), the calculation agrees with the classic 2-port definition.
Noisy 2-port Parameters

$NF_{\text{min}}$, $R_n$ and $S_{\text{opt}}$ are the noisy two-port parameters. They describe the noise properties of a two-port and how the noise changes with respect to the source impedance. They describe circles of constant noise figure on the Smith chart.

$NF_{\text{min}}$ is the minimum noise figure that the circuit can produce, when the source has the optimum reflection coefficient $S_{\text{opt}}$.

$R_n$ is the noise resistance and controls how fast the noise increases as the source reflection coefficient changes from $S_{\text{opt}}$.

$$F = F_{\text{min}} + \frac{4R_n|\Gamma_{\text{src}} - \Gamma_{\text{opt}}|^2}{(1 - |\Gamma_{\text{src}}|^2)^2(1 + \Gamma_{\text{opt}})^2}$$

where $\Gamma_{\text{opt}} = S_{\text{opt}}$, and $\Gamma_{\text{src}}$ is the reflection coefficient of the source.

Chapter 7: Harmonic Balance Simulation

Harmonic balance is a frequency-domain analysis technique for simulating distortion in nonlinear circuits and systems. It is usually the method of choice for simulating analog RF and microwave problems, since these are most naturally handled in the frequency domain. Harmonic balance simulation obtains frequency-domain voltages and currents, calculating the spectral content of voltages or currents in the circuit.

Use the HB simulation controller component to:

- Compute quantities such as third-order intercept (TOI) points, total harmonic distortion (THD), and intermodulation distortion components
- Perform power amplifier load-pull contour analyses
- Perform nonlinear noise analysis

Refer to the following topics for details on Harmonic Balance simulation:

- “Performing a Harmonic Balance Simulation” on page 7-2 describes the minimum setup requirements for a harmonic balance simulation.
- “Examples” on page 7-3 describes in detail how to set up a basic single-point and a swept harmonic balance simulation, using a power amplifier.
- “Harmonic Balance Overview” on page 7-9 is a brief description of the harmonic balance simulator.
- “Troubleshooting a Simulation” on page 7-18 offers suggestions on how to improve a simulation.
- Chapter 8, Harmonic Balance for Nonlinear Noise Simulation describes how to use the simulator for calculating noise.
- Chapter 9, Harmonic Balance for Oscillator Simulation describes how to use the simulator with oscillator designs.
- Chapter 11, Harmonic Balance for Mixers describes how to use the simulator with mixer designs.

Note: You must have the Harmonic Balance simulator license (included with all Circuit Design suites except RF Designer) to run the following simulations. You can build the examples in this chapter without the license, but you will not be able to simulate them.
Performing a Harmonic Balance Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis:

- Add the HarmonicBalance simulation component to the schematic and double-click to edit it. Fill in the fields under the Freq tab:
  - Enter at least one fundamental frequency and the number (order) of harmonics to be considered in the simulation.
  - Make sure that frequency definitions are established for all of the fundamentals of interest in a design. For example, mixers should include definitions for RF and LO frequencies.
  - If more than one fundamental is entered, set the maximum order. This limits the number of mixing products to be considered in the simulation. For more information on this parameter, refer to “Harmonics and Maximum Order” on page 7-10.

- If you have a circuit with a large number of nonlinear devices or harmonics, the Krylov solver can improve simulation speed. For tips on using this option, refer to “Selecting a Solver” on page 7-15. For the details of the Krylov algorithm, refer to “Harmonics and Maximum Order” on page 7-10.

- You can use previous simulation solutions to speed the simulation process. For more information, refer to “Reusing Simulation Solutions” on page 7-17.

- You can perform budget calculations as part of the simulation. For information on budget analysis, refer to Chapter 18, Using Circuit Simulators for RF System Analysis.

- You can perform small-signal analysis. Enable the Small-signal option and fill in the fields under the Small-Sig tab. For details, refer to Chapter 11, Harmonic Balance for Mixers.
You can perform nonlinear noise analysis. Enable the Nonlinear noise option and fill in the fields under the Noise(1) and Noise(2) tabs. For details, refer to Chapter 8, Harmonic Balance for Nonlinear Noise Simulation.

If your design includes NoiseCon components, select the NoiseCons tab and fill in the fields. For more information, refer to Chapter 8, Harmonic Balance for Nonlinear Noise Simulation.

If your design includes an OscPort component, enable Oscillator and fill in the fields under the Osc tab. Chapter 9, Harmonic Balance for Oscillator Simulation focuses specifically on simulating oscillator designs.

For details about each field, click Help from the dialog box.

Examples

This section gives detailed setups to perform these simulations on a power amplifier:

- “Single Tone Harmonic Balance Simulation” on page 7-4 applies a single tone to the power amplifier. This tone and 7 harmonics are analyzed.

- “Swept Harmonic Balance Simulation” on page 7-6 sweeps the input from 500 to 1500 MHz and analyzes the performance of the amplifier at points along the sweep.
Single Tone Harmonic Balance Simulation

Figure 7-1 illustrates the setup for simulating a power amplifier circuit.

![Figure 7-1. Example setup for a basic harmonic balance simulation](image)

**Note** This design, HB1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in HB1.dds.

1. From the Sources-Freq Domain palette, select and place a V_1Tone source on the schematic. Edit the component to set these values:
   • V = 0.01 V
   • Freq = 20 MHz. This is the first and only fundamental.

2. Ensure that the inputs and outputs of nodes at which you want data to be reported are appropriately labeled. In this example, the output node has been labeled Vout.
3. From the Simulation-HB palette, select and place the HB component on the schematic, then double-click to edit it. Select the Freq tab and edit these parameters:
   - Frequency = 20 MHz
   - Order = 7

Click Add. If this line appears as the second fundamental frequency in the list, select the one above it and click Cut. Make sure that 1 20 MHz 7 is the only line that appears in the list of fundamental frequencies.

**Note** With only a single frequency defined, the parameter Maximum order is not available.

4. Simulate. When the simulation is finished, a Data Display window opens. The following plot illustrates the results of the simulation, showing the fundamental and seven harmonics, declining in voltage (Vout) with increasing frequency.
Swept Harmonic Balance Simulation

In this example, the fundamental is swept from 500 MHz to 1500 MHz in 25 MHz steps. At 41 points along the sweep, data is collected for the value of the fundamental and 8 harmonics.

Figure 7-2 illustrates the setup for a swept harmonic-balance simulation of a power amplifier circuit.
To perform a swept harmonic balance simulation:

1. From **Sources-Freq Domain** palette, select and place a **P_1Tone** component on the schematic and edit it to set these values:
   - Num = 1
   - P = `dbmtow(-10)`. The function `dbmtow()` is used to convert power in dBm to watts for the purpose of simulation.
   - Freq = `freq_swp`. This sets the frequency to a variable which will be defined later.

2. Label the nodes at which you want data to be reported, in this example, label the output node as **Vout**.

3. From the **Simulation-HB** palette, select and place an **HB** simulation component on the schematic and edit it to select the **Freq** tab. Set the following parameters:
   - Frequency = `freq_swp MHz`. This is the only fundamental, or Freq[1].
   - Order = 8
   Click **Add**. If any other fundamentals appear in the list, select each line and click **Cut**. Make sure that 1 `freq_swp MHz 8` is the only line that appears in the list of fundamental frequencies.

Note: Ensure that frequencies are established for all of the frequencies of interest in a design under test (for example, RF, LO, and IF frequencies). You may want to display them on the schematic to facilitate editing.

4. Select the **Sweep** tab. Ensure that **Start/Stop** is selected and set the following:
   - Parameter to sweep = `freq_swp`
   - Sweep Type = **Linear**
   - Start = 500
   - Stop = 1500
Harmonic Balance Simulation

• Step = 25

5. Click OK to accept changes and close the dialog box.

6. From Data Items palette, select and place a VAR (variables and equations) component on the schematic and edit it:
   • Under Variable or Equation Entry Mode, select Name=Value.
   • Under Select Parameter, select the default equation \( X=1.0 \) and in the field to the right, change it to \( \text{freq}_\text{swp}=1 \). This assigns \( \text{freq}_\text{swp} \) to the fundamental.

7. Click OK to accept changes and close the dialog box.

8. Simulate. When the simulation is finished, a Data Display window opens. One way to plot Vout is against the harmonic index. By setting the Trace Type to a spectral display, it shows the strength of each of the eight harmonics as the fundamental changes from 500 MHz to 1500 MHz.
Harmonic Balance Overview

Harmonic balance is a frequency-domain analysis technique for simulating distortion in nonlinear circuits and systems. It is well-suited for simulating analog RF and microwave problems, since these are most naturally handled in the frequency domain. You can analyze power amplifiers, frequency multipliers, mixers, and modulators, under large-signal sinusoidal drive.

Harmonic balance simulation makes possible the multitone simulation of circuits that exhibit intermodulation frequency conversion. This includes frequency conversion between harmonics. Not only can the circuit itself produce harmonics, but each signal source (stimulus) can also produce harmonics or small-signal sidebands. The stimulus can consist of up to twelve nonharmonically related sources. The total number of frequencies in the system is limited only by such practical considerations as memory, swap space, and simulation speed.

The Simulation Process

The harmonic balance method is iterative. It is based on the assumption that for a given sinusoidal excitation there exists a steady-state solution that can be approximated to satisfactory accuracy by means of a finite Fourier series. Consequently, the circuit node voltages take on a set of amplitudes and phases for all frequency components. The currents flowing from nodes into linear elements, including all distributed elements, are calculated by means of a straightforward frequency-domain linear analysis. Currents from nodes into nonlinear elements are calculated in the time-domain. Generalized Fourier analysis is used to transform from the time-domain to the frequency-domain.

A frequency-domain representation of all currents flowing away from all nodes is available. According to Kirchoff's Current Law (KCL), these currents should sum to zero at all nodes. The probability of obtaining this result on the first iteration is extremely small.

Therefore, an error function is formulated by calculating the sum of currents at all nodes. This error function is a measure of the amount by which KCL is violated and is used to adjust the voltage amplitudes and phases. If the method converges (that is, if the error function is driven to a given small value), then the resulting voltage amplitudes and phases approximate the steady-state solution.
Comparing Harmonic Balance and Time Domain Simulators

In the context of high-frequency circuit and system simulation, harmonic balance has a number of advantages over conventional time-domain transient analysis:

- Designers are usually most interested in a system's steady-state behavior. Many high-frequency circuits contain long time constants that require conventional transient methods to integrate over many periods of the lowest-frequency sinusoid to reach steady state. Harmonic balance, on the other hand, captures the steady-state spectral response directly.

- The applied voltage sources are typically multitone sinusoids that may have very narrowly or very widely spaced frequencies. It is not uncommon for the highest frequency present in the response to be many orders of magnitude greater than the lowest frequency. Transient analysis would require an integration over an enormous number of periods of the highest-frequency sinusoid. The time involved in carrying out the integration is prohibitive in many practical cases.

- At high frequencies, many linear models are best represented in the frequency domain. Simulating such elements in the time domain by means of convolution can result in problems related to accuracy, causality, or stability.

Harmonics and Maximum Order

With multiple sources in a circuit, mixing products will occur. The parameter Maximum order (under the Freq tab) determines how many mixing products are to be included in the simulation. Consider an example with two sources and three harmonics:

<table>
<thead>
<tr>
<th>Source</th>
<th>Frequency</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fund 1</td>
<td>5 MHz</td>
<td>3</td>
</tr>
<tr>
<td>Fund 2</td>
<td>7 MHz</td>
<td>3</td>
</tr>
</tbody>
</table>
If Maximum order is 0 or 1, no mixing products are simulated. The frequency list consists of the fundamental (Fund) frequency and the first, second, and third harmonics of each source, as follows:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Hz</td>
<td>DC term</td>
</tr>
<tr>
<td>5 MHz</td>
<td>Fund 1</td>
</tr>
<tr>
<td>7 MHz</td>
<td>Fund 2</td>
</tr>
<tr>
<td>10 MHz</td>
<td>Second harmonic of Fund 1</td>
</tr>
<tr>
<td>14 MHz</td>
<td>Second harmonic of Fund 2</td>
</tr>
<tr>
<td>15 MHz</td>
<td>Third harmonic of Fund 1</td>
</tr>
<tr>
<td>21 MHz</td>
<td>Third harmonic of Fund 2</td>
</tr>
</tbody>
</table>

If Maximum order is 2, the sum and difference frequencies of the two fundamentals are added to the list:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 MHz</td>
<td>Fund 2 – Fund 1</td>
</tr>
<tr>
<td>12 MHz</td>
<td>Fund 2 + Fund 1</td>
</tr>
</tbody>
</table>

If Maximum order is 3, the second harmonic of one source can mix with the fundamental of the other. These frequencies are also added to the list:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 MHz</td>
<td>Second harmonic of Fund 1 – Fund 2</td>
</tr>
<tr>
<td>9 MHz</td>
<td>Second harmonic of Fund 2 – Fund 1</td>
</tr>
<tr>
<td>17 MHz</td>
<td>Second harmonic of Fund 1 + Fund 2</td>
</tr>
<tr>
<td>19 MHz</td>
<td>Second harmonic of Fund 2 + Fund 1</td>
</tr>
</tbody>
</table>

This pattern is also used when there are three sources. The combined order is the sum of the individual signal orders that are added or subtracted to make up the frequency list.
Harmonic Balance Simulation

Consider an example with the following three sources, where Maximum order = 5:

<table>
<thead>
<tr>
<th>Source</th>
<th>Frequency</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fund 1</td>
<td>10.00 GHz</td>
<td>5</td>
</tr>
<tr>
<td>Fund 2</td>
<td>10.95 GHz</td>
<td>2</td>
</tr>
<tr>
<td>Fund 3</td>
<td>11.05 GHz</td>
<td>2</td>
</tr>
</tbody>
</table>

The frequencies that are used in the simulation and the combinations that produce them are listed in Table 7-1.

Table 7-1. Frequency List (Maximum order = 5)

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Combination</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000 Hz</td>
<td>DC term always present</td>
<td></td>
</tr>
<tr>
<td>100.0 MHz</td>
<td>Fund 3 – Fund 2</td>
<td>2</td>
</tr>
<tr>
<td>200.0 MHz</td>
<td>Second harmonic of Fund 3 – Second harmonic of Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>850.0 MHz</td>
<td>Second harmonic of Fund 2 – Fund 1 – Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>950.0 MHz</td>
<td>Fund 2 – Fund 1</td>
<td>2</td>
</tr>
<tr>
<td>1.050 GHz</td>
<td>Fund 3 – Fund 1</td>
<td>2</td>
</tr>
<tr>
<td>1.150 GHz</td>
<td>Second harmonic of Fund 3 – Fund 1 – Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>1.900 GHz</td>
<td>Second harmonic of Fund 2 – Second harmonic of Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>2.000 GHz</td>
<td>Fund 2 + Fund 3 – Second harmonic of Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>2.100 GHz</td>
<td>Second harmonic of Fund 3 – Second harmonic of Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>7.900 GHz</td>
<td>Third harmonic of Fund 1 – Second harmonic of Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>8.000 GHz</td>
<td>Third harmonic of Fund 1 – Fund 2 – Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>8.100 GHz</td>
<td>Third harmonic of Fund 1 – Second harmonic of Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>8.850 GHz</td>
<td>Second harmonic of Fund 1 + Fund 2 – Second harmonic of Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>8.950 GHz</td>
<td>Second harmonic of Fund 3 – Second harmonic of Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>9.050 GHz</td>
<td>Second harmonic of Fund 2 – Second harmonic of Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>9.150 GHz</td>
<td>Second harmonic of Fund 1 + Fund 3 – Second harmonic of Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>9.800 GHz</td>
<td>Fund 1 + Second harmonic of Fund 3 – Second harmonic of Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>9.900 GHz</td>
<td>Fund 1 + Fund 2 – Fund 3</td>
<td>3</td>
</tr>
<tr>
<td>Frequency</td>
<td>Combination</td>
<td>Order</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>10.00 GHz</td>
<td>Fundamental 1</td>
<td></td>
</tr>
<tr>
<td>10.10 GHz</td>
<td>Fund 1 + Fund 3 – Fund 2</td>
<td>3</td>
</tr>
<tr>
<td>10.20 GHz</td>
<td>Fund 1 + Second harmonic of Fund 3 – Second harmonic of Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>10.85 GHz</td>
<td>Second harmonic of Fund 2 – Fund 3</td>
<td>3</td>
</tr>
<tr>
<td>10.95 GHz</td>
<td>Fundamental 2</td>
<td></td>
</tr>
<tr>
<td>11.05 GHz</td>
<td>Fundamental 3</td>
<td></td>
</tr>
<tr>
<td>11.15 GHz</td>
<td>Second harmonic of Fund 3 – Fund 2</td>
<td>3</td>
</tr>
<tr>
<td>11.90 GHz</td>
<td>Second harmonic of Fund 2 – Fund 1</td>
<td>3</td>
</tr>
<tr>
<td>12.00 GHz</td>
<td>Fund 2 + Fund 3 – Fund 1</td>
<td>3</td>
</tr>
<tr>
<td>12.10 GHz</td>
<td>Second harmonic of Fund 3 – Fund 1</td>
<td>3</td>
</tr>
<tr>
<td>12.95 GHz</td>
<td>Second harmonic of Fund 2 + Fund 3 – Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>13.05 GHz</td>
<td>Fund 2 + Second harmonic of Fund 3 – Second harmonic of Fund 1</td>
<td>5</td>
</tr>
<tr>
<td>18.95 GHz</td>
<td>Third harmonic of Fund 1 – Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>19.05 GHz</td>
<td>Third harmonic of Fund 1 – Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>19.90 GHz</td>
<td>Second harmonic of Fund 1 + Fund 2 – Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>20.00 GHz</td>
<td>Second harmonic of Fund 1</td>
<td></td>
</tr>
<tr>
<td>20.10 GHz</td>
<td>Second harmonic of Fund 1 + Fund 3 – Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>20.85 GHz</td>
<td>Fund 1 + Second harmonic of Fund 2 – Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>20.95 GHz</td>
<td>Fund 2 + Fund 1 – 2nd order</td>
<td></td>
</tr>
<tr>
<td>21.05 GHz</td>
<td>Fund 3 + Fund 1 – 2nd order</td>
<td></td>
</tr>
<tr>
<td>21.15 GHz</td>
<td>Fund 1 + Second harmonic of Fund 3 – Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>21.90 GHz</td>
<td>Second harmonic of Fund 2</td>
<td></td>
</tr>
<tr>
<td>22.00 GHz</td>
<td>Fund 2 + Fund 3 – 2nd order</td>
<td></td>
</tr>
<tr>
<td>22.10 GHz</td>
<td>Second harmonic of Fund 3</td>
<td></td>
</tr>
<tr>
<td>22.95 GHz</td>
<td>Fund 3 + Second harmonic of Fund 2 – Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>23.05 GHz</td>
<td>Fund 2 + Second harmonic of Fund 3 – Fund 1</td>
<td>4</td>
</tr>
<tr>
<td>28.95 GHz</td>
<td>Fourth harmonic of Fund 1 – Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>29.05 GHz</td>
<td>Fourth harmonic of Fund 1 – Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>Frequency</td>
<td>Combination</td>
<td>Order</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>29.90 GHz</td>
<td>Third harmonic of Fund 1 + Fund 2 - Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>30.00 GHz</td>
<td>Third harmonic of Fund 1</td>
<td></td>
</tr>
<tr>
<td>30.10 GHz</td>
<td>Third harmonic of Fund 1 + Fund 3 - Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>30.85 GHz</td>
<td>Second harmonic of Fund 1 + Second harmonic of Fund 2 - Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>30.95 GHz</td>
<td>Second harmonic of Fund 1 + Fund 2</td>
<td>3</td>
</tr>
<tr>
<td>31.05 GHz</td>
<td>Second harmonic of Fund 1 + Fund 3</td>
<td>3</td>
</tr>
<tr>
<td>31.15 GHz</td>
<td>Second harmonic of Fund 1 + Second harmonic of Fund 3 - Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>31.90 GHz</td>
<td>Fund 1 + Second harmonic of Fund 2</td>
<td>3</td>
</tr>
<tr>
<td>32.00 GHz</td>
<td>Fund 1 + Fund 2 + Fund 3</td>
<td>3</td>
</tr>
<tr>
<td>32.10 GHz</td>
<td>Fund 1 + Second harmonic of Fund 3</td>
<td>3</td>
</tr>
<tr>
<td>32.95 GHz</td>
<td>Fund 3 + Second harmonic of Fund 2</td>
<td>3</td>
</tr>
<tr>
<td>33.05 GHz</td>
<td>Fund 2 + Second harmonic of Fund 3</td>
<td>3</td>
</tr>
<tr>
<td>34.00 GHz</td>
<td>Second harmonic of Fund 2 + Second harmonic of Fund 3 - Fund 1</td>
<td>5</td>
</tr>
<tr>
<td>40.00 GHz</td>
<td>Fourth harmonic of Fund 1</td>
<td></td>
</tr>
<tr>
<td>40.95 GHz</td>
<td>Third harmonic of Fund 1 + Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>41.05 GHz</td>
<td>Third harmonic of Fund 1 + Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>41.90 GHz</td>
<td>Second harmonic of Fund 1 + Second harmonic of Fund 2</td>
<td>4</td>
</tr>
<tr>
<td>42.00 GHz</td>
<td>Second harmonic of Fund 1 + Fund 2 + Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>42.10 GHz</td>
<td>Second harmonic of Fund 1 + Second harmonic of Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>42.95 GHz</td>
<td>Fund 1 + Second harmonic of Fund 2 + Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>43.05 GHz</td>
<td>Fund 1 + Fund 2 + Second harmonic of Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>44.00 GHz</td>
<td>Second harmonic of Fund 2 + Second harmonic of Fund 3</td>
<td>4</td>
</tr>
<tr>
<td>50.00 GHz</td>
<td>Fifth harmonic of Fund 1</td>
<td></td>
</tr>
<tr>
<td>50.95 GHz</td>
<td>Fourth harmonic of Fund 1 + Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>51.05 GHz</td>
<td>Fourth harmonic of Fund 1 + Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>51.90 GHz</td>
<td>Third harmonic of Fund 1 + Second harmonic of Fund 2</td>
<td>5</td>
</tr>
<tr>
<td>52.00 GHz</td>
<td>Third harmonic of Fund 1 + Fund 2 + Fund 3</td>
<td>5</td>
</tr>
<tr>
<td>52.10 GHz</td>
<td>Third harmonic of Fund 1 + Second harmonic of Fund 3</td>
<td>5</td>
</tr>
</tbody>
</table>
Selecting a Solver

Many harmonic balance simulators rely on the Newton-Raphson technique to solve the nonlinear systems of algebraic equations that arise in large-signal frequency-domain circuit simulation problems. Each iteration of Newton-Raphson requires an inversion of the Jacobian matrix associated with the nonlinear system of equations. When the matrix is factored by direct methods, memory requirements climb as $O(H^2)$, where $H$ is the number of harmonics. Thus, the factorization of a Jacobian at $H=500$ will require 2500 times as much RAM as one at $H=10$.

An alternate approach to solving the linear system of equations associated with the Jacobian is to use a Krylov subspace iterative method such as GMRES (generalized minimum residual). This method does not require the explicit storage of the Jacobian matrix $J$, but rather only the ability to carry out matrix-vector products of the form $J \cdot V$, where $V$ is an arbitrary vector. But the information needed to carry out such an operation can be stored in $O(H)$ memory, not in $O(H^2)$, in the context of harmonic balance. Thus, Krylov subspace solvers offer substantial savings in memory requirements for large harmonic-balance problems. Similar arguments show that even larger increases in computational speed can be obtained.

Note: For circuits involving large numbers of frequencies, consider using the Circuit Envelope simulator.
Harmonic Balance Simulation

Use the following guidelines when selecting a solver:

- **Direct Solver**
  The Direct Solver option is recommended for the majority of small problems. A small problem can be roughly described as one where the circuit contains relatively few nonlinear components, there are one or two fundamental frequencies, relatively few harmonics, etc. In general, in such cases the direct solver is not only faster, but also exhibits superior convergence.

- **Krylov Solver**
  The Krylov Solver option should be used when solving large problems. A large problem can roughly be described as one where memory usage exceeds 100 MB or the memory capacity of the computer (whichever occurs first). A problem may be large because of a large number of nonlinear components, a large number of harmonics required for simulation, or both.

- **Auto Select**
  Selecting this option allows the simulator to choose which solver to use. The simulator analyzes factors such as circuit or spectral complexity and compares memory requirements for each solver against the available computer memory. Based on this analysis it selects either direct solver or Krylov solver in a manner transparent to the user. Furthermore, if the Krylov solver is chosen by the simulator, several options for that solver are also automatically set.

In most cases, the choice between the Direct Solver or Krylov Solver should be fairly easy to make. If uncertain, use Auto Select.

Simulation time may not be a good indicator for the choice. Some problems are small, but still take a long time to simulate because parameters are being swept over many steps; such a problem should really be viewed as a sequence of small problems, and thus Krylov is not necessarily applicable. When a parameter is swept, if it takes $X$ seconds to compute a single solution using the Krylov solver, it will probably take approximately $10X$ seconds to compute 10 solutions. On the other hand, if it takes $Y$ seconds to compute a single solution using the direct solver, it will probably take far less than $10Y$ seconds to compute 10 swept steps of the analysis.

For system-level applications (behavioral mixers, amplifiers, etc.), the Krylov solver should be the preferred method of solution, as it is very robust in this area. For some transistor-level circuits, the Krylov solver may experience convergence difficulties at high input power levels. If this occurs, an analysis using the direct solver or the Envelope simulator should be attempted.
To select a specific solver:

1. Select the **Solver** tab in the Harmonic Balance controller dialog box.
2. Select the desired solver option. In general, we recommend that you accept the defaults and click **OK** to close the dialog box (or select another tab to set additional simulation specifications, as needed).

For descriptions of the options and parameters associated with the solvers, click **Help** from the dialog box.

### Reusing Simulation Solutions

Harmonic balance simulation solutions can be saved and used later as an initial guess for another simulation, including harmonic balance, large-signal S-parameter, gain compression, or circuit envelope. Reusing solutions can save a considerable amount of simulation time. For example, you can save a harmonic balance solution and perform a nonlinear noise simulation, using this saved solution as the initial guess. Doing so removes the time required to re-compute the nonlinear HB solution. Another instance would be to solve for an initial harmonic balance solution and then sweep a parameter to see the changes.

To save a simulation for reuse:

1. Add the desired simulation component to your schematic—harmonic balance, large-signal S-parameter, gain compression, or envelope. Double-click to edit it.
2. Select the **Params** tab. If using the Envelope controller, select the **HB Params** tab.
3. Enable **Write Final Solution**. Enter a filename and any extension, or use the default which is `<project_name>.hbs`. The file will be saved in the networks folder of the project.
4. Click **OK**. When you run the next simulation, the solution will be saved.

To select a solution file to be used as the initial guess:

1. Place the simulation component of interest on the schematic if one is not present, then double-click to edit it.
2. Select the **Params** tab.
3. Enable **Use Initial Guess**. Enter the filename and extension.
4. To view any messages regarding how the initial guess affects the simulation, add an Options component and set the Annotate parameter. For more
Harmonic Balance Simulation

information on how to do this, refer to “Reusing DC Simulation Solutions” on page 1-7.

Note Since harmonic balance simulations also use the DC solution, for optimum speed improvement, both the DC solution and the HB solution should be saved and re-used as initial guesses.

The initial guess file does not need to contain all of the harmonic balance frequencies. For example, you could perform a one-tone simulation with a very nonlinear LO, save the solution, and then use it as an initial guess in a two-tone simulation.

The exact frequencies do not have to match between the present analysis and the initial guess solution. However, the fundamental indexes should match. For example, a solution saved from a two-tone analysis with Freq[1] = 1GHz and Freq[2] = 1kHz would not be a good match for a simulation with Freq[1] = 1kHz and Freq[2] = 1 GHz.

Troubleshooting a Simulation

This section presents suggestions to help achieve a successful simulation that is fast and accurate. It includes the following topics:

• “Selecting the Number of Harmonics” on page 7-19 describes how to select an optimum number of harmonics in a simulation to ensure accuracy and minimize simulation time.

• “Reducing Simulation Time” on page 7-20 describes other methods to help reduce simulation time.

• “Solving Convergence Problems” on page 7-21 offers a variety of methods to try if your simulation is failing because it cannot converge.

• “Oversampling to Prevent Aliasing” on page 7-23 describes how the Oversampling parameter can be set to prevent aliasing during a simulation.

• “Linearizing Nonlinear Devices” on page 7-26 describes how making a circuit more linear can improve the chances of a successful simulation.

Also refer to the troubleshooting sections in Chapter 4, DC Simulation and Chapter 5, Linear AC and Noise Simulation.
Selecting the Number of Harmonics

The number of harmonics needed to simulate a circuit accurately depends on nonlinearities in the circuit, the accuracy desired, and the acceptable simulation time.

Consider the following in choosing or changing the number of harmonics:

- If the simulation has been run with fewer than the minimum number of harmonics that are required for convergence, no solution is found. The simulation may not converge.
- If the simulation has been run with the minimum number of harmonics that are required for convergence, the simulation converges but the results are inaccurate.
- If the simulation has been run with the optimum number of harmonics that are required for convergence, the simulation converges and the results are accurate.
- If the simulation has been run with the more than the optimum number of harmonics that are required for convergence, the simulation converges and the results are accurate, but the simulation takes longer than necessary.

To verify that the simulation is accurate, choose a certain number of harmonics and perform the simulation. Then double the number of harmonics and simulate the circuit again.

If the results are the same, the first choice used enough harmonics for an accurate simulation. If the results differ, double the second choice and run the simulation again. Continue this procedure until there is no change (or there is an acceptable change) in the simulation after the number of harmonics is doubled.
Reducing Simulation Time

To reduce simulation time while still maintaining an accurate simulation, consider the following factors in your simulation and make the appropriate trade-offs:

- **Simulation frequencies**—The number of harmonically unrelated frequencies applied to a network increases the required memory and simulation time considerably.

- **Power levels**—Power levels that increase the degree of nonlinearity will increase simulation time.

- **Harmonics**—The more harmonics to be considered, the greater the memory and simulation time required.

- **Sample points**—Increasing the amount of oversampling may result in a modest increase in simulation time, unless the Krylov option is selected. (Refer to “Oversampling to Prevent Aliasing” on page 7-23).

- **Error tolerances**—The smaller the error tolerances, the greater the simulation time.

- **Device models**—The number of nonlinear elements in a device model and the number of devices used in the circuit will affect both the memory and simulation time required.
Solving Convergence Problems

Nonconvergence is a numerical problem encountered by the harmonic balance simulator when it cannot reach a solution, within a given tolerance, after a given number of numerical iterations. There is no one specific solution for solving convergence problems. However, consider the following guidelines:

- Increase the value of Order (or other harmonic controls); this is the most basic technique for solving convergence problems. However, if the time penalty for doing so becomes severe, other methods should be attempted.

- Convergence criteria are controlled by Voltage relative tolerance, and Current relative tolerance (in the Options component, under the Convergence tab). In general, convergence speed is improved by increasing these values, but at the expense of accuracy. Similarly, the smaller these values are, the more accurate the results but the slower the convergence.

- The Samanskii constant is a real non-negative number that is used to determine whether the simulator can skip the Jacobian evaluation in some iterations, and just reuse an approximate Jacobian instead (Samanskii steps). The Samanskii steps can significantly speed up the solution process. However, using an approximate Jacobian, particularly for a larger number of iterations, may result in poor or even no convergence. The constant is used in two ways. First, it becomes a more absolute measure when it is smaller. It then approaches the requirement that each iteration reduces the relevant norm by one-third. Decreasing the Samanskii constant beyond a certain point (which in turn depends on the quality of the most recent Newton step) will make no difference. However, setting the Samanskii constant to zero will effectively disable any Samanskii steps altogether.

Increasing the Samanskii constant relaxes this requirements in general, but the condition becomes more dependent on the quality of the standard most recent Newton iteration. In other words, a more rapid convergence of the Newton step would also require better convergence of the Samanskii steps.

The default Samanskii constant of 2.0 corresponds to the quality of the Newton step such that the norm is reduced by 50% (decreasing the Samanskii constant will not make any difference if this is the case). Such a cut-off value gets larger with a more significant norm reduction of the Newton step. For example, an 83% norm reduction would result in a cut-off value of 10.
Harmonic Balance Simulation

**Note** To edit this parameter, select the Display tab in the Harmonic Balance Simulation component and set SamanskiConstant to display on the schematic page. You can then edit the parameter directly on the schematic.

- Set the Oversample parameter to a value greater than 1.0, such as 2.0 or 4.0. However, remember that although this can often solve convergence problems, it does so at the cost of computer memory and simulation time. For multiple-tone harmonic balance simulations, make sure that the largest signal in the circuit is assigned to Freq[1]. The simulator’s FFT algorithm is set up so that aliasing errors are much less likely to affect Freq[1] than any other tone. For instance, assign the LO signal in a receiver chain to Freq[1] and the RF signal Freq[2]. For more information, refer to “Oversampling to Prevent Aliasing” on page 7-23.

- Circuits that contain MOSFETS may fail to converge as a result of inherent limitations in the SPICE MOSFET model (Level 1, 2 and 3). This problem can often be circumvented by using the device’s Xqc parameter. Setting Xqc to a nonzero value allows the simulator to use a charge-based model (based upon the work of Ward and Dutton) for the gate capacitance. This often enables the simulator to converge, but at the cost of extracting an extra SPICE model parameter. Also, time-domain simulations usually handle MOSFETs with no difficulty.


- Try sweeping a parameter. Find a circuit element that, when set to some different value, makes the circuit more linear. For instance, in an amplifier circuit there may be a resistor that can be used to lower the amplifier’s gain. The simulator may be able to find a solution to the circuit under a low-gain condition. Then, if the component’s value is swept toward the desired value, the simulator may be able to find a final solution. Start with a value that works, and stop with the desired value. Also, select Restart, under the Params tab. Usually, a better initial guess at each step helps the simulator to converge.

7-22 Troubleshooting a Simulation
• Harmonic balance simulation uses the parameters Current relative tolerance, Voltage relative tolerance, Current absolute tolerance, and Voltage absolute tolerance in the same way as DC simulation does. Refer to “Adjusting Tolerance Parameters” on page 4-16 for more information on this topic.

• Do not use more harmonics than you really need. Too many harmonics can introduce so many unknowns to the solution that the simulator cannot find an answer (because the Jacobian becomes ill-conditioned). Removing harmonics can make the circuit easier (and faster) to solve, but if too many harmonics are removed, Fourier series truncation errors will become important. The simulator will converge on an answer, but it will not be accurate. Refer to “Selecting the Number of Harmonics” on page 7-19 for instructions on how to minimize the number of harmonics.

• If the simulator cannot find a solution, it will set all large-signal AC sources to zero (leaving DC sources alone) and attempt to sweep their value slowly up to the desired level, using the results of each simulation as the initial guess for the next. Very rarely, circuits can exhibit a bizarre behavior that can cause this technique to fail.

• If the harmonic balance simulation cannot converge on a solution within a given number of iterations, try increasing the limit in the Max. Iterations field, under the Params tab.

Oversampling to Prevent Aliasing

Oversampling increases the accuracy of the solution by reducing the FFT aliasing error. In addition, oversampling sometimes improves convergence.

Aliasing error is introduced during the evaluation of nonlinear devices when a time waveform is converted into a spectrum. Aliasing is caused by using too few time-points to represent the time waveform. The Nyquist criterion states that to avoid aliasing errors, the sampling frequency should be at least twice the highest frequency present in the time waveform. If not, error is introduced into the high-frequency Fourier coefficients. The only way to eliminate aliasing error is to increase the sampling rate (that is, to increase the number of data points). This can be done by oversampling the time waveform (select Oversample, under the Params
The relationship between Order, Oversample, and the number of samples is as follows:

<table>
<thead>
<tr>
<th>Order</th>
<th>Oversample</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>64</td>
</tr>
</tbody>
</table>

To calculate the FFT size for a single-tone simulation, the system adds 1 to the user-specified order (to account for DC), multiplies this number by the Oversample value, and then rounds this value up to the next power of two. See the table below.

<table>
<thead>
<tr>
<th>Order</th>
<th>Oversampling</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>64</td>
</tr>
</tbody>
</table>

Note that for the same value of Order, different values of Oversample can yield the same number of samples. Also note that oversampling can occur even with an Oversample value of 1, because the system rounds the FFT size up to the next power of two.

If you click More to the right of Oversample, a Fundamental Oversample dialog box appears. This box allows you to set a separate oversampling for each fundamental.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>256</td>
</tr>
</tbody>
</table>
Note that in the n-tone case, the number of FFT samples grows roughly by the value of Oversample.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>512</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1024</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2</td>
<td>256</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3</td>
<td>1024</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>1024</td>
</tr>
</tbody>
</table>
Linearizing Nonlinear Devices

The number of nonlinear devices in a network has a dramatic effect on the speed of harmonic balance simulation. For circuits with a large number of nonlinear devices, such as those found in RFICs and high-speed digital applications, a standard harmonic balance simulation may become extremely slow, especially for a multitone simulation or where large numbers of harmonics are involved. However, usually only a few devices in such circuits are driven into nonlinearity. Depending upon the application, many devices in the circuit may be operating in their linear region.

A significant increase in the speed of a harmonic balance simulation can be achieved without loss of accuracy if the devices that are operating in the linear region can be identified and linearized about their DC operating point. When these devices are simulated, they are treated as other linear elements and are evaluated in the frequency domain instead of the time domain. The increase in simulation speed is generally proportional to the number of valid device linearizations.

You may want to try the Krylov solver to improve simulation of nonlinear circuits. For more information, refer to “Selecting a Solver” on page 7-15.
Chapter 8: Harmonic Balance for Nonlinear Noise Simulation

The nonlinear noise options in the Harmonic Balance simulator enable you to calculate:

- Nonlinear spot noise
- Swept noise
- Noisy 2-port parameters

There are two methods of noise simulation. One is to use the parameters on the Noise tabs in the harmonic balance dialog box. A second method is to use the NoiseCon component. Using NoiseCon components, you can set up several noise simulations, thereby eliminating the need to change the values on the Noise tabs in the Harmonic Balance dialog box. With NoiseCons, you can also set parameters to calculate phase noise.

If you are not familiar with the harmonic balance simulator, refer to Chapter 7, Harmonic Balance Simulation, before continuing with this chapter.

Refer to the following topics for details on harmonic balance for nonlinear noise simulation:

- “Performing a Nonlinear Noise Simulation” on page 8-2 describes the minimum setup for calculating noise.
- “Performing a Noise Simulation with NoiseCons” on page 8-4 describes how to set up NoiseCons and include them in the simulation.
- “Nonlinear Noise Overview” on page 8-5 is an overview on how nonlinear noise is calculated in a simulation.
- “NoiseCon Overview” on page 8-6 provides more details about the NoiseCon component.
- Chapter 10, Oscillator Noise Simulation, focuses on setting up noise simulations for oscillators.
- Chapter 12, Mixer Noise Simulation, focuses on noise simulations for mixers.
Performing a Nonlinear Noise Simulation

Use this type of nonlinear noise simulation for spectral noise simulations of circuits that do not employ mixers, such as amplifiers.

For a successful analysis:

- Terminate all inputs and outputs with sources or terms. Use sources such as the P_1Tone component in the Sources-Freq Domain palette. Terminate outputs with the Term component, which can be found on several simulation palettes, including HB. Noise parameters are meaningful only with respect to two ports.
- Place a noise source where noise is to be injected and edit the component as required. Noise sources are found under the Sources-Noise palette.

**Note** The built-in BJT model (BJT_Model) has 1/f noise sources that can be turned on by setting certain model parameters.

**Note** When simulating noise figure, noise sources should not be added at the input. Noise sources can be added elsewhere in the circuit, as desired.

- From the Simulation-HB palette, select and place the HarmonicBalance component, and double-click to edit it. Enable Nonlinear noise at the bottom of the dialog box.
- Select the Noise(1) tab:
  - To calculate spot noise, set the Sweep Type to Single point and enter a frequency. If an input frequency is not specified, it is assumed to be the same as the spot noise frequency. This way, the input and output frequencies are the same.
  - For a swept noise analysis, set Sweep Type to Linear or Log and define the sweep with Start/ Stop or Center/ Span values.
  - In the Input frequency field, enter a frequency for the sideband that will mix to the spot-noise frequency of interest.
  - In the Noise input port field, enter the number of the port at which noise is injected.
• In the Noise output port field, enter the number of the Term component at which noise is retrieved.

• Select the Noise (2) tab:
  • To specify the nodes at which noise will be computed, select a node name from the Edit list and click Add.
  • The Noise contributors fields enable you to specify and sort the items that contribute noise to the simulation, and specify a threshold that determines whether contributors are reported or not.
  • You can calculate noisy 2-port parameters such as minimum noise figure, optimum source match, and effective noise resistance. These results will appear in the dataset as NFmin, Sopt, and Rn, respectively.

For more information about each field, click Help from the HarmonicBalance dialog box.

---

**Note**  Once you have entered these settings, you can switch the noise simulation off by disabling Nonlinear noise at the bottom of the dialog. Your settings will remain in the Noise tabs and become active when Nonlinear noise is enabled again.

---

• Add an Options component to the schematic and set the temperature either by editing Temp=16.85 in the Schematic window, or by selecting the Misc tab and editing Simulation temperature to that value. This temperature, 16.85°C or 290 Kelvin, is the standard temperature for noise figure measurement as defined by the IEEE definition for noise figuration.

For more information about nonlinear noise analysis, refer to "Nonlinear Noise Overview" on page 8-5.
Performing a Noise Simulation with NoiseCons

This section describes using the NoiseCon component to enable a noise simulation.

1. Select NoiseCon from the Simulation-HB palette and place it in the Schematic window. Edit it to select the Freq tab and enter noise frequency information.

2. Select the Nodes tab and enter information about the circuit nodes at which to compute noise.

3. Select the Misc tab and enter information for noise figure, noise contribution reports and noise bandwidth.

4. Select the PhaseNoise tab and enter information about phase noise simulation. Click OK.

5. Edit the Harmonic Balance simulation control item to select the NoiseCons tab and add the newly placed NoiseCon component to the list of NoiseCons to be simulated. It is not necessary to enable the Nonlinear Noise option at the bottom of the dialog box. Click OK.

When the simulation is performed, the noise simulation requested in the NoiseCon item will be simulated as well.

For more information about NoiseCons, refer to “NoiseCon Overview” on page 8-6.
Nonlinear Noise Overview

The results of the harmonic balance simulation are output to the dataset and are then used to determine the periodic operating point for the nonlinear noise simulation. The periodic operating point is the steady-state voltages and currents within the circuit at the fundamental, harmonic, and all mixing frequencies. Every upper and lower noise sideband is modeled for each large-signal spectral component; consequently, the number of noise frequencies simulated is double the harmonic frequencies and the sparse-matrix memory requirement is quadrupled (unless the Krylov option is used). The result is that a nonlinear noise simulation requires four times the memory of a normal harmonic balance simulation. Use low values for the parameter Maximum order (under the Freq tab) to limit the demands on computer memory.

**Note** In nonlinear noise analyses, we recommend that the Options component be used to establish a global simulation temperature of 16.85°C if a noise calculation is to be performed. This can be done by editing Temp=16.85 in the Schematic window, or by selecting the Misc tab and editing Simulation temperature to that value. This temperature, 16.85°C or 290 Kelvin, is the standard temperature for noise figure measurement as defined by the IEEE definition for noise figure.

If computer memory is insufficient for a noise simulation, eliminate the large-signal tone at the input and replace it with a Term component. Also, reduce the number of tones in the harmonic balance simulation component by 1. The results of the noise figure simulation should not change significantly. To view an example of how to simulate a mixer’s output noise without a large-signal RF input tone, look at examples/RFIC/Mixers_prj/NoiseFloor.

If the noise figure is needed, add ports to the circuit. In addition, because the single-sideband definition of noise figure is used, the correct input sideband frequency must be specified, by the parameter Input frequency (under the Noise tab). This identifies which input frequency will mix to the spot-noise frequency of interest.

If the input frequency is not specified, it is assumed to be the same as the spot-noise frequency. This way, the input and output frequencies are the same, as is typical of amplifiers. Input frequency is the frequency at which the signal enters the mixer’s RF port. For mixers, input frequency is typically determined by an equation that involves the local oscillator frequency and the noise frequency. Either the sum of or difference between these values is used, depending on whether upconversion or downconversion
Harmonic Balance for Nonlinear Noise Simulation

is taking place. If an upper or lower sideband cannot be found that agrees with the specified input frequency, a warning is issued and the nearest sideband is used.

If noisy 2-port parameters such as $S_{opt}$, $R_n$, or $NF_{min}$ are required, select Calculate noisy two-port parameters. Noise data will be computed only for those nodes requested. The noise voltages and currents scale with the square root of the noise parameter Bandwidth. The default bandwidth is 1 Hz, so that the results have units of $V/\sqrt{\text{Hz}}$. The option Include port noise tells the simulator to include the contributions of port noise in the analysis of noise voltages and currents. The option Use small-signal frequencies (on by default) gives more accurate results, but also requires more memory.

**NoiseCon Overview**

The NoiseCon nonlinear noise controller allows more flexibility in the simulation of noise than can be specified using the Noise(1) and Noise(2) tabs of the Harmonic Balance dialog box. The NoiseCon allows the computation of many noise-related quantities, including noise voltages and currents, differential noise voltages, noise figure and phase noise around any large-signal carrier. Multiple NoiseCons can be placed and simulated in one harmonic balance simulation.

Everything that can be specified in the Noise(1) and Noise(2) tabs can be specified in a NoiseCon. If you use NoiseCons to perform the noise simulation in harmonic balance, then you do not need to use the Noise(1) and Noise(2) tabs, nor do you need to select the Nonlinear Noise option (at the bottom of the Harmonic Balance dialog box).

In the dataset, noise results from the Noise(1) tab appear in the dataset with a name of the form HB1.HB_NOISE.vout.noise, where HB1 is the instance name of the harmonic balance control item. Noise results from a NoiseCon will appear in the dataset with a name of the form HB1.NC1.vout.noise, where NC1 is the instance name of the NoiseCon.
Chapter 9: Harmonic Balance for Oscillator Simulation

The Harmonic Balance simulation component includes features specifically designed for simulating oscillators. You can obtain results such as:

- Oscillator frequency and spectrum
- Small-signal loop gain
- Large-signal loop gain
- Phase noise
- VCO tuning

If you are not familiar with the harmonic balance simulator, refer to Chapter 7, Harmonic Balance Simulation before continuing with this chapter.

Refer to the following topics for details on harmonic balance for oscillator simulation:

- “Performing an Oscillator Simulation” on page 9-2 has the minimum setup requirements for simulating an oscillator.
- “Examples” on page 9-3 describes in detail how to set up simulations for many items of interest in an oscillator, such as finding the oscillation frequency, steady-state conditions, and loop gain.
- “Oscillator Simulation Overview” on page 9-21 is a brief description of the oscillator simulation process.
- “Troubleshooting a Simulation” on page 9-25 offers suggestions on how to improve a simulation.
- Chapter 10, Oscillator Noise Simulation focuses on noise simulations for oscillators.

Note The Krylov Solver is not supported for oscillator analyses.
Performing an Oscillator Simulation

You can obtain certain oscillator characteristics using simulators other than the Harmonic Balance simulator:

- A DC simulation can be run separately to find the device's DC operating point.
- A linear AC simulation can be used to find $f_0$, the frequency of oscillation.
- An S-parameter simulation can be used to find $S_{21}$, and therefore oscillator gain and $f_0$.

For a successful analysis:

- Make sure your oscillator circuit has sufficient feedback gain to permit oscillation. To check this, run an S-parameter simulation on the oscillator before attempting the harmonic balance simulation. A brief overview of how to set this up is in the example “Finding the Frequency of Oscillation” on page 9-4. For more details on how to set up S-parameter simulations, refer to Chapter 6, S-parameter Simulation.
- Add an Oscillator probe component to your design:
  - OscPort is a grounded oscillator port (essentially a probe) and can be found on the Simulation-HB palette. This component calculates the oscillator waveform and amplitude.
  - OscPort2 can calculate waveform and amplitude, small-signal loop gain, or large-signal loop gain. Use this with circuits that have differential oscillator topologies. This component is on the Simulation-HB palette.
  - OscTest calculates oscillator loop gain. Use it in circuits where you would use the OscPort. This component has a simulation process already included in it, so you do not set up a harmonic balance component. This component is on the Probe Components palette.

The placement and setup of these components vary. For instructions on how to do this, place one in your schematic, double-click the component, click the Help button in the dialog box and follow the instructions that appear.

**Note** Do not use more than one oscillator probe component in a circuit.
• Place a HarmonicBalance simulation component (except if using the OscTest component) on the schematic and double-click to edit it. Enable Oscillator at the bottom of the dialog. Select the Osc tab and enter the name of the oscillator probe component that you added to the circuit.

• Select the Freq tab and enter the oscillation frequency as the fundamental. Enter the number of harmonics of interest in the Order field.

For details about each field, click Help from the Harmonic Balance dialog box.

Examples

This section gives detailed setups for:

• “Finding the Frequency of Oscillation” on page 9-4
• “Calculating Large-Signal, Steady-State Oscillation Conditions” on page 9-6
• “Calculating Oscillator Loop Gain” on page 9-11
• “Using OscPort2 for Oscillator Analysis” on page 9-14

Note  You must have the Harmonic Balance simulator license (included with all Circuit Design suites except RF Designer) to run the following simulations. You can build the examples without the appropriate licenses, but you will not be able to simulate them.
Harmonic Balance for Oscillator Simulation

**Finding the Frequency of Oscillation**

Fundamental to any oscillator analysis is establishing the frequency of oscillation. The simplest way to do this is to use the S-parameter simulator. Figure 9-1 illustrates how to use the S-parameter simulator and Term components to find the resonant frequency. This simulation just gives a rough estimate of the frequency of oscillation.

If you are not familiar with S-parameter simulations, refer first to Chapter 6, S-parameter Simulation.

---

**Note**  This design, OscFeedbackTest.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in OscFeedbackTest.dds.

---

![S-Parameters Diagram](image)

**Figure 9-1. Example Setup for Finding an Oscillator's Resonant Frequency**

To find the frequency of oscillation:

1. Place Term components at the input and output of the resonant circuit, and run the S-parameter simulation.
2. Open a Data Display window and plot $S(2,1)$. Place a marker where the curve shows the peak response:

![Diagram showing a peak at m1 with frequency of 102 MHz and m1 value of -3.638118]

3. Note the frequency of oscillation, in this example, approximately 102 MHz. This frequency will be used in the next simulation example.
Calculating Large-Signal, Steady-State Oscillation Conditions

Figure 9-2 illustrates a setup that uses the Harmonic Balance simulator and OscPort component.

For oscillators with a differential (or balanced) topology, the OscPort2 component in automatic mode should be used instead of OscPort. For an example using OscPort2, refer to “Using OscPort2 for Oscillator Analysis” on page 9-14.

Note  This design, OSC2.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in OSC2.dds.

This oscillator circuit consists of two subnetworks: an active gain stage and a frequency-determining feedback stage.

Figure 9-2. Example setup for a basic oscillator test using the harmonic balance simulator and the OscPort component
Figure 9-3 depicts the active gain stage.

![Figure 9-3. Active gain subnetwork](image)

Figure 9-4 depicts the resonant feedback stage.

![Figure 9-4. Resonator subnetwork](image)

To determine large-signal, steady-state oscillation conditions:

1. From the *Simulation-HB* palette, select an *OscPort* grounded oscillator port (which is essentially a probe) and place it in the circuit so that it breaks the feedback loop.

   - In a feedback oscillator, use this port to break the feedback path. The arrow (and pin 2) should point in the direction of injection (the direction of the gain). Refer to Figure 9-2.

   - In a reflection oscillator, place the port so that it separates the negative-resistance portion of the oscillator from the resonator portion. The arrow (and pin 2) should point in the direction of the negative resistance.
Harmonic Balance for Oscillator Simulation

Note A circuit should contain only one oscillator port.

2. From the Simulation-HB palette, select the HarmonicBalance simulation component and place it on the schematic. Edit the component, scroll to the Osc tab and select it.
   • At the bottom of the dialog box, select Oscillator.
   • Enter the name of the OscPort item (in this example, oscport1, in the OscPort name field. This will report the values the simulator finds at this component. The circuit simulator is capable of automatically finding the only OscPort or OscPort2 in the circuit if the name is entered as “yes”. This is required if the OscPort2 component is used.

3. Scroll to the Freq tab and enter a frequency for the fundamental (Freq[1]), if you know it (in this example, 102 MHz). The oscillator will use this value and the values for the OscPort parameters NumOctaves and Steps to search a range of frequencies until the small-signal oscillation conditions are satisfied.

4. Set Order[1], the order of the fundamental, to 8.

5. Label nodes at which you want data to be reported. In this example, Output indicates the output of the gain stage.
6. **Simulate.** When the simulation is complete, open a Data Display window and do the following:

Plot the power relationship of the tones:

- Select the Rectangular plot option and select **Output** as the parameter to plot. Accept **dB** as the default plot type.
- After adding **Output** to the **Traces** field, select it and click **Trace Options**.
- Click **Trace Expression**, and enter the following equation:
  
  \[ \text{dBm(OSET2..Output,1000)} \]

This provides a value in dBm as referenced to the 1-kohm resistor in the gain stage (the default would otherwise be 50 ohms).

Click **OK**. The following appears, showing the power relationship of the eight tones (Order = 8), plotted as the harmonic index (harmindex):
Plot voltage output of the gain stage over time:

- Still in the Data Display window, select an Equation plot and enter the following equation:

\[ \text{time\_wave} = \text{ts(Output)} \]

Plot the equation \text{time\_wave} to yield the following:
Calculating Oscillator Loop Gain

To calculate oscillator loop gain, use the OscTest component as described here. Because the OscTest component incorporates a simulation component into itself, no HarmonicBalance simulation component is required.

For oscillators with a differential topology, use the OscPort2 component for calculating loop gain. For an example, refer to “Using OscPort2 for Oscillator Analysis” on page 9-14.

Figure 9-5 illustrates a setup using the OscTest component.

---

Note This design, OSC1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in OSC1.dds.

---

Figure 9-5. Example Setup for Using the OscTest Component

To determine oscillator loop gain:

1. From the **Probe Components** palette, select the **OscTest** grounded oscillator port and place it in the circuit so that it breaks the feedback loop:

   - In a feedback oscillator, use the OscTest component to break the feedback path. The arrow (and pin 2) should point in the direction of injection (the direction of the gain). Refer to Figure 9-5.

   - In a reflection oscillator, place the component so that it separates the negative resistance portion of the oscillator from the resonator portion. The arrow (and pin 2) should point in the direction of the negative resistance.
2. Double-click the OscTest component to edit it. Edit values for Start and Stop frequencies, as well as Points (the number of frequency points to be considered). In this example:
   • Start = 90 MHz
   • Stop = 110 MHz
   • Points = 101

3. **Simulate.** When the simulation is complete, open a Data Display window and:
   • Select a Polar plot, plot S(1,1), and place a marker on the result, as follows.
   • Plot both the magnitude and phase of S(1,1), and place a marker directly over the zero crossing of the phase plot (shown next)
As required, the gain is greater than unity at the resonant frequency.

**Hint**  Sometimes the simulated loop gain obtained by this procedure is sensitive to the OscTest impedance, $Z$. If the small-signal loop-gain conditions cannot be satisfied, try running the simulation with $Z$ as a swept parameter.
Using OscPort2 for Oscillator Analysis

The OscPort2 component combines the features of the OscPort and OscTest components and adds the abilities to examine large signal loop gain and to analyze oscillators with a differential topology. The OscPort2 can be used in the harmonic balance simulation of an oscillator to compute its large signal solution and frequency, like OscPort. It can be used to examine the small signal loop gain of an oscillator over a range of frequencies, like OscTest. It adds the ability to examine the large signal loop gain over a range of frequencies and injected signal levels.

The OscPort2 component provides the ability to easily handle oscillators with a differential, or balanced, circuit topology, which are commonly found in integrated circuits. OscPort2 can also be used to analyze a single-ended oscillator by grounding the negative input and output of the OscPort2.

Connecting OscPort2 in a Differential Oscillator

Figure 9-6 is the schematic of a differential ring oscillator. It is made of four differential amplifiers with differential inputs and outputs. Unlike a single-ended ring oscillator, a differential ring oscillator will oscillate with an even number of stages, provided that one of the connections between stages crosses over, connecting a positive output to a negative input and vice-versa.

Note This design, hbosc.dsn, is in the Examples directory under Tutorial/OscPort2_prj. The results are in hbosc.dds.

OscPort2 is available in both the Simulation-HB and Probe Components palettes. It should be inserted in a differential oscillator so that it intercepts both the positive and negative signals in the feedback loop. The arrow should point in the direction of loop gain or signal flow.
Calculating Oscillator Frequency and Harmonics

An OscPort2 in Automatic oscillator mode is used in a harmonic balance to simulate an oscillator and determine both its large signal oscillation spectrum as well as its actual frequency of oscillation. The Mode of the OscPort2 should be set to Automatic. The FreqPlan and VinjPlan are not used.

The simulation results for the oscillator are shown below. The frequency of oscillation was found to be 133.7 MHz. Both the differential output, from node vplus to vminus, and a single-ended output, from node vplus, are shown. As expected, the differential signal contains only odd-order harmonics.

![Graph showing differential and single-ended output frequencies](image)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Differential Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-100</td>
</tr>
<tr>
<td>0.2</td>
<td>-40</td>
</tr>
<tr>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>0.6</td>
<td>-40</td>
</tr>
<tr>
<td>0.8</td>
<td>-100</td>
</tr>
<tr>
<td>1.0</td>
<td>-40</td>
</tr>
<tr>
<td>1.2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Single-Ended Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-120</td>
</tr>
<tr>
<td>0.2</td>
<td>-80</td>
</tr>
<tr>
<td>0.4</td>
<td>-40</td>
</tr>
<tr>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.8</td>
<td>-40</td>
</tr>
<tr>
<td>1.0</td>
<td>-80</td>
</tr>
<tr>
<td>1.2</td>
<td>-120</td>
</tr>
</tbody>
</table>

Oscillation frequency: 133.7 MHz
Calculating Large-Signal Loop Gain

OscPort2 can be used to simulate the large-signal loop gain of an oscillator using harmonic balance. The equivalent circuit of the OscPort2 in Large signal loop gain mode is shown below. The OscPort2 is used to break the oscillator feedback loop at the fundamental frequency only. The loop remains connected at DC and the other harmonics. A signal, LoopIn, is injected into the oscillator and the resulting signal that comes back around the loop, LoopOut, is obtained. The complex loop gain is computed using LoopOut/LoopIn and is placed in the dataset with the name LoopGain. The user can sweep both the injected signal frequency as well as its voltage. The circuit will oscillate at the injected frequency and voltage where the loop gain is exactly 1+j0.

Note This design, loopgain_large.dsn, is in the Examples directory under Tutorial/OscPort2_prj. The results are in loopgain_large.dds

![Diagram of OscPort2 in Large-Signal Loop Gain Mode]

Figure 9-7. Equivalent circuit of OscPort2 in large-signal loop gain mode
Set the Mode parameter (of OscPort2) Large signal loop gain. Because the control items are built into OscPort2, no analysis controller is necessary, when using OscPort2 in this mode. The names of the appropriate SweepPlans should be set for FreqPlan and VinjPlan. For this oscillator, the injected frequency is swept from 130 MHz to 138 MHz using the SweepPlan named SweepFreq. The injected voltage level is swept from 0.01 V to 1.0 V using the SweepPlan SweepVinj.

The simulation results are shown above. In the dataset, the result named LoopGain is the complex loop gain for the oscillator. One typical way to present this information is to plot the loop gain phase against the loop gain magnitude in dB. The point of oscillation will then be at the 0,0 point on the graph. The lines on the plot are lines of constant frequency, plotted with increasing injected voltage. On these plots, the injected voltage increases from right to left, showing the gain compression of the oscillator as the injected signal increases.
Calculating Small-Signal Loop Gain

OscPort2 can be used to simulate the small-signal loop gain of an oscillator using small-signal AC analysis (like the OscTest component allows). The equivalent circuit of OscPort2 in Small-signal loop gain mode is shown below. The OscPort2 is used to break the oscillator feedback loop at the small signal frequency only. The loop remains connected at DC. A signal, LoopIn, is injected into the oscillator and the resulting signal that comes back around the loop, LoopOut, is obtained. The complex loop gain is computed using LoopOut/LoopIn and is placed in the dataset with the name LoopGain. The user can sweep the injected signal frequency. The circuit will oscillate near the injected frequency where the loop gain is exactly 1+j0.

Note  This design, loopgain_small.dsn, is in the Examples directory under Tutorial/OscPort2_prj. The results are in loopgain_small.dds

Figure 9-9. Equivalent circuit of OscPort2 in small-signal loop gain mode.
Examples 9-19

Figure 9-10. Simulation setup to simulate the small-signal loop gain of an oscillator.

Set the Mode parameter (of OscPort2) to Small-signal loop gain. Because the control items are built into OscPort2, no analysis controller is necessary (when using OscPort2 in this mode). The name of the frequency SweepPlan should be set for FreqPlan. For this oscillator, the injected frequency is swept from 15 MHz to 25 MHz using the SweepPlan named Plan1.
The simulation results are shown here. In the dataset, the result named LoopGain is the complex loop gain for the oscillator. The magnitude and phase of the loop gain are plotted and shown above. This oscillator will oscillate near 20 MHz.
Oscillator Simulation Overview

Oscillators are nonlinear by nature, using positive feedback to achieve oscillation. The loop gain must be greater than one with a phase shift of zero. Once it has been placed in the appropriate point in the circuit, the OscPort (oscillator port) component checks for this condition. When the loop-gain calculations are being performed, the oscillator port becomes a short circuit to all harmonic frequencies and a unidirectional isolator to the fundamental. During these calculations, the port effectively turns the closed-loop oscillator into an open-loop amplifier.

Although a loop gain greater than one is required for oscillation to begin, at some level the amplitude of oscillation is limited. This limit may come about through the saturation of some active device, or may be the result of an active gain control (AGC) network. Either way, the limiting action requires that the gain of the loop be a function of the amplitude of the signal. This requires a nonlinear analysis. The simulator uses an iterative process to try to find a set of nonzero steady-state solutions.

When the oscillator simulation is launched, a harmonic-balance search algorithm varies the operating frequency to find the oscillation frequency, as well as the final operating frequency and power levels, of the oscillator circuit.

Note  Power levels are computed by means of a reference impedance value. However, because this value typically does not represent your circuit's impedance, the power levels may not be physically meaningful. For example, in a high-impedance oscillator, the simulation power levels may be much greater than the power the oscillator can actually deliver to its load.

The simulator performs a fully automated search for the oscillator's operating characteristics. The automated search involves:

- Searching for a frequency where the circuit satisfies the linear oscillation condition.
- Searching for a frequency and power such that the oscillator's open-loop gain is at unity magnitude and zero phase angle (within a given tolerance).
- A full closed-loop harmonic balance analysis in which frequency and oscillator power are further refined to a more exact solution.
Harmonic Balance for Oscillator Simulation

Performing VCO Tuning

VCO tuning is accomplished by performing an oscillator simulation in conjunction with a swept variable analysis (refer to Chapter 19, Parameter Sweeps and Sweep Plans). The simulator performs a separate oscillator analysis for each swept variable value, then attempts to find the actual oscillation condition for each value. After the analysis is complete, you can display the frequency of oscillation and the output power of that frequency as a function of, for example, the swept variable.

You should not sweep a variable in large steps, because that may snap the algorithm out of the vicinity of the prior oscillation condition, resulting in a convergence failure.

Measuring Oscillator Loop Gain

A quick and efficient way to ensure that the network meets the necessary conditions for oscillation is to use the OscTest component to verify whether the linear oscillation conditions are satisfied. OscTest performs a small-signal loop gain analysis on the oscillator. It does the initial analysis that is performed by the OscPort component, and it acts like a signal source, load, and ideal circulator. The impedance of the source and load are the same, and are set via the Z-parameter on the OscTest component. For some oscillators, the loop gain and phase shift vary with this Z-parameter, so it is sometimes a good idea to check the loop gain and phase shift versus frequency for different values of the Z-parameter. The results of a simulation with an OscTest component are the small-signal open loop gain and phase shift, as a function of frequency. Generally, the circuit will oscillate if the open loop gain satisfies

\[ |G_l| > 1, \quad I_m(G_l) = 0 \]

or equivalently, where the output impedance (Z_{out}) satisfies

\[ Re(Z_{out}) < 0, \quad I_m(Z_{out}) = 0 \]

For a more rigorous description of the oscillation conditions that must be satisfied, refer to the oscillator component descriptions in the Circuit Components manual.

Note: Because the control items are built into OscPort, no analysis controller is necessary.
In addition, an example of this type of analysis can be found in the example file /examples/Tutorial/LearnOsc_pr/OscTest_VCO. The application note, PN 85150-4 Simulating Noise in Nonlinear Circuits Using MDS/RFDS, provides further information about determining the expected frequency of oscillation for an oscillator, including a description of a method that does not use the OscTest component. This application note is accessible from the Agilent EEsof website.

After performing the simulations with OscTest and finding the potential frequency of oscillation, OscTest can be replaced with an OscPort and oscillator harmonic balance simulations can then be run.

It is important to note that an OscTest analysis is not an accurate way to determine the oscillation frequency, nor does it help determine the operating power levels of the oscillator. During oscillation, reactances within the device change considerably, causing frequency shifts not predicted by the linear simulation.

A large-signal nonlinear analysis accounts for the variations in device reactances and provides a way to determine the oscillation frequency with far greater accuracy. It also provides information on the output power level of the oscillator, output power spectrum, and voltage and current waveforms.
Harmonic Balance for Oscillator Simulation

Consider the operation of a negative-resistance oscillator from a point in the network between the resonating circuit and the nonlinear device:

- Ambient noise usually provides the initial stimulus. When a small voltage disturbance at the appropriate frequency is presented at this node, it encounters negative resistance at the device and reflects back to the resonator with a greater magnitude.

- The resonator reflects the signal, which experiences some loss as it propagates through the circuit. As long as the negative resistance of the device is sufficient to exceed the losses in the circuit, the stimulus signal continues to grow.

- At some point, the magnitude of the signal becomes sufficient to reduce the negative resistance of the device.

- Steady state is reached when the magnitude of the signal causes the negative resistance and the loss effects to be balanced. Nonlinearities in the device will generate harmonics, which also reach steady-state values.

There are two unknowns in the process of simulating an oscillator:

- Amplitude of the injection/feedback signal
- Frequency of oscillation

The injection amplitude must be adjusted and applied over a range of frequencies until the oscillation conditions are met at one of the frequencies. Only then is the oscillator’s steady-state operating point obtained.
Troubleshooting a Simulation

This section presents suggestions for improving the accuracy of results.

Solving Convergence and Speed Problems

You may find that simulation of your oscillator circuit does not converge properly. It may converge to a frequency, but with negligible AC power (a DC solution), or it may completely fail to converge. You may find instead that the oscillator analysis moves too slowly and you may want to speed it up. There are several methods you can apply to correct these problems.

- Ensure that the OscPort component is placed properly (refer to “Calculating Large-Signal, Steady-State Oscillation Conditions” on page 9-6).

- Ensure that the OscPort component marks a node in the circuit where a nonzero voltage at the fundamental oscillation frequency is expected. It should not mark a node that contains only a DC bias voltage.

- When there are convergence problems, or when there is reason to question the accuracy of the result, try setting Oversample (under the Params (1) tab) to a value between 1 and 3 (2 is recommended). If sampling within this range does not help, leave Oversample at 2 and try some of the other techniques listed next.

- Modify the value of Order (under the Freq tab). A larger value is helpful when there are convergence problems, or when there is reason to question the accuracy of the result. This value cannot be increased arbitrarily, however, without a severe time penalty. Reducing the value of Order is an effective way to increase the analysis speed. You can often do so without dramatically affecting the results.

- If tolerances are set for Frequency relative tolerance and Frequency absolute tolerance (in the Options component, under the Convergence tab), decrease them to improve convergence and increase them to improve speed. This can help especially in the case of high-Q oscillators. Reductions in frequency tolerance can improve the accuracy of results, but the values should be reduced carefully to avoid very lengthy analyses.
Correcting for Search Failures

Status messages inform you of the simulation process and results for each step in the three-step process. A failure in one step may lead to a failure in a successive step. Be sure to check the status messages to verify that a successful analysis was accomplished. A negligible power level resulting from the analysis is also the sign of a search failure.

If any step fails, consider the following to achieve proper results:

• If the linear oscillation search fails, first verify that the circuit is properly biased and the OscPort component is located properly. If this is so, modify the OscPort parameters NumOctaves, Steps, and MaxLoopGainStep to either expand the frequency search or reduce the step size of the search. When a step size is too coarse, the simulator may not notice high-Q effects.

• If the approximate frequency and power search fails (or is extremely slow), you may need to specify an initial guess at the fundamental voltage. Give a value to the parameter V in the OscPort component.

When You Question the Accuracy of Frequency Results

The accuracy of the completed oscillation frequency is a function of many variables. If you are concerned that results might be inaccurate, consider the following steps:

• It is possible that you have an insufficient number of harmonics to model your circuit. Increase the number of harmonics or add loss effects to your circuit, so that the higher-order harmonics become negligible.

• Decrease the values of Frequency relative tolerance and Frequency absolute tolerance (in the Options component, under the Convergence tab). This forces the simulator to take additional iterations when refining the frequency-of-oscillation solution. The oscillator frequency is considered converged if $|\Delta F| \leq \varepsilon_{\text{rel}} |F| + \varepsilon_{\text{abs}}$

where $\varepsilon_{\text{rel}}$ is Frequency relative tolerance and $\varepsilon_{\text{abs}}$ is Frequency absolute tolerance.

• Decrease the values of Voltage relative tolerance, Voltage absolute tolerance, Current relative tolerance, and Current absolute tolerance (in the Options component, under the Convergence tab). This will increase the accuracy of the voltage and current computations and indirectly improve the accuracy of the oscillation frequency.
Chapter 10: Oscillator Noise Simulation

This chapter focuses on setting up noise simulations for oscillators. It describes how to calculate:

- Phase noise around fundamentals and harmonics
- Absolute noise voltage spectrum around a harmonic
- Relative noise voltage spectrum around a fundamental

There are two methods of noise simulation: using the parameters on the Noise tabs (in the harmonic balance dialog box) or using NoiseCon components. Using NoiseCons, you can set up several noise simulations, thereby eliminating the need to change the values on the Noise tabs in the Harmonic Balance dialog box. With NoiseCons, you can also set parameters to calculate phase noise.

If you are not familiar with the general procedures for simulating oscillators, refer to Chapter 9, Harmonic Balance for Oscillator Simulation before continuing with this chapter.

Refer to the following topics for details on oscillator noise simulation:

- “Simulating Phase Noise Using OscPort” on page 10-2 describes how to set up a phase noise simulation in the context of an example.
- “Simulating Phase Noise with NoiseCons” on page 10-4 describes how to use NoiseCon components to calculate phase noise, relative noise voltage spectrum, phase noise voltage around a harmonic, and absolute noise voltage spectrum,
- “Phase Noise Overview” on page 10-13 describes the calculations made during a phase noise analysis and an overview of the elements of phase noise.

---

**Note** You must have the Harmonic Balance simulator license (included with all Circuit Design suites except RF Designer) to run the following simulations. You can build the examples without the appropriate licenses, but you will not be able to simulate them.

---

**Note** ADS does not currently support the use of the Krylov option in oscillator or phase noise analyses.
Simulating Phase Noise Using OscPort

Figure 10-1 illustrates an example setup for determining oscillator phase noise.

Note This design, OSC3.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in OSC3.dds.

This example is similar to “Calculating Large-Signal, Steady-State Oscillation Conditions” on page 9-6 with the addition of the settings described below.

To determine oscillator phase noise:

1. Label all nodes where you want to collect noise data.
2. Open the Harmonic Balance dialog box and enable Nonlinear noise and Oscillator at the bottom of the dialog.
3. Select the Noise (1) tab and enable Include FM noise (osc. only). Also set these parameters under Noise frequency:
   - Sweep Type = Log
   - Start = 1 kHz
   - Stop = 10 MHz
   - Pts./decade = 2
4. Select Noise (2). In the section Nodes for noise parameter calculation, display the Edit drop-down list and select each node at which to report noise data and click Add. That node is listed in the Select list box.
5. **Simulate.** Plot Gain_out.pnfm and Gain_out.pnmx, for phase noise due to frequency modulation and noise mixing, respectively. When these traces are plotted, they will appear as follows (FM noise is the top trace):

![Phase Noise at node "Gain_out" graph]

Note that the noise is at least 80 dB below the fundamental (resonant) frequency.
Simulating Phase Noise with NoiseCons

When you use a NoiseCon nonlinear noise controller with an oscillator, you have more flexibility in controlling over how noise simulation is performed than you do with Noise(1) and Noise(2). You can compute both single-sideband phase noise as well as a double-sideband noise voltage spectrum. In addition, you can compute phase noise around the oscillator fundamental, an oscillator harmonic or a mixing product. Figure 10-2 shows an oscillator with a frequency multiplier that will be used to demonstrate the use of NoiseCons with an oscillator.

Note  Refer to the project examples/Tutorial/Noisecon_prj for an illustration of how to use NoiseCons for oscillator phase noise simulation. See oscmul.dsn for the design and oscmul.dds for the results.

Four Noisecon components will be used to demonstrate:

- Phase noise around the fundamental
- Relative noise voltage spectrum around the fundamental
- Absolute noise voltage spectrum around a harmonic
- Phase noise around a harmonic
Single Sideband Phase Noise

The first NoiseCon illustrates phase noise around the fundamental frequency of the oscillator.

1. Place a NoiseCon from the Simulation-HB palette. Name it NC1.
2. Select the Freq tab to specify the phase noise offset frequencies. Select a Log sweep and set the Start frequency to 1 kHz, the Stop frequency to 1 MHz and Pts./decade to 3.
3. Select the Nodes tab and enter the name of the oscillator output node, vosc, in the Pos Node and Add it to the list.
4. Select the PhaseNoise tab and select Phase noise spectrum as the Phase Noise Type. Specify the Phase noise carrier (using Frequency) as 100 MHz. The simulator will find the closest actual frequency to this, which is the oscillation frequency of 103.6 MHz.
5. Edit the Harmonic Balance controller and select the NoiseCons tab. Select NC1 and Add it to the list of NoiseCons to be simulated.

The simulated single-sideband phase noise results are shown next. The results in the dataset have the names HB1.NC1.vosc.pnmx and HB1.NC1.vosc.pnfm and are phase noise in dBc. These are the results from the two different phase noise analyses that are performed for an oscillator. The results should be similar. Refer to “How ADS Simulates Phase Noise” on page 10-14 for more information on phase noise analyses.
Simulating Phase Noise with NoiseCons
Relative Noise Voltage Spectrum

The second NoiseCon illustrates the relative noise voltage spectrum. Rather than compute the single sideband phase noise, the noise voltage is computed both at the carrier frequency minus the offset frequency and at the carrier frequency plus the offset frequency. The results are presented as a relative noise voltage spectrum around zero, with the results extending from minus the largest offset frequency to plus the largest offset frequency.

1. From the Simulation-HB palette, select and place a NoiseCon. Name it NC1.

2. Select the Freq tab to specify the phase noise offset frequencies. Select a Log sweep and set the Start frequency to 1 Hz, the Stop frequency to 100 kHz and the Pts./decade to 3.

3. Select the Nodes tab. Select the name of the oscillator output node, vosc, from the Pos Node drop-down list and Add it to the list box for noise parameter calculation.

4. Select the PhaseNoise tab and select Relative vnoise spectrum as the Phase Noise Type. Specify the phase noise carrier (using Carrier mixing indices) as 1, which corresponds to the fundamental oscillation frequency.

5. Edit the Harmonic Balance controller and select the NoiseCons tab. Select NC2 and Add it to the list of NoiseCons to be simulated.

The simulated single-sideband phase noise results are shown next. The result in the dataset has the name HB1.NC2.vosc.noise and is a noise voltage. The spectrum extends from -100 kHz to +100 kHz, and is assumed to be centered relative to the carrier frequency.
Oscillator Noise Simulation

![Graph showing oscillator noise simulation](image)

10-8 Simulating Phase Noise with NoiseCons
Phase Noise Voltage Around a Harmonic

The third NoiseCon illustrates the phase noise spectrum around the seventh harmonic of the oscillator after it passes through a diode frequency multiplier.

1. From the Simulation-HB palette, select and place a NoiseCon. Name it NC3.

2. Select the Freq tab to specify the phase noise offset frequencies. Select a Log sweep and set the Start frequency to 1 kHz, the Stop frequency to 1 MHz and the Pts./decade to 3.

3. Select the Nodes tab and select the frequency multiplier output node, vmul, from the Pos Node drop-down list and Add it to the list box for noise parameter calculation.

4. Select the PhaseNoise tab and select Phase noise spectrum as the Phase Noise Type. Specify the phase noise carrier (using Carrier mixing indices) as 7, corresponding to the frequency multiplier output harmonic with the largest power.

5. Edit the HarmonicBalance controller and select the NoiseCons tab. Select NC3 and Add it to the list of NoiseCons to be simulated.

The simulated single-sideband phase noise results are shown next. The results in the dataset have the names HB3.NC1.vosc.pnmx and HB3.NC1.vosc.pnfm and are phase noise in dBc. These are the results from the two different phase noise analyses that are performed for an oscillator. The results should be similar. Refer to “How ADS Simulates Phase Noise” on page 10-14 for more information on the phase noise analyses. If the phase noise results from this simulation at the seventh harmonic and the phase noise results around the fundamental are compared, the phase noise is seen to be increased $20 \log(7)$, or 16.9 dB, as expected.
Oscillator Noise Simulation

![Graph showing Noise vs. Frequency](image)

10-10 Simulating Phase Noise with NoiseCons
Absolute Noise Voltage Spectrum

The fourth NoiseCon illustrates the absolute noise voltage spectrum. Rather than compute the single sideband phase noise, the noise voltage is computed both at the carrier frequency minus the offset frequency and at the carrier frequency plus the offset frequency. The results are presented as an absolute noise voltage spectrum around the carrier frequency, with the results extending from the carrier frequency minus the largest offset frequency to the carrier frequency plus the largest offset frequency.

1. From the Simulation-HB palette, select and place a NoiseCon. Name it NC4.
2. Select the Freq tab to specify the phase noise offset frequencies. Select a Log sweep and set the Start frequency to 1 Hz, the Stop frequency to 100 kHz and the Pts./decade to 3.
3. Select the Nodes tab and select the frequency multiplier output node, vmul, from the Pos Node drop-down list and Add it to the list box for noise parameter calculation.
4. Select the PhaseNoise tab and select Absolute vnoise spectrum as the Phase Noise Type. Specify the phase noise carrier (using Carrier mixing indices) as 7, corresponding to the frequency multiplier output harmonic with the largest power.
5. Edit the Harmonic Balance controller and select the NoiseCons tab. Select NC4 and Add it to the list of NoiseCons to be simulated.

The simulated single-sideband phase noise results are shown next. The result in the dataset has the name HB1.NC4.vosc.noise and is a noise voltage. The spectrum extends from 725.44 MHz to 725.64 MHz, centered around the seventh harmonic of the oscillator.
10-12 Simulating Phase Noise with NoiseCons
Phase Noise Overview

An oscillator phase noise analysis computes the noise sidebands of the oscillator carrier frequency as well as:

- Normal frequency conversion
- Amplitude-noise-to-frequency-noise conversion
- Frequency translation of noise, caused by component nonlinearities in the presence of large-signal oscillator signals. Upconverted flicker noise is a commonly observed effect.
- Bias changes due to the oscillator signals. Any shift in DC bias that occurs in the presence of the oscillation waveforms is taken into account. This bias-shift calculation is needed for accurate calculations of nonlinear device noise.

Selecting Include FM noise (osc. only) (under the Noise tab) causes the phase noise analysis to include additional phase noise results from a second phase noise analysis based on a frequency sensitivity technique. The results from the basic phase noise analysis have .pnmx (phase noise, in dBc/Hz) and .anmx (AM noise, in dBc/Hz) appended to the selected node name(s). If this option is selected, an additional phase noise result is computed with ".pnfm" (phase noise, in dBc/Hz) appended to the selected node names.

**Note** Most of the nonlinear devices in the semiconductor libraries do not have stable 1/f noise parameters. Instead, noise sources must be added to model the 1/f noise.
Oscillator Noise Simulation

How ADS Simulates Phase Noise

Phase noise in an oscillator can be analyzed by two separate, independent methods: from the oscillator frequency’s sensitivity to noise, and from small-signal mixing of noise. The frequency sensitivity to noise can be viewed as the oscillator acting as a VCO and changing its operating frequency due to FM modulation caused by noise generated in the oscillator. The small-signal mixing of noise comes from the nonlinear behavior of the oscillator, where noise mixes with the oscillator signal and harmonics to mix to sideband frequencies on either side of the oscillator signal. These two models are two different ways of looking at the same problem.

The mixing analysis for phase noise is automatically performed by ADS when an oscillator analysis is performed with nonlinear noise. The frequency sensitivity to noise must be enabled by selecting Include FM noise (osc. only), on the Noise tab in the HarmonicBalance controller dialog box.

These two models are different views of the same process. The phase noise computed by both models is available to the user directly in dBc. Assuming there is a node labeled vout for which phase noise is to be computed, the following variables are available for use in a data display following a phase noise analysis.

- \( \text{hb\_noise\_vout\_pnfm} \) phase noise (in dBc) from frequency sensitivity analysis
- \( \text{hb\_noise\_vout\_pnmx} \) phase noise (in dBc) from mixing analysis
- \( \text{hb\_noise\_vout\_anmx} \) AM noise in (dBc) from mixing analysis

Both variables \( \text{hb\_noise\_vout\_pnmx} \) and \( \text{hb\_noise\_vout\_pnfm} \), which represent the different models of phase noise, should be plotted to view the simulated phase noise results. There should be a region of offset frequencies where they both produce similar results.

The frequency sensitivity to noise is obtained from the large-signal (harmonic balance) oscillator solution. The sensitivity of the oscillation frequency \( \omega_0 \) is obtained with respect to any injected noise in the circuit. After summing over all of the noise sources, the total spectral density of frequency fluctuations is obtained and converted to phase noise. This noise has a characteristic shape of \( f^{-2} \) or \( f^{-3} \) if 1/f noise sources are present. This description is valid at small offset frequencies, but is no good at large offsets as it goes to zero; it will not exhibit a noise floor but continues down as \( f^{-2} \).

The ADS model for frequency sensitivity to noise is based on a sensitivity analysis (evaluated at DC) of the oscillator frequency due to perturbations from noise. This
model may not predict phase noise well if there are lowpass filters in the circuit that tend to filter out noise, e.g. power supply decoupling.

To model oscillator phase noise with a noise mixing analysis, the noise at the sidebands on either side of the carrier \((\omega_0 \pm \omega)\) are obtained from a small-signal mixer analysis where noise sources \((\omega \pm \omega_0)\) mix with the oscillator large signals \((\kappa \omega_0)\) to produce noise sidebands. The noise at these two sideband frequencies and their correlation is then used to compute the phase noise. The phase noise from a mixing analysis tends to be valid at large offset frequencies and will show a finite noise floor. The mixing analysis additionally computes the AM noise as well as phase noise.

**Possible Problems with Phase Noise Analysis**

Not all oscillators will show good agreement between phase noise results from the frequency sensitivity analysis and the noise mixing analysis. There can exist an offset frequency below which the noise mixing analysis shows phase noise that looks flat instead of showing the expected \(f^{-2}\) behavior at smaller offsets. The noise mixing analysis of phase noise is more valid at large offsets. Not all oscillators will exhibit this problem. To some extent, this corner frequency can be reduced by increasing oversample, at the expense of increased run time. A minimum oversample value of 4 is recommended.

For the mixing analysis, it is important that the Use all small-signal frequencies option in the Harmonic Balance dialog box is selected, which is the default. This parameter controls the number of small-signal frequencies used in the mixing analysis; if TRUE, then the sidebands around all of the large-signal frequencies are used \((2 \cdot \text{order} + 1 \text{ small-signal tones});\) otherwise sidebands are used only around the lowest half of the large-signal frequencies \((\text{order} + 1 \text{ small-signal tones}).\) (Note that order is the number of harmonics (including the fundamental) of the oscillator simulated.) Disabling this option can severely underpredict the phase noise from a mixing analysis, especially at small offsets; this is not because the extra sidebands contribute more noise, but is due to a matrix stability issue. Verify that this option is selected to obtain good phase noise results from the mixing analysis.

The following figure illustrates the different components of phase noise for a simple oscillator. Phase noise from the mixing analysis, \(pnmx\), is shown both when all small-signal frequencies are used \((\text{all=yes})\) and when they are not used \((\text{all=no}).\) The phase noise from frequency sensitivity analysis, \(pnfm\), is also shown. Good agreement can be seen between the phase noise from the frequency sensitivity analysis and the...
Oscillator Noise Simulation

mixing analysis all=yes case. There is not a good agreement at smaller offsets when all=no.
Basic Phase Noise Theory

Phase noise occurs naturally in electronic circuits. It can be observed in the time domain as phase jitter of the signal on an oscilloscope display or as time fluctuations of the zero crossings (see the figure that follows).

\[
V(t) = V_s \cos(\omega t)
\]

Frequency and phase are related by:

\[
f(t) = \frac{1}{2\pi} \frac{d\phi(t)}{dt}
\]
Relationship of Phase Deviation and Sideband Level

Phase noise and frequency fluctuations are the same physical phenomenon. Noise in angular frequency can be obtained from the derivative of phase with respect to time. The modulation of the signal phase manifests itself in the sidebands of the oscillator carrier as offsets from the carrier frequency; these offsets are related to the multiples of angle-modulation frequency. For small angle modulation, only the first term is important and the relationship between the phase deviation and the sideband level is approximated as follows:

\[ P_{ssb} \approx \frac{1}{2} \Delta \phi_{rms}^2(f) \]
Single-Sideband Noise Power in a 1-Hz Bandwidth

A common but indirect representation of phase noise is denoted $L(f)$ (see the figure that follows). This is the ratio of the single-sideband noise power (in a 1-Hz bandwidth at an offset frequency from the carrier) to the total carrier power. This common representation is applicable only to small phase deviations.

\[
L(f) = \frac{P_{ssb}(\text{per 1Hz})}{P_s} \quad \text{[dBc/Hz]}
\]
Oscillator Phase-Noise Analysis

The simulator computes the noise at an offset from the unknown oscillation frequency. After the normal harmonic-balance noise analysis has determined the steady-state oscillation frequency and amplitude, the phase-noise analysis computes the noise amplitude as an offset from the carrier. With this noise spectral density $S_v(f)$, the phase noise can be computed as

$$L_f(f) = \frac{S_v(f)}{A_c}$$

where $A_c$ is the amplitude of the fundamental oscillator output.

There are four distinct regions of phase noise, as shown next. Note that not all oscillators will show all four regions.

- The lowest frequency is dominated by flicker FM noise, which is device flicker noise that causes a random frequency modulation. This has a slope of $1/f^3$.
- White FM noise is white noise that causes a random frequency modulation. This has a slope of $1/f^2$.
- Flicker PM noise is modeled by flicker noise that mixes up to the oscillation frequency. This has a slope of $1/f$. 

---

10-20  Phase Noise Overview
• White PM noise is simply white noise that mixes up to the oscillation frequency. This has the typical flat white-noise floor.

**Note**  To model added white noise, select a V\_Noise (Noise Voltage Source) component if you know the rms noise voltage, and an I\_Noise (Noise Current Source) component if you know the rms noise current (these are found in the Noise and Controlled Sources palette). To model flicker PM or flicker FM noise, a device (such as a BJT) with flicker noise parameters must be in the circuit.
Oscillator Noise Simulation
Chapter 11: Harmonic Balance for Mixers

This chapter discusses the use of the Harmonic Balance simulator for standard mixer analyses as well as the small signal option in the Harmonic Balance dialog box that simplifies mixer analyses.

The Small-signal option makes it possible to do a multitone analysis where sidebands are involved, as in a mixer analysis. By reducing by 1 the number of large-signal tones required in a conventional harmonic balance analysis, and eliminating the need to calculate large-signal products, the simulation is computationally efficient.

The starting point of this chapter assumes you are familiar with the basics of harmonic balance simulation, as described in Chapter 7, Harmonic Balance Simulation.

Refer to the following topics for details on harmonic balance for mixers:

- “Performing a Basic Mixer Simulation” on page 11-2 gives the minimum setup for a mixer simulation.
- “Examples” on page 11-3 gives detailed setups for a variety of mixer analyses, such as looking at output tones, performing a small-signal mixer simulation, calculating conversion gain, and calculating intermodulation distortion.
- “Small-Signal Overview” on page 11-17 is a brief explanation of small-signal mixer simulation.
- Chapter 12, Mixer Noise Simulation focuses on noise simulation setups for mixers.
Performing a Basic Mixer Simulation

For a successful analysis:

- By convention, the mixer input port is considered to be port 1, the IF output port is port 2, and the LO input port is port 3. Set up sources and port numbers so that they match the mixer convention. You do this by editing the Num field for these components.

- Ensure that frequencies are established for all of the frequencies of interest—RF, LO, and IF—in the design.

- Add the Harmonic Balance simulation component to the schematic and double-click to edit it. Fill in the fields under the Freq tab:
  - Enter the RF, LO, and IF as fundamental frequencies and set the order.
  - Assign the LO frequency to Freq[1]; it is easier to achieve convergence if the frequency of the signal with the largest amplitude is assigned to Freq[1].
  - Since the number of tones in the simulation can affect simulation time, set the Maximum order to limit the number of tones that are considered in the simulation. For more information, refer to “Harmonics and Maximum Order” on page 7-10.
  - Tone assignment and aliasing error can affect the outcome of a mixer simulation. You may want to select the Params tab and set the FFT Oversample parameter. For more information refer to “Oversampling to Prevent Aliasing” on page 7-23.

For details about other fields, click Help from the dialog box.
Examples

This section gives detailed setups to perform harmonic balance simulations for:

- “Finding Mixer Output Tones” on page 11-3
- “Performing a Small-Signal Simulation of a Mixer” on page 11-7
- “Determining Mixer Conversion Gain” on page 11-10
- “Determining Mixer Intermodulation Distortion” on page 11-14

Note  You must have the Harmonic Balance simulator license (included with all Circuit Design suites except RF Designer) to run the following simulations. You can build the examples without the appropriate license, but you will not be able to simulate them.

Finding Mixer Output Tones

Figure 11-1 illustrates an example setup for determining the amplitudes and relationships of mixer output tones.

Note  This design, Mix1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in Mix1.dds. Refer also to the examples in /examples/RFIC/Mixers_prj.

This example is a two-tone harmonic balance analysis. One tone is used for the RF, one for LO. At the IF we expect to find two fundamental tones at $f_{RF} - f_{LO}$ and $f_{RF} + f_{LO}$. We also expect to find spurious tones (attributed in this case to an intermodulation table). The mixer in this example is a MixerIMT component (available in the System-Data Models palette), which references the intermodulation table db1.imt. Conversion gain is determined by the parameter ConvGain, which uses the function dbpolar to set a gain of 6 dB at an angle of 0 degrees. The reflection parameters $S_{11}$, $S_{22}$, and $S_{33}$ have been set to 0.1, 0.33, and 0.1, respectively.
Note  By convention, the mixer input port is port 1, the IF output port is port 2, and the LO input port is port 3. This is different from the automatic component-naming convention used by the P_1Tone sources when they are placed sequentially in the Schematic window.

You can leave SS_Sideband set to its default value of UPPER for this simulation.

To find the amplitudes and relationships of mixer output tones:

1. From the Sources-Freq Domain palette, select P_1Tone and place one instance of it at the RF input (PORT1) and another at the LO input (PORT2).

2. Edit the RF input source:
   - Num = 1. This is port 1.
   - Z = 50 Ohms (default value)
   - P = dbmtow(-10). This converts the 0-dBm input to watts (the power unit used by the system).
   - Freq = RFfreq. This variable will be defined by a VarEqn component.
3. Edit the LO input source:
   • Num = 2. This is port 2.
   • Z = 50 Ohms (default value)
   • P = dbmtow(7)
   • Freq = LOfreq. This variable will be defined by a VarEqn component.
   • Noise = No. Unless you are doing a noise simulation, this parameter is ignored.

4. From the Data Items palette, select Var eqn (Variables and Equations), and place the VAR component on the schematic. Edit it and select the default equation, replacing it with the following equations:
   • LOfreq = 1750 MHz
   • RFfreq = 2 GHz

5. From the Simulation-HB palette, select the HB simulation component and place it on the schematic. Edit to select the Freq tab and edit the following parameters:
   • Maximum order = 8
   • Frequency = LOfreq. This is Freq[1]. Set its order to 8.
   • Click Add to enter the second fundamental, Freq[2]. Set its frequency to RFfreq, and leave its order set to 8.

Note Normally you would want to set orders for the RF tones lower, to achieve a faster simulation.
6. Simulate. When the simulation is finished, plot $V_{if}$ in dBm, then place markers on the two fundamental IF tones, $RF_{freq} - LO_{freq}$ ($m_1$) and $RF_{freq} + LO_{freq}$ ($m_2$), as shown below:

![Graph showing harmonic balance for mixers](image)

Place more markers to view other frequencies and their magnitudes. The conversion gain is 6 dB, as expected, since the available source power is 0 dBm.
Performing a Small-Signal Simulation of a Mixer

Figure 11-2, similar to the preceding example, illustrates an example setup for performing a small-signal mixer analysis.

**Note**  This design, Mix1_ssmix.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results of the simulation are in Mix1_ssmix.dds. For a detailed example refer to examples/RFIC/MixerDiffMode_prj.

---

**Figure 11-2. Small-signal mixer analysis example**

The Small-signal option, which takes less time and uses less memory, makes it possible to do a multitone analysis where sidebands are involved. When you reduce by one the number of large-signal tones required in a conventional harmonic balance analysis (thereby eliminating the need to calculate large-signal products), the simulation is computationally efficient, because it treats the RF signal as a small-signal, and no harmonics of the RF signal are generated, yet it still forms mixing terms with the LO. This procedure requires that a P_1Tone component be placed where you want to simulate the effects of mixing a fundamental with its sidebands.
Harmonic Balance for Mixers

This procedure uses the upper sideband as an example only. Modify it to view the effects of mixing with the lower sideband. This basic example uses only one RF tone. With two RF tones defined in the RF source, this setup can also be used for a multitone analysis. Small-signal mixer data are indicated by SM.

To perform a small-signal analysis of a mixer:

1. Start with the circuit used in the example, “Performing a Small-Signal Simulation of a Mixer” on page 11-7.

   **Note** The Freq parameter of the P_1Tone component can be empty, as it will be ignored in this analysis.

2. Edit the HarmonicBalance simulation component as follows:
   - Select the Freq tab and ensure that Frequency is LOfreq and its Order is 4. LOfreq (Freq[1]) is defined in the VAR component.
   - Select the Small-Sig tab and ensure that Sweep Type is Single point, and that Frequency is IFfreq (this is the IF resulting from RFfreq – LOfreq). Also select Use all small-signal frequencies.

   **Note** Because we are specifying the power in the upper sideband in the RF source (PORT1), the frequency for this source is the sum of the difference frequency (250 MHz) and the LO frequency (1750 MHz). The RF input is therefore 2 GHz.
3. **Simulate.** When the simulation is finished, place a Rectangular plot of SM.Vif in dBm, as follows:

Markers at 250 MHz (\(f_F\)) and 3.750 MHz (\(f_{LO} + f_{RF}\)) indicate the effects—for our idealistic model—of 6 dB of gain on the input RF power (-60 dBm).
Determining Mixer Conversion Gain

Figure 11-3 illustrates an example setup for determining mixer conversion gain.

Note  This design, Mix1_convgain.dsn, is in the Examples directory under Tutorial/SimModel_prj. The results are in Mix1_convgain.dds.

This example uses the basic Mixer component from the System-Amps & Mixers palette, and also illustrates the use of the Krylov method. While a small-signal mixer analysis can be used to determine conversion gain, the method described here, in which the RF signal is not assumed to be small, should be used if the RF signal is large enough to drive the mixer into compression.
To determine mixer conversion gain:

1. Start with the circuit in the example, “Finding Mixer Output Tones” on page 11-3.

2. Edit the P_1Tone component at the RF source (PORT1)s:
   - \( P = \text{dbmtow}(\text{Power\_RF}) \). \text{Power\_RF} will be established by an equation.
   - \( Freq = \text{RFfreq} \)

3. Edit the P_1Tone component at the LO source (PORT2):
   - \( P = \text{dbmtow}(0) \)
   - \( Freq = \text{LOfreq} \)

4. Using the VAR component, enter the following equations (note the use of the underscore before dBm):
   - \( RF\_power = -30\ _\text{dBm} \)
   - \( LO\_power = -10\ _\text{dBm} \)
   - \( LOfreq = 1750\ MHz \)
   - \( IFfreq = \text{RFfreq} - \text{LOfreq} \)
   - \( \text{RFfreq} = 2000\ MHz \)

   **Note**  In ADS, dBm is not a recognized unit. The underscore in front of dBm (as in _dBm) allows this unit to be used for documentation purposes without causing the simulator to issue a warning.

5. Edit the Harmonic Balance Simulation component, select the Freq tab, and edit the following parameters:
   - Maximum order = 5
   - \( Freq[1] = \text{LOfreq} \)
   - Copy and paste the frequency above, creating \( Freq[2] \), the RF frequency. Then edit it so that its frequency is \( \text{RFfreq} \) and its order is 3.

6. Scroll to the Solver tab, then select Krylov Solver.
Hint To pass the variable RF_power to the dataset for subsequent manipulation by equations, edit the Harmonic Balance Simulation component and select the Display tab. Display the parameter Other, then edit this parameter on the schematic so that Other=OutVar="RF_power".

7. Simulate. When the simulation is finished, do the following:

• Plot Mix(1) and Mix(2) in List format. This will show you the coefficients that generated the sum and difference tones. In this example we are interested in the fundamental IF tone $f_{LO} + f_{RF}$. As indicated in the area outlined in Figure 11-4, these coefficients are 1,1.

Note Inserting a listing column of the frequencies in the simulation also makes it possible to determine which index number to use to select a particular spectral component. For example, if you have a mixer that is upconverting to $f_{LO} + f_{RF}$, and you want to select that spectral component of Vif, then you select Vif[7], because $f_{LO} + f_{RF}$ (= 3.75 GHz) is the seventh frequency in the array.

• Using the mix function, write an equation in the Data Display window to find the power in dBm at $f_{LO} + f_{RF}$, which we know to be 3.750 GHz. That equation is:

$$VIF\text{tone} = \text{dBm}(\text{mix(Vif),(1,1)})$$

• Write an equation in the Data Display window that expresses conversion gain as the difference between IF power ($VIF\text{tone}$) and RF_power:

$$\text{MixConvGain} = VIF\text{tone} - \text{RF\_power}[0]$$

Any number could be used as the tonal index to the right of RF_power, as this value is a constant.

Note The above equations could also be placed as MeasEqn components in the Schematic window. This would allow MixConvGain to be optimized. For a discussion of optimization, refer to the Tuning, Optimization and Statistical Design manual.
Figure 11-4 shows the mixing indexes, the equations, and a List plot of MixConvGain. This figure, 6.000, is precisely the value assigned to the parameter ConvGain in MIX1.

<table>
<thead>
<tr>
<th>freq</th>
<th>Vif</th>
<th>Vif</th>
<th>Mix1</th>
<th>Mix2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.000 Hz</td>
<td>8.79E-17 / 1.000</td>
<td>0.000 / 0.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>250 MHz</td>
<td>6.00E-17 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>500 MHz</td>
<td>2.00E-16 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>1.00 GHz</td>
<td>6.00E-16 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>1.65 GHz</td>
<td>1.15E-16 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>2.00 GHz</td>
<td>0.000 / 1.000</td>
<td>11.000 / 0.000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2.25 GHz</td>
<td>0.000 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>2.50 GHz</td>
<td>0.000 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>3.45 GHz</td>
<td>0.750E-10 / 1.000</td>
<td>0.000 / 0.000</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>3.75 GHz</td>
<td>0.750E-10 / 1.000</td>
<td>0.000 / 0.000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3.95 GHz</td>
<td>2.00E-18 / 213.5</td>
<td>0.000 / 0.000</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4.00 GHz</td>
<td>6.04E-18 / 47.5</td>
<td>0.000 / 0.000</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4.05 GHz</td>
<td>8.30E-18 / 75.4</td>
<td>0.000 / 0.000</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>6.50 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>-4</td>
<td>-1</td>
</tr>
<tr>
<td>5.55 GHz</td>
<td>0.000 / 6.000</td>
<td>0.000 / 0.000</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>5.60 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6.60 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6.50 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>7.40 GHz</td>
<td>0.555E-18 / 11.0</td>
<td>0.000 / 0.000</td>
<td>-4</td>
<td>-3</td>
</tr>
<tr>
<td>7.65 GHz</td>
<td>0.555E-18 / 77.3</td>
<td>0.000 / 0.000</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td>7.50 GHz</td>
<td>1.06E-10 / 14.9</td>
<td>0.000 / 0.000</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8.05 GHz</td>
<td>1.47E-10 / 103.5</td>
<td>0.000 / 0.000</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9.50 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>9.55 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>10.0 GHz</td>
<td>0.000 / 3.000</td>
<td>0.000 / 0.000</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 11-4.
Determining Mixer Intermodulation Distortion

Figure 11-5 illustrates an example for determining mixer intermodulation distortion (IMD). The approach used here is commonly referred to as third-order intercept (TOI) analysis.

Note  This design, Mix2_TOI.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in Mix2_TOI.dds.

This example is similar to “Performing a Small-Signal Simulation of a Mixer” on page 11-7, except that it involves three tones as well as equations that determine Freq[2] and Freq[3].

![Figure 11-5. Mixer intermodulation distortion example](image)

11-14  Examples
To determine mixer intermodulation distortion:

1. Start with the circuit in the example, “Performing a Small-Signal Simulation of a Mixer” on page 11-7.

2. Edit the P_nTone component at the RF source (PORT1) as follows:
   - \( P[1]=\text{dbmtow}(\text{Power\_RF}) \). Power\_RF will be established by an equation.
   - \( P[2]=\text{dbmtow}(\text{Power\_RF}) \)

   **Note** The indexes [1] and [2] are local to this source component only, and are not related to Freq[1] or Freq[2] in the Harmonic Balance Simulation component.

3. Edit the P_1Tone component at the LO source (PORT2) as follows:
   - \( P=\text{dbmtow}(–5) \)
   - \( \text{Freq} = 1750 \text{ MHz} \)

4. Enter the following equations in the VAR component:
   - \( \text{Power\_RF} = –50 \text{ _dBm} \)
   - \( \text{RF\_Freq} = 2000 \text{ MHz} \)
   - \( \text{F\_Spacing} = 100 \text{ kHz} \)

5. Edit the Harmonic Balance Simulation component, select the Freq tab, and edit the following parameters:
   - Maximum order = 8
   - Frequency = 1750 MHz. This is Freq[1], the LO frequency. Set its order to 5.
   - Copy and paste the frequency above, creating Freq[2], the RF frequency. Then edit it so that its frequency is \( \text{RF\_Freq+F\_Spacing}/2 \) and its order is 3.
   - Copy the above to create Freq[3], and edit so that its frequency is \( \text{RF\_Freq-F\_Spacing}/2 \). Its order is also 3.

6. Scroll to the Solver tab, then select Krylov Solver.

7. **Simulate**. When the simulation is finished, plot the following:
Harmonic Balance for Mixers

- Plot $V_{if}$, in dBm, in Rectangular format as shown below. A marker placed on the largest tone ($f_{LO} + f_{RF}$) confirms its frequency on the first line, its power on the second.

  ![Plot of $V_{if}$](image1)

  \text{freq}=3.749750\text{GHz}  \\
  \text{dBm(Mix2_T01..Vif)}=-376.598400

- Plot $V_{if}$ again, this time changing the scale of the $x$-axis as follows:

  Select the plot, then choose \textbf{Edit > Item Options} (or double-click). When the window opens, click the \textbf{Plot Options} tab.

  Deselect \textbf{Auto Scale On/Off}, and make sure that \textbf{xAxis} is selected. Set Min = $3.7498e9$, Max = $3.7502e9$, Step = $4e5$, and click \textbf{OK}. The following will appear, showing the tones centered around the fundamental IF tone ($f_{LO} + f_{RF}$) at 3.750 GHz:

  ![Plot of $V_{if}$](image2)

  \text{freq}=3.750050\text{GHz}  \\
  \text{dBm(Mix2_T01..Vif)}=-44.010380  \\
  \text{freq}=3.750150\text{GHz}  \\
  \text{dBm(Mix2_T01..Vif)}=-112.000000

11-16 Examples
Small-Signal Overview

When the Small-signal mode option is selected, the simulator uses the frequency (or frequencies, in the case of a sweep) entered under the Freq tab to perform a normal large-signal simulation. It then takes the resulting data and performs a small-signal mixer simulation at all the original large-signal frequencies plus and minus the small-signal frequency or frequencies. The simulator essentially linearizes the circuit around the precomputed large-signal operating point.

All the time-varying nonlinearities that result from the presence of large signals are simulated, but with the assumption that the small signal does not change these nonlinearities. For example, a mixer will appear to be perfectly linear with respect to the amplitude of the small-signal input, but all the frequency translation terms will be computed. If a sweep is being performed, then the large-signal analysis is performed only once, followed by the required number of small-signal analyses.

The following assumptions and restrictions apply to mixer analysis:

- It is assumed that the amplitude(s) of the RF signal(s) are sufficiently small so that they generate negligible harmonics.
- It is assumed that the power level of the RF signal(s) is smaller than the power level of the LO signal. The LO source is therefore taken to be that with the highest signal level. For this reason, we do not recommend setting RF and LO sources at the same power level.
Chapter 12: Mixer Noise Simulation

This chapter focuses on setting up noise simulations for mixers. It describes how to calculate:

- Noise analysis frequency translation of the noise
- Nonlinear spot noise
- Nonlinear swept noise

There are two methods of noise simulation. One is to use the parameters on the Noise tabs in the harmonic balance dialog box. A second method is to use NoiseCons. Using NoiseCons, you can set up several noise simulations, thereby eliminating the need to change the values on the Noise tabs in the Harmonic Balance dialog box. With NoiseCons, you can also set parameters to calculate phase noise.

If you are not familiar with the general procedures for simulating oscillators, refer to Chapter 11, Harmonic Balance for Mixers, before continuing with this chapter.

Refer to the following topics for details on mixer noise simulation:

- “Determining Mixer Noise” on page 12-2 is an example of how to set up a simulation to calculate noise.
- “Simulating Mixer Noise with NoiseCons” on page 12-5 illustrates how to perform noise simulations using NoiseCon components.
- “Small-Signal Noise Simulation” on page 12-12 is an overview of how nonlinear swept noise is calculated.

Note You must have the Harmonic Balance simulator license (included with all Circuit Design suites except RF Designer) to run the following simulations. You can build the examples without the appropriate licenses, but you will not be able to simulate them.
Determining Mixer Noise

Figure 12-1 illustrates an example for determining mixer noise. It is based on the example, “Finding Mixer Output Tones” on page 11-3.

Note  This design, Mix1_noise.dsn, can be found in the examples directory under Tutorial/SimModels_prj. The results of the simulation can be found in Mix1_noise.dds.

The additional steps configure the simulation for noise analysis. The MixerIMT parameters are the same as those in the previous example, except that NF (noise figure) has been set to 3 dB. The noise figure computed by a harmonic balance noise simulation is a single sideband noise figure.

Note  Port sources and an output termination (Term component) are necessary only if a noise figure simulation is being performed. The model used in the current design is a 2-port model, and so ignores LO noise. In a circuit-based model, take LO noise into account.

To determine mixer noise:

2. From the Sources-Freq Domain palette, select a Term, place it at the output, and edit it:
   • Num =2. This is port 2.
   • Z =50 Ohms (default value)
   • Noise =No
Figure 12-1. Determining mixer noise example

3. Edit the HarmonicBalance simulation component select Nonlinear noise at the bottom of the dialog box. Select the Noise (1) tab and edit the following fields:
   - Sweep Type = Single Point
   - Frequency = 3.750 GHz (f_lo + f_rf). This is the frequency of the mixer output.
   - Input frequency = RFfreq (frequency of the RF input)
   - Noise input port = 1 (the RF input port)
   - Noise output port = 2 (the output or Term port)

4. From the Simulation-HB palette, select an Options component and place it on the schematic. Set Temp to 16.85 Celsius.
Mixer Noise Simulation

Note  In nonlinear noise analyses, we recommend that the Options component be used to establish a global simulation temperature of 16.85°C if a noise calculation is to be performed. This can be done by editing Temp=16.85 in the Schematic window, or by selecting the Misc tab and editing Simulation temperature to that value. This temperature, 16.85°C or 290 K, is the standard temperature for noise figure measurement as defined by the IEEE definition for noise figureation.

5. Simulate. When the simulation is finished, plot \( nf(2) \) and \( te(2) \) in dB, as shown below. These are the noise figure and equivalent noise temperature (in Kelvin) at port 2, respectively.

<table>
<thead>
<tr>
<th>noisefreq</th>
<th>( nf(2) )</th>
<th>( te(2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.950GHz</td>
<td>3.000</td>
<td>288.626</td>
</tr>
</tbody>
</table>

The noise figure of 3 dB indicates that only the mixer is adding noise (\( NF = 3 \text{ dB} \), from the MixerIMT component).
Simulating Mixer Noise with NoiseCons

When you use a NoiseCon nonlinear noise controller with a mixer, you have more flexibility in controlling how the noise simulation is performed than you do with Noise(1) and Noise(2). You can compute noise figure, noise at different nodes and frequencies, and integrate the effects of oscillator phase noise.

Figure 12-2 shows a mixer that will be used to demonstrate the use of NoiseCons components with a mixer.

Note Refer to the project examples/Tutorial/Noisecon_prj for an illustration of how to use NoiseCons for mixer noise simulation. See mixer.dsn for the design and mixer.dds for the results.

Figure 12-2. Example demonstrating NoiseCons with a mixer
Mixer Noise Simulation

Mixer Noise Figure

The NoiseCon components, NC_USB and NC_LSB, are used to compute the noise figure of the mixer. NC_USB is used to compute the noise figure to the up conversion (RF + LO) frequency. NC_LSB is used to compute the noise figure of the down conversion (RF - LO) frequency.

1. From the Simulation-HB palette, select a NoiseCon, place it in the design and name it NC_USB.
2. Select the Freq tab, select Single point for the frequency and set it to 1750 MHz, the up-conversion frequency for this mixer.
3. Select the Misc tab, set Noise input port to 1 and Noise output port to 2. Set the Input frequency to 900 MHz.
4. Place another NoiseCon and name it NC_LSB.
5. Select the Freq tab, select Single point for the frequency and set it to 50 MHz, the down-conversion frequency for this mixer.
6. Select the Misc tab, set Noise input port to 1 and Noise output port to 2. Set the Input frequency to 900 MHz.
7. Edit the Harmonic Balance controller and select the NoiseCons tab. Select NC_USB and NC_LSB and Add them to the list of NoiseCons to be simulated.

The simulation results are shown next.
Simulating Mixer Noise with NoiseCons 12-7

- NC_USB.nf(2) is the noise figure when the mixer is used for up conversion.
- NC_LSB.nf(2) is the noise figure when the mixer is used for down conversion.

These values are slightly different due to high-frequency rolloff in the mixer.

### Noise at Different Nodes and Frequencies

The NoiseCon components, NC_RF and NC_IF, are used to compute noise voltage at different nodes and frequencies in one simulation. NC_RF also illustrates the use of a differential noise measurement.

1. From the Simulation-HB palette, select a **NoiseCon**, place it in the design and name it **NC_RF**.
2. Select the **Freq** tab, select **Single point** for the frequency and set it to **900 MHz**, the RF frequency for this mixer.
3. Select the **Nodes** tab to compute both noise at a single node and differential noise between two nodes. The nodes mix1.RFampp and mix1.RFampn are two internal nodes in the mixer subnetwork at the output of the internal RF preamplifier.
Mixer Noise Simulation

- First, select the node name `mix1.RFampn` from the Pos Node drop-down list. Make sure the Neg Node entry field is blank. Now click Add to create the first entry in the list of nodes. This computes the noise at one node.

- Next, select the node name `mix1.RFampp` from the Pos Node drop-down list and select the node name `mix1.Rfampn` from the Neg Node drop-down list. Click Add to create the second entry in the list of nodes. This computes the differential noise between two nodes.

4. Select the Misc tab and erase the default values for Noise input port and Noise output port so the noise figure is not computed with this NoiseCon.

5. Place another NoiseCon and name it NC_IF.

6. Select the Freq tab, select Single point for the frequency and set it to 50 MHz, the IF frequency for this mixer.

7. Select the Nodes tab, enter the node name vif for Pos Node and click Add.

8. Select the Misc tab, erase the default entries for Noise input port and Noise output port so the noise figure is not computed with this NoiseCon.

   Note that this could have also been done on the NC_LSB NoiseCon.

9. Edit the Harmonic Balance controller and select the NoiseCons tab. Select NC_RF and NC_IF and Add them to the list of NoiseCons to be simulated.

Note the following when viewing the results.

- **NC_RF.mix1.RFampp.noise** is the noise at the single node in the RF preamplifier at the RF frequency.
- **NC_RF.mix1.RFampp_minus_mix1.RFampn** is the noise measured differentially across the two nodes in the RF preamplifier at the RF frequency.
- **NC_IF.vif** is the noise at the output of the mixer at the IF frequency.

Effects of LO Phase Noise on Noise Figure

You can set up an analysis, using a NoiseCon, that includes the effect of LO phase noise on the noise figure of the mixer. Figure 12-1 shows the setup to be used. This design is similar to the first design except the P_1Tone ideal LO source has been replaced with an OSCwPhNoise source that includes phase noise.
Note Refer to the project examples/Tutorial/Noisecon_prj for an illustration of how to use NoiseCons for mixer noise simulation. See mixer_pn.dsn for the design and mixer_pn.dds for the results.

Figure 12-3. Setup using NoiseCons with a mixer and effects of LO phase noise

Two NoiseCons are used to set up this simulation. The first computes the mixer noise figure without phase noise. This is done by setting the mixer input frequency and (output) noise frequency to be exactly equal to the large signal frequencies. No LO phase noise is considered. The second computes the mixer noise figure including the phase noise from the LO. The input and output frequencies will be the same, but the total noise power at the output will be obtained by integrating over a bandwidth of 30 kHz centered around the IF frequency.
Mixer Noise Simulation

1. From the Simulation-HB palette, select and place a NoiseCon and name it NC1. This will be set up the same as the NC_LSB that was used previously.

2. Select the Freq tab, select Single point for the frequency and set it to 50 MHz, the output frequency for this mixer.

3. Select the Misc tab, set the Noise input port to 1 and the Noise output port to 2. Set the Input frequency to 900 MHz.

4. Place another NoiseCon and name it NC2. This will be used to compute the mixer noise figure including the effects of phase noise integrated over a 30 kHz bandwidth.

5. Select the Freq tab to set up the frequency that will serve as the offset frequency for the phase noise analysis that will be swept over the 30 kHz bandwidth. The mixer output frequency for this analysis will be set later on the PhaseNoise tab. Select a Log sweep, set the Start frequency to 1 Hz and the Stop frequency to 100 kHz. Set the Pts./decade to 3.

6. Select the Misc tab, set the Noise input port to 1 and the Noise output port to 2. Set the Input frequency to 900 MHz.

7. Select the PhaseNoise tab, select a Phase Noise Type of Integrate over bandwidth. The phase noise carrier frequency, which for this example is the output frequency of the mixer, is specified using Carrier mixing indices. For an IF frequency of 50 MHz, the mixing indices of interest is 1,-1. Enter the first Index of 1 and click Add; then enter the second Index of -1 and click Add.

8. Edit the Harmonic Balance controller and select the NoiseCons tab. Select NC1 and NC2 and Add them to the list of NoiseCons to be simulated.

The simulation results are shown next.

<table>
<thead>
<tr>
<th>noisefreq</th>
<th>NC1_nf(2)</th>
<th>NC2_nf(2)</th>
<th>Cgain_Lsb</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.00MHz</td>
<td>12.518</td>
<td>62.865</td>
<td>16.458</td>
</tr>
</tbody>
</table>

- NC1_nf(2) is the noise figure without phase noise.
- NC2_nf(2) is the noise figure including the effects of LO phase noise.
Note that the second value is larger than the first, demonstrating the degradation of the mixer noise figure due to the extra noise injected by the LO phase noise.

A standard noise figure computation is performed using the equation:

$$NF = 10 \log \left( \frac{v_n^2(f)/R}{kT_0G} + 1 \right)$$

where

- $k$ is Boltzmann’s constant ($1.389658 \times 10^{-23}$)
- $T_0$ is the IEEE standard temperature for noise figure (290 K)
- $G$ is the conversion gain of the mixer
- $R$ is the resistance of the output termination
- $v_n$ is the noise voltage at the output port at the output frequency where the input and output terminations do not contribute any noise

When the effects of phase noise are included, the $v_n^2(f)/R$ term is replaced with total noise power at the output integrated over a bandwidth $B$ centered at the output IF frequency $f_{IF}$:

$$NF = 10 \log \left( \frac{\int_{(f_{IF} - B/2)}^{(f_{IF} + B/2)} (v_n^2(f)/R) df}{kT_0G} + 1 \right)$$

In a mixer, the LO will generate noise over a range of frequencies from $f_{LO} - B/2$ to $f_{LO} + B/2$. These will mix with the RF tone at $f_{RF}$ to produce noise from $f_{IF} - B/2$ to $f_{IF} + B/2$, where $f_{IF} = |f_{RF} - f_{LO}|$. 
Small-Signal Noise Simulation

The small-signal noise simulator is a subset of the nonlinear noise simulator. (For a discussion of nonlinear noise, refer to “Nonlinear Noise Overview” on page 8-5.) In a small-signal noise analysis, there is no frequency translation of the noise. Since the noise is assumed to be a small-signal perturbation, there can be no noise mixing without a large-signal harmonic balance source. The only mixing frequencies that are accounted for are the upper and lower sidebands of the normal harmonic-balance analysis.

Nonlinear Spot-Noise Simulation

A nonlinear spot-noise simulation computes the noise at a specific noise frequency, as a function of another variable (such as LO power). For noise figure to be calculated, noise must be injected at the input port. The noise input frequency is entered into the Input frequency field of the Noise (1) tab. Since frequency translation occurs in a mixer, you must specify at which input frequency noise is to be injected.

For example, for a downconverter you would specify

\[ \text{InputFreq} = \text{noisefreq} + \text{LOfreq} \]

where LOfreq is the local oscillator frequency input to the mixer.

The downconverter will shift INPUTFREQ to

\[ \text{InputFreq} - \text{LOfreq} = (\text{noisefreq} + \text{LOfreq}) - \text{LOfreq} = \text{noisefreq} \]

For an upconverter, INPUTFREQ would be

\[ \text{InputFreq} = \text{noisefreq} - \text{LOfreq} \]

Single-Frequency Nonlinear Spot Noise

The simulation results of a single-point analysis have no independent variable and therefore cannot be plotted on a rectangular plot, however, the data can be displayed in a listing column. Note that noise voltages are always rms.

If the amplitude of the RF signal is small compared to that of the LO, then the noise mixed by the LO and its harmonics will dominate, and simulations can be run much more quickly by simply including the LO and its harmonics in the harmonic balance noise analysis.
Multiple-Frequency Nonlinear Spot Noise

A sweep can also be used to compute noise at the specified noise frequency as a function of a variable. If an independent variable is in the nonlinear noise data, the data can be plotted on a rectangular plot.

Swept-Noise Simulation

A nonlinear swept-noise simulation computes the noise at a swept list of noise frequencies with another optional swept variable (for example, LO power). The result is analogous to a spectrum analyzer display. Nonlinear swept-noise analysis differs from nonlinear spot-noise analysis in that the noise frequency (in the Noise frequency area under the Noise (1) tab) is allowed to be swept in a manner similar to the way a spectrum analyzer sweeps a noise floor.

The noise frequency can be an expression if the noise frequency is swept. For example, to compute the noise figure of an upper sideband mixer in which the noise output frequency is swept but the LO frequency is fixed, set the parameter Input frequency as follows:

\[ \text{InputFreq} = \text{noisefreq} + \text{LOfreq} \]

where noisefreq is the noise frequency and LOfreq is an equation defining the fixed LO frequency.

The results of the analysis can be plotted on a rectangular plot. The independent variable of the swept nonlinear noise analysis is always noisefreq.
**Computing Mixer Noise Figure**

ADS computes the mixer noise figure using the IEEE standard definition of single sideband noise figure:

\[
F_{ssb} = \frac{(v_n^2/R_l) + kT(G_1+G_2+\ldots+G_n)}{(kT)(G_1)}
\]

\[
NF_{ssb} = 10 \times \log_{10}(F_{ssb})
\]

It is computed with the input port specified by the Noise input port parameter, and the output port specified by the Noise output port parameter. It is saved in the dataset as the variable \(nf(i)\), where \(i\) is the number of the noise output port. ADS also saves this value as \(NF_{ssb}\) in the output dataset.

ADS computes the double sideband noise figure using the following equation:

\[
F_{dsb} = \frac{kT(G_1+G_2+\ldots+G_n) + (v_n^2/R_l) + \ldots}{kT(G_1+G_2+\ldots+G_n)}
\]

\[
NF_{dsb} = 10 \times \log_{10}(F_{dsb})
\]

This is the value that would be computed by a noise figure meter. It is saved in the dataset as \(NF_{dsb}\).
Chapter 13: Large-signal S-parameter Simulation

The LSSP Simulation component, in the Simulation-LSSP palette, computes S-parameters for nonlinear circuits such as those that employ power amplifiers and mixers. In the latter case, S-parameters can be computed across frequencies, that is, from the RF input to the IF output. LSSP simulation is based on the harmonic balance simulation and uses harmonic balance techniques.

Refer to the following topics for details on large-signal S-parameter simulation:

- “Performing a Large Signal S-parameter Simulation” on page 13-2 shows the minimum setup requirements for a large-signal S-parameter simulation.
- “Example” on page 13-3 is a detailed setup for calculating large-signal S-parameters, using a BJT as the example.

For an example of an LSSP analysis of a mixer circuit, refer to the design LSSP2.dsn, and its associated dataset, LSSP2.dds. It is in the same project folder as the BJT example. This example illustrates how to use a term at the output port, and uses a voltage source at the LO.

- “LSSP Overview” on page 13-7 is a brief explanation of the LSSP simulator and how it compares to the S-parameter simulator.

Note You must have the LSSP simulator license to run the simulation. You can build the examples in this chapter without the license, but you will not be able to simulate them.
Performing a Large Signal S-parameter Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis:

- Apply ports to all inputs and outputs. Use a P_1Tone or P_nTone power source to drive a port. Terminate other ports using port-impedance terminations (Term). Verify impedance.

  **Note** The power level at a passive port (Term) will be calculated by turning on power sources and measuring the power at the port; this value will be used to drive the port.

- Check the Num field for each port. The S-parameter port numbers are derived from these fields. For a 2-port circuit, you want the input labeled as Num = 1 and the output as Num = 2.
- In circuits with mixers, use a voltage source for the LO, not a power source. This prevents the LO input from being recognized as a port and consequently having the S-parameters calculated with respect to it.
- Add the LSSP control element to the schematic. Double-click to edit it. Fill in the fields under the Freq and Ports tabs:
  - For Freq, set the fundamental frequency and order.
  - For Ports, set the port frequency for each port. S-parameters will be measured at this frequency. Port frequency can be the input frequency or a harmonic.
  - The Krylov option can be enabled for circuits that contain a large number of nonlinear devices or harmonics. Refer to “Selecting a Solver” on page 7-15 for instructions on how to use this option.
  - You can use previous simulation solutions to speed up the simulation process. For more information, refer to “Reusing Simulation Solutions” on page 7-17.

For details about each field, click Help from the dialog box.
Example

Figure 13-1 illustrates the setup for a large-signal S-parameter (LSSP) simulation of a BJT. Power sources drive all ports in this example.

Note  This design, LSSP_test.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in LSSP_test.dds.
Large-signal S-parameter Simulation

1. From the Simulation-LSSP palette, select a P_1Tone component and place it at the input of the circuit. Edit the component and set the following values:
   • Num=1
   • \( P = \text{dbmto}((10),0) \)
   • \( Freq=\text{LSSP}_\text{freq} \)

2. Select another P_1Tone power source and place it at the output of the circuit. The values are the same as for the input source, except here Num = 2.

   **Note** The values in the Num parameter on the sources and terminations should reflect the placement of the ports in the circuit, so that the S-parameter data is meaningful. The number of the input (source) should be set to Num=1, and that of the output (load) to Num=2.

3. Select and place an LSSP simulation component on the schematic, edit it, and select the Freq tab to set the following parameters:
   • Frequency=\( \text{LSSP}_\text{freq} \)
   • Order[1]=3

   Click Add. Make sure that 1 \( \text{LSSP}_\text{freq} \) 3 is the only line that appears in the list of fundamental frequencies. (If \( \text{LSSP}_\text{freq} \) appears as the second fundamental in the frequency list, select the line above it and click Cut.)

4. Select the Sweep tab. Ensure that Start/Stop is selected and Sweep Type is Linear, then set the following values:
   • Start = 0
   • Stop = 100
   • Step = 1
5. Select the **Ports** tab. In the **Frequency** field, at the right of the dialog box, enter LSSP_freq and click **Add**. This establishes the frequency at port 1, where the large-signal S-parameter will be measured. It does not have to be the same value as the fundamental frequency, it can be a harmonic.

For example, a port may have harmonics present at 0 Hz, 1 MHz, 99 MHz, 100 MHz, and 101 MHz. You can then specify your interest in the 99 MHz component by entering 99 MHz here. On the schematic, this appears as LSSP_FreqAtPort[n]. In this example, it would appear as LSSP_FreqAtPort[2]=99 MHz.

6. To set the frequency for port 2, click **Add** again. You should see two entries in the Port Frequency list box, each set to LSSP_freq.

7. Click **OK** to accept changes and close the dialog box.

8. From the **Data Items** palette, select **VarEqn**. Place and edit the component to define the variable LSSP_freq and set its value. Select the default equation (X=1.0). In the Variable Value field at the right, enter 1.0 GHz.

9. Click **OK** to accept changes and close the dialog box.

10. Since the fundamental is set to a single frequency, you can use a ParamSweep component to sweep a frequency range. Return to Simulation-LSSP, select and place a ParamSweep component, and edit it. Select the **Sweep** tab and set the following values:

    • Parameter to sweep = LSSP_freq
    • Sweep Type = Linear
    • Enable Start/Stop
    • Start = 1 GHz
    • Stop = 10 GHz
    • Step-size = 0.1 GHz

11. Click **Simulations** and set Simulation 1 to HB1. Click **OK**.

12. Launch the simulation, and when it is finished, display the results. LSSP data items may identified with an HB prefix. The following plot displays S(1,2).
Large-signal S-parameter Simulation

![Graph showing Large-signal S-parameter Simulation]
LSSP Overview

Unlike small-signal S-parameters, which are based on a small-signal simulation of a linearized circuit, large-signal S-parameters are based on a harmonic balance simulation of the full nonlinear circuit. Because harmonic balance is a large-signal simulation technique, its solution includes nonlinear effects such as compression. This means that the large-signal S-parameters can change as power levels are varied. For this reason, large-signal S-parameters are also called power-dependent S-parameters.

Like small-signal S-parameters, large-signal S-parameters are defined as the ratio of reflected and incident waves:

\[ S_{ij} = \frac{B_i}{A_j} \]

The incident and reflected waves are defined as:

\[ A_j = \frac{V_j - Z_{0j} I_j}{2 \sqrt{R_{0j}}} \]  \[ B_j = \frac{V_i + Z_{0i}^* I_j}{2 \sqrt{R_{0i}}} \]

where

- \( V_i \) and \( V_j \) are the Fourier coefficients, at the fundamental frequency, of the voltages at ports \( i \) and \( j \),
- \( I_i \) and \( I_j \) are the Fourier coefficients, at the fundamental frequency, of the currents at ports \( i \) and \( j \),
- \( Z_{0i} \) and \( Z_{0j} \) are the reference impedances at ports \( i \) and \( j \), and
- \( R_{0i} \) and \( R_{0j} \) are the real parts of \( Z_{0i} \) and \( Z_{0j} \).

This definition is a generalization of the small-signal S-parameter definition in that \( V \) and \( I \) are Fourier coefficients rather than phasors. For a linear circuit, this definition simplifies to the small-signal definition.
LSSP Simulation Process

The simulator performs the following operations to calculate the large-signal S-parameters for a two-port:

- Terminates port 2 with the complex conjugate of its reference impedance. Applies a signal with the user-specified power level $P_1$ at port 1, using a source whose impedance equals the complex conjugate of that port's reference impedance. Using harmonic balance, calculates the currents and voltages at ports 1 and 2. Uses this information to calculate $S_{11}$ and $S_{21}$.

- Terminates port 1 with the complex conjugate of its reference impedance. Applies a signal of power $P_2 = |S_{21}|^2 P_1$ at port 2 using a source whose impedance equals the complex conjugate of the reference impedance of port 2. Using harmonic balance, calculates the currents and voltages at ports 1 and 2. Uses this information to calculate $S_{12}$ and $S_{22}$.

Comparing LSSP and S-parameter Simulations

S-parameter simulations are performed on linear circuits. LSSP simulations can be performed on nonlinear circuits and thus include nonlinear effects such as gain compression and variations in power levels.

To compare LSSP and S-parameter simulations, refer to LSSP_test.dsn and SP_test.dsn. They are in the examples directory under Tutorial/SimModel_prj. The data displays are LSSP_test.dds and SP_test.dds.

For a review of the S-parameter simulator, refer to “S-parameter Simulation Overview” on page 6-10.
Chapter 14: P2D Simulation

The P2D simulator can be used to generate a .p2d file, which contains S-parameters that can be used to develop component models. A P2D simulation generates power-dependent S-parameters, which can then be used in a behavioral model for faster, system-level simulations. The resulting .p2d file can be accessed by components such as AmplifierP2D (System-Data Models palette) by selecting P2D Filename as the Parameter Entry Mode for the P2DFile= parameter.

Refer to the following topics for details on P2D simulation:

- “Performing a P2D Simulation” on page 14-2 describes the minimum setup requirements for generating a .p2d file from a simulation.
- “Example” on page 14-3 is a detailed setup for generating a .p2d file from a simulation.
- “Using the P2D file” on page 14-6 describes how to use the file with a component that accepts file-based parameters.
- “Using Terms in a P2D Simulation” on page 14-6 describes how to terminate the circuit with Term components instead of sources and loads.
- “P2D Format” on page 20-46 describes how to write a .p2d file using a text editor. This is as an alternate method of creating a .p2d.

Note You must have the LSSP license to run the simulator. You can build the examples in this chapter without the license, but you will not be able to simulate them.
Performing a P2D Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis:

- Add the P2D control component to the schematic and double-click to edit it. Fill in these fields under the Freq tab:
  - Specify the name of the .p2d file. This is where the S-parameters that are generated by the simulation will be saved. The .p2d file will be saved in the project in the same location where dataset (.ds) files are saved (/data directory), unless you specify a path.
  - Select a sweep type. For a single point, enter the frequency. For a linear or logarithmic sweep, elect to define the frequency sweep with start/stop or center/span values.
- For a power sweep, fill in these fields under the Power tab:
  - Select a sweep type. For a single point, enter the power level for that point. For a linear or logarithmic sweep, define the power levels of the sweep with start/stop or center/span values. Power is in dBm.
  - You can perform budget calculations as part of the simulation. For more information on budget analyses, refer to Chapter 18, Using Circuit Simulators for RF System Analysis.
  - You can apply the Krylov subspace solution option for faster calculations of circuits with large numbers of nonlinear devices or large numbers of harmonics. Do not select this option for one-tone or power-sweep problems. Refer to “Selecting a Solver” on page 7-15 for instructions on how to use this option.
  - You can use previous simulation solutions to speed up the simulation process. For more information, refer to “Reusing Simulation Solutions” on page 7-17.

For details about each field, click Help from the dialog box.
Example

This example simulates an amplifier, sweeping input power over various levels. Figure 14-1 illustrates the setup for simulating a simple amplifier.

![Figure 14-1. Setup for P2D Simulation](image)

1. From the Simulation-LSSP palette, select an P2D simulation component, place it on the schematic, and double-click to edit it.

2. Select the Frequency tab. In the P2D file to output field, enter a name for the .p2d file. This is the file that will be generated by the simulation. The filename must be in quotes. The default filename is “default.p2d”.

3. In the Order field, accept the default value of 3 or edit this value.

4. Select a sweep type and values for Start, Stop, and Step-size as desired. Optionally, you can select a sweep plan.

5. Select the Power tab and select a sweep type, as well as values for Start, Stop, and Step-size as desired. Optionally, you can select a sweep plan. Power is in dBm.
Note: The Krylov option should not be selected for one-tone problems such as this, especially when they involve sweeping.

6. Simulate. The result is a .p2d file that can be used in subsequent simulations. Note that with these power sweep settings, gain compression will occur in the amplifier. You can see this if you set up and run this simulation, then plot the results HB.S(2,1) on a rectangular plot. You can view the .p2d file with any ASCII text editor. A portion of the file from the example is shown here.
BEGIN ACDATA
# AC( MHZ  S  RI  R 50  FC 1 0 )
! small signal s-parameter
% F n11x n11y n21x n21y n12x n12y n22x n22y
1500  -0.14973 -0.74466 1.05898e-003 -4.57539 -5.8788e-002 -3.51371e-002 0.841147 -0.275706
1700  -3.54194e-002 5.47026e-002 -5.5386 -2.94283 -9.49884e-002 4.81790e-002 0.287816 -0.647277
1900  0.506917 -8.94666e-002 -4.61943 1.34196 -3.60905e-002 8.40664e-002 4.42222e-002 -0.382819
2000  0.566256 -0.241273 -3.64262 2.06492 -1.75349e-002 8.20927e-002 4.42686e-002 -0.321673
! power dependent s-parameter
% F
1500
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
5  18.1744 -0.14594 -0.75056 3.43087e-002 -4.55728 -5.81017e-002 -3.66150e-002 0.799862 -0.247955
6  19.1668 -0.145027 -0.751947 4.20916e-002 -4.55323 -5.73255e-002 -3.76813e-002 0.766918 -0.225389
7  20.158 -0.143936 -0.753596 5.12981e-002 -4.54852 -5.58564e-002 -3.92082e-002 0.715344 -0.190726
8  21.1449 -0.142695 -0.75555 6.20423e-002 -4.54154 -5.39404e-002 -4.09731e-002 0.651845 -0.149458
9  22.0968 -0.141491 -0.758293 7.63355e-002 -4.51623 -5.18580e-002 -4.27849e-002 0.583724 -0.106445
10  22.9241 -0.140564 -0.763445 0.101229 -4.4268 -4.99511e-002 -4.43975e-002 0.521083 -6.77973e-002
% F
1700
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
5  21.0224 -6.04337e-002 4.76748e-002 -5.51531 3.09798 -1.02685 4.12691e-002 0.237226 -0.561164
6  22.0197 -6.64604e-002 4.68957e-002 -5.49727 -3.12597 -1.06569 3.65968e-002 0.207403 -0.509516
7  22.9328 -7.7704e-002 4.93914e-002 -5.41591 -3.1412 -1.106 3.15724e-002 0.175171 -0.455106
8  23.7037 -9.74298e-002 5.46652e-002 -5.23313 -3.13353 -1.14291 2.68724e-002 0.145083 -0.405278
9  24.0993 -8.99288e-002 1.09087e-002 -4.85087 -2.97041 -1.16225 2.43362e-002 0.128901 -0.378963
10  24.3455 -5.62783e-002 -4.99032e-002 -4.46457 -2.69559 -0.117431 2.27256e-002 0.118628 -0.362467
% F
1900
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
5  18.7339 0.501498 -8.06491e-002 -4.67845 1.31828 -3.72235e-002 8.40764e-002 3.84839e-002 -0.374854
6  19.7564 0.500116 -7.85094e-002 -4.69323 1.31235 -3.74907e-002 8.41649e-002 3.65473e-002 -0.372692
7  20.783 0.49845 -7.59467e-002 -4.71075 1.30514 -3.79189e-002 8.43492e-002 3.29121e-002 -0.368753
8  21.8042 0.496769 -7.26969e-002 -4.72576 1.29569 -3.90612e-002 8.46008e-002 2.40136e-002 -0.357892
9  22.7553 0.496553 -6.66354e-002 -4.70825 1.27675 -4.14758e-002 8.47738e-002 7.27654e-003 -0.335699
10  23.5895 0.499408 -5.39443e-002 -4.61837 1.23452 -4.44966e-002 8.48838e-002 -1.32114e-002 -0.308663
Using the P2D file

Once the .p2d file is generated, you can use it to describe the behavior of a component such as the AmplifierP2D or MixerP2D, using file-based parameters.

1. From the System-Data Models palette, select the AmplifierP2D component and place it on the schematic. Double-click to edit it.

2. Select P2DFile= from the Select Parameter list box. In the File Name field, enter the name of the .p2d file. Alternatively, use the Browser to select a data file from within the project or click Data files list to select a file from all files found based on the paths set for the DATA_FILES variable. If using a file outside the project, include the complete path.

Using Terms in a P2D Simulation

When setting up a circuit for a P2D simulation, you should substitute Term components for both the source and load. By setting the Power parameters on the P2D simulation component, you can sweep input power to the circuit (see example).
Chapter 15: Gain Compression Simulation

To calculate the X dB gain compression point of an amplifier or mixer, use the XDB simulation component, from the Simulation-XDB palette. The simulator sweeps the input power upward from a small value, and when the required amount of gain compression is seen at the output, the analysis is complete. A common calculation is the 1-dB gain-compression point; this is the default.

Refer to the following topics for details of gain compression simulation:

- “Performing a Gain Compression Simulation” on page 15-2 describes the minimum setup requirements for a gain-compression simulation.
- “Example” on page 15-3 is a detailed setup for calculating gain compression, using a power amplifier as the example.
- “Gain Compression Overview” on page 15-8 is a brief explanation of the gain compression calculation, plus information on how it compares to a harmonic balance simulation.

Note You must have the Gain Compression license to run the simulator. You can build the examples in this chapter without the license, but you will not be able to simulate them.
Performing a Gain Compression Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis:

- Use port-type sources at the inputs, such as the P_1Tone under Sources-Freq Domain.
- Terminate outputs with port-impedance terminations. You can find this type of port under Simulation-S_Param. Verify the impedance value.
- The Num values in the port-impedance terminations are used to specify input and output ports.
- Add the Gain Compression control element to the schematic. Fill in the fields under the Freq and Xdb tabs:
  - For Freq, set the fundamental frequency and order.
  - For X db, specify the input and output ports, the frequency at each port, power variation for each port, and maximum input power.
- For a faster simulation of large circuits, use the Krylov option. Refer to “Selecting a Solver” on page 7-15 for instructions on how to use this option.
- You can use previous simulation solutions to speed up the simulation process. For more information, refer to “Reusing Simulation Solutions” on page 7-17.

For details about each field, click Help from the dialog box.
Example

Figure 15-1 illustrates the setup for a basic gain compression simulation of a power amplifier circuit.

Note  This design, XDB1.dsn, is in the examples directory under Tutorial/SimModels_prj. The simulation results are in XDB1.dds.

Figure 15-1. Gain-compression simulation example
Gain Compression Simulation

To perform a gain-compression simulation:

1. From the Sources-Freq Domain palette, select a port-type source such as a P_1Tone or P_nTone. Place the source at the input of the component or circuit under test, then define input power and edit other parameters as required.

2. From the Simulation-S_Param palette, select a Term component and place it at the output of the component or circuit under test. Edit the impedance if necessary.

3. Ensure that the input port source is identified as Num=1, and the output port termination as Num=2. Also ensure that you have labeled the inputs and outputs of nodes at which you want data to be reported.

4. From the Simulation-XDB palette list, select an XDB simulation component, and place it on the schematic, then edit it.

5. Select the Freq tab, and edit frequencies, orders, and fundamentals as required.

---

**Hint** In a one-tone simulation such as this one, MaxOrder is irrelevant.

- Ensure that values are established for all of the frequencies of interest (for example, RF, LO, and IF frequencies) in the design under test. (Where mixers are employed, set the LO frequency to Freq[1]; it is easier to achieve convergence if the frequency of the signal with the largest amplitude is assigned to Freq[1].) To add or delete frequencies, select them in the Select field, then click Add or Cut. Display them on the schematic to facilitate editing.
6. Select the X dB tab and set the following:
   - In the Gain compression field, either accept the default or enter a new gain-compression value. The default is 1 dB.
   - Under Port numbers, ensure that appropriate values are entered for the input port source (generally Num=1) and the output termination (generally Num=2).
   - Under Port frequencies, set appropriate values for the input and output frequencies. In a circuit employing a mixer, the input frequency would be that of the RF input, the output frequency that of the IF output.
   - Under Power tolerances, set appropriate values for acceptable variations in power at the input and output ports. In the Max. Input Power field, set the maximum input power over which the simulator is allowed to search.

7. To reduce memory requirements significantly where many frequencies and orders are involved, select the Solver tab, then select Krylov Solver.

---

**Note** We do not recommend using the Krylov option on small problems. For details, refer to “Selecting a Solver” on page 7-15.
8. **Simulate.** Gain compression data items may be identified with an HB prefix. The following is a Rectangular plot of the results of the simulation. At the top are the gain compression results, with the standard harmonic balance results at the bottom for comparison.
The following graphs and their associated equations illustrate how to extract more meaningful results from the simulation. The equation Gain finds gain by subtracting the input power Pin from the output voltage of the fundamental \([\text{Vout}[1]]\). The equation linear adds input power to gain to produce the linear plot in the graph. To find the input power that leads to a compression point of approximately 1 dB, place a marker on each trace and move them so that their difference is as close to 1 dB as the simulation results allow. The simple equation compression subtracts the values of the markers, and a List plot of that equation shows the input power that yields a gain compression of approximately 1 dB.

\[
\text{Gain} = \text{dBm}(\text{XDB1,H1,H1,H1, Vout[1]} - \text{XDB1,H1,H1,H1, Pin})
\]

\[
\text{linear} = \text{Gain}[1] + \text{XDB1,H1,H1,H1, Pin}
\]

\[
\text{m1} \quad \text{Pin} = 4.000000 \quad \text{m2} \quad \text{dBm} (\text{XDB1,H1,H1,H1, Vout[1]} = 22.815169
\]

<table>
<thead>
<tr>
<th>Pin</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.000</td>
<td>1.136</td>
</tr>
</tbody>
</table>
Gain Compression Overview

Gain compression is the difference, in dB, between a point on an idealized linear (small-signal) power-gain slope and a corresponding point on the actual power curve. That is, for a gain compression of $x$ dB,

$$-x = 10 \log_{10} \left( \frac{G_x}{G_{ss}} \right)$$

where $G_x$ is the point on the idealized linear gain slope that is $x$ dB directly below the point $G_{ss}$ on the small-signal power curve, as illustrated below.

The simulator stops its analysis when it reaches that point. The default setting is 1 dB.

The XDB simulator uses a harmonic balance algorithm, and as such shares many of the parameters and options that the Harmonic Balance simulator provides. However, this simulation requires that the input and output of the component or circuit whose gain compression is being simulated be defined by an appropriate source and termination.
Chapter 16: Circuit Envelope Simulation

The Envelope simulator component, in the Simulation-Circuit Envelope palette, simulates high-frequency amplifiers, mixers, oscillators, and subsystems that involve transient or modulated RF signals. You can simulate:

- Amplifier spectral regrowth and adjacent channel power leakage with digitally modulated RF signals at the input
- Oscillator turn-on transients and frequency output versus time in response to a transient control voltage
- PLL transient responses
- AGC and ALC transient responses
- Circuit effects on signals having transient amplitude, phase, or frequency modulation
- Amplifier harmonics in the time domain
- Subsystem analyses using modulation signals such as multilevel FSK, CDMA, or TDMA
- Efficient third-order-intercept (TOI) and higher-order intercept analyses of amplifiers and mixers
- Time-domain optimization of transient responses
- Intermodulation distortion (although the Harmonic Balance simulator, with the new Krylov option selected, may provide a faster solution in most cases)

Refer to the following topics for details on Circuit Envelope simulation:

- “Performing a Circuit Envelope Simulation” on page 16-2 describes the minimum setup requirements for a Circuit Envelope simulation.
- “Example” on page 16-3 is a detailed setup for calculating intermodulation distortion, using a Gilbert Cell mixer as the example. The location of the mixer example is also given.
- “Circuit Envelope Simulation Overview” on page 16-8 is a brief explanation of the Circuit Envelope simulator.
- “Troubleshooting a Simulation” on page 16-14 offers suggestions on how to improve a simulation.
Performing a Circuit Envelope Simulation

Note You must have the Circuit Envelope simulator license (included with the RFIC Pro and Premier suites, the RF Board Premier suite, or the Microwave Circuits Premier suite) to use the simulator.

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis, be sure to:

- Use either time domain or frequency domain sources in your circuit. In a circuit employing a mixer, provide a source for the LO.
- Add the Circuit Envelope controller to the schematic. (From the Component palette, choose Simulation-Envelope. Add the ENV component to the schematic.) Double-click to edit it. Fill in the fields under the Env Setup tab:
  - A Circuit Envelope simulation runs in the both the time and frequency domain. Set the stop time and time step (start time is 0). Time step defines the maximum allowed bandwidth (± 0.5/Time step) of the modulation envelope. The analysis bandwidth (1/Time step) should be at least twice as large as the modulation bandwidth to ensure accurate results at the maximum modulation frequencies.
  - Enter fundamental frequencies and order.
- You can perform small signal analysis. Enable Small-signal and fill in the fields under the Small-Sig tab.
- You can perform nonlinear noise analysis. Enable Nonlinear noise and fill in the fields under the Noise(1) and Noise(2) tabs.
- If your design includes an OscPort component, select the Env Params tab and fill in the Oscillator options.
- If your design includes NoiseCon components, select the NoiseCons tab and fill in the fields.
- You can use previous simulation solutions to speed the simulation process. For more information, refer to “Reusing Simulation Solutions” on page 7-17.
Note Unless there are convergence problems, it is recommended that you leave the other parameters under the Env Params and HB Params at their default settings.

• After the simulation is complete, in the Data Display, Envelope data variables are identified by the prefix ENV.

Example

Figure 16-1 illustrates an example setup for using the Envelope simulator to find mixer intermodulation distortion (IMD).

Note You must have the Circuit Envelope simulator license to simulate examples. You may build the Circuit Envelope example without this license, but will be unable to run the simulations.

Note This design, IMDRFSwpEnv.dsn, is in the Examples directory under RFIC/ Mixers_prj. The results are in IMDRFSwpEnv.dds.
Figure 16-1. Example Setup for Using the Envelope Simulator to Find Mixer IMD
In this example:

- An RF center frequency of 2000 MHz and an LO frequency of 1750 MHz have been established by a VarEqn component. The spacing between tones has been established by the equation $f_{\text{spacing}} = 100$ kHz.

- An $I_{\text{nTone}}$ source establishes two intermodulating RF frequencies by means of the following equations:
  
  $$F_{\text{req}[1]} = \text{RFfreq} - f_{\text{spacing}}/2 \text{ and } F_{\text{req}[2]} = \text{RFfreq} + f_{\text{spacing}}/2$$

- An $I_{\text{1Tone}}$ source establishes the LO frequency by means of $F_{\text{req}} = \text{LOfreq}$.

**Hint**

Using current sources instead of voltage sources leads to faster simulations, because one fewer equation per source is generated. The function $\text{dbmtoa}$ converts power to current at a default reference impedance of 50 ohms. $P_{\text{1Tone}}$ and $P_{\text{nTone}}$ components can also be used.

- A ParamSweep component establishes RF power as the parameter to be swept. This component also establishes the Start, Stop, and Step values for the power sweep.

- In the Envelope Simulation component, LOfreq and RFfreq have been assigned to $F_{\text{req}[1]}$ and $F_{\text{req}[2]}$, respectively.

- Stop time has been determined by $t_{\text{stop}}$, which in turn is defined by an equation in the VarEqn component. Similarly, Time step has been determined by $t_{\text{step}}$.

**Note**

Because this example will later use the $\text{fs}()$ function, the number of time points (determined by numpts=20 in the VarEqn component) must be even. numpts is the number of timepoints that are simulated per period of the modulation frequency. Modulation frequency is determined by $f_{\text{spacing}}/2$, and $f_{\text{spacing}}$ has been established as 100 kHz.

- Transient responses are discarded by extrapts, the number of extra points to simulate at the start. This is the same as the Sweep offset parameter (under the Env Params tab).

The following figure shows the results of the simulation.
IF spectral power is plotted against frequency in kHz, by means of the equation

\[ \text{IF spectrum} = \text{dBm}(\text{fs}(\text{Vif}[1])) \]

**Note**  The function \( \text{fs} \) performs a time-to-frequency transform, transforming the IF \( \text{Vif}[1] \) into the frequency domain.

The value zero on the x-axis represents the IF, with values to the left and right representing the mixing products that are offset from the IF. The marker M2, at +150 kHz, indicates one of the third-order IMD products. The equation TOI output uses simple geometry.
More Examples

For more Circuit Envelope simulations, refer to these example projects:

- For ways of generating sources for use in envelope simulations (such as $\pi/4$-DQPSK, FSK, QAM, and CDMA), see Tutorial/ ModSources_prj.

- To simulate amplifier spectral regrowth and adjacent channel power leakage with digitally modulated RF signals at the input, see RF_Board/ NADC_PA_prj.

- To simulate PLL transient responses, see RF_Board/ PLL_5th_Order_prj and DECT_LO_Synth_prj.
Circuit Envelope Simulation Overview

The Envelope simulator combines features of time- and frequency-domain representation, offering a fast and complete analysis of complex signals such as digitally modulated RF signals. Briefly, this simulator permits input waveforms to be represented in the frequency domain as RF carriers, with modulation “envelopes” that are represented in the time domain (Figure 16-2).

More specifically, in an Envelope simulation each node voltage is represented by a discrete spectrum having time-varying Fourier coefficients. The set of spectral frequencies is user-defined; the amplitude and phase at each spectral frequency can vary with time, so the signal representing the harmonic is no longer limited to a constant, as it is with harmonic balance. Each spectral frequency can be thought of as the center frequency of a spectrum; the width of each spectrum is ±0.5/Time step. Figure 16-3 illustrates this, where the minimum envelope bandwidth is equal to the bandwidth of the modulation signal. In most cases the bandwidth of the modulation signal is much smaller than the lowest user-defined spectral frequency (which corresponds to the “carrier” frequency), unlike what is shown in the figure.
The bandlimited signal within each spectrum can contain periodic, transient, or random tones. The actual time-domain waveform is represented as a sum of carriers (with harmonics and intermodulation products), where each envelope can vary with time:

\[ v(t) = \text{real} \left[ \sum_{k=0}^{N} V_k(t) e^{j2\pi f_k t} \right] \]

Here, \( v(t) \) is a voltage at any node in the circuit, including the input. The Fourier coefficients, \( V_k(t) \), are allowed to vary with time and may represent an arbitrary modulation of each carrier. Since each time-varying spectrum \( V_k(t) \) can be thought of as a modulation waveform with a center frequency \( f_k \), these are often referred to as “envelopes.” This spectrum may represent transient signals with continuous spectra, such as a digital modulation envelope over an RF carrier, or periodic signals with discrete spectral lines, such as the two RF tones required for intermodulation distortion analysis.

Figure 16-4 illustrates a modulated signal and the time-varying spectrum that results from the simulation. Any spectral component obtained from the simulation can be processed and displayed in amplitude, phase, I (the in-phase modulation component), or Q (the quadrature modulation component). By computing the Fourier transform of the spectral component, the simulator can present the spectrum around the component, as in a spectrum analyzer display.
This simulator does not require that a spectral component be present at the center frequency. Table 16-1 gives examples of how the spectrum can be defined. You can use a V_Tone source component to generate the various signals depicted in the table, assigning to the source parameter V the various expressions for \( V_k \). In a given envelope simulation, there are \( N + 1 \) possible spectral components. The one at DC (also referred to as the baseband component) is limited to a bandwidth of 0.5/Time step, and only the real portion of \( V_k \) is used. The other \( N \) spectra have a double-sided bandwidth of 1/Time step and are usually complex, as shown in Figure 16-3.

The envelope waveform \( V_k(t) \) has many useful properties. For example, to find the instantaneous amplitude of the signal around \( f_k \) at time \( t_s \), simply compute the magnitude of the complex number \( V_k(t_s) \). Similarly, the phase, real, and imaginary values of instantaneous modulation can be extracted by simply computing the phase, real, and imaginary values of \( V_k(t_s) \).

Note This process only extracts the magnitude of the modulation around \( f_k \). It does not include any of the spectral components of adjacent \( f_{k-1} \) or \( f_{k+1} \) spectra, even if these spectra actually overlap. If this \( f_k \) spectrum has multiple tones inside of it, then this demodulation does include their effects.
This simple technique does not allow the demodulation of only one of the tones inside this $f_k$ spectrum and the exclusion of the others. To demodulate only one tone, first select the desired tone by using an appropriate filter in the circuit to be simulated. Also, note that because the baseband (DC) spectrum represents a real signal and not a complex envelope, its magnitude corresponds to taking the absolute value of that signal, and its phase is either 0 or 180 degrees.

<table>
<thead>
<tr>
<th>#</th>
<th>Formula that specifies the envelope</th>
<th>Description of the signal in the time domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$V_k=1$</td>
<td>Pure cosine: $\cos(2\pi f_k \text{time})$</td>
</tr>
<tr>
<td>2</td>
<td>$V_k=\exp(j\pi/2)$ or polar($1.90$) or $1^j$</td>
<td>Pure sine: $\sin(2\pi f_k \text{time})$</td>
</tr>
<tr>
<td>3</td>
<td>$V_k=A \exp(j^2 \pi f_m \text{time}+B)$</td>
<td>One tone (SSB) $A \cos(2\pi f_k(f_k+f_m) \text{time}+B)$</td>
</tr>
<tr>
<td>4</td>
<td>$V_k=A \exp(j^B)$; freq=$1.1$ GHz</td>
<td>Same as (3) (assuming $f_k + f_m = 1.1$ GHz)</td>
</tr>
<tr>
<td>5</td>
<td>$V_k=2\cos(2\pi f_m \text{time})$</td>
<td>Two tones (AM suppressed carrier)</td>
</tr>
<tr>
<td>6</td>
<td>$V_k=\exp(j^2 \pi f_m \text{time}) + \exp(-j^2 \pi f_m \text{time})$</td>
<td>Same as (5)</td>
</tr>
<tr>
<td>7</td>
<td>$V_k=$pulse(time,...); freq=$f_k+f_m$</td>
<td>Pulsed RF at a frequency of $f_k+f_m$</td>
</tr>
<tr>
<td>8</td>
<td>$V_k= -\text{step(time – delay)}$</td>
<td>A negative cosine wave, gated on at t=delay</td>
</tr>
<tr>
<td>9</td>
<td>$V_k= (\text{vreal}(time)+j\text{vimag}(time)) \exp(j^2 \pi f_m \text{time})$</td>
<td>I/Q modulated source centered at $f_k+f_m$ (vreal(), vimag() user-defined functions)</td>
</tr>
<tr>
<td>10</td>
<td>$V_k=(1 + vr1) \exp(j^2 \pi vr2)$</td>
<td>Amplitude- and noise-modulated source at $f_k$. (vr1, vr2 are user-defined randtime functions, not available in Release 1.0)</td>
</tr>
<tr>
<td>11</td>
<td>$V_k=\exp(j^2 \pi (-f_0 + a_0 \text{time}/2)\text{time})$</td>
<td>Chirped FM signal starting at $f_k - f_0$, rate $= a_0$</td>
</tr>
</tbody>
</table>
Circuit Envelope Simulation Overview

Circuit Envelope and Frequency-Domain-Defined Devices (FDDs)
As the applications for wireless communications continue to grow, it is no longer possible to satisfy all modeling needs with standard, preconfigured models. Users need methods to define their own nonlinear models in either the time or the frequency domain. The frequency-domain-defined device (FDD) has been developed to allow a user to describe current and voltage spectral values directly, in terms of the algebraic relationships of other voltage and current spectral values.

Another trend in digital communication systems is the issue of timing, since it is increasingly common to encounter subsystems that behave in ways that cannot be modeled as time-invariant. Clocked systems, sampled systems, TDMA pulsed systems, and digitally controlled systems are all becoming more common, even in the RF and microwave area, and behavioral models must be able to include these effects. So, in addition to its frequency-domain modeling attributes, the FDD also allows the modeler to define trigger events, to sample voltages and currents at trigger events, and to generate outputs that are arbitrary functions of either the time of the trigger or of the complex spectral voltage and current values at these trigger events.

Circuit Envelope and Components
In general, any of the components available in Advanced Design System can be used in circuits where envelope simulations are performed. Some components lend themselves especially well to such circuit designs, such as n-state modulators and demodulators or spectral waveform sources.
Using Functions and Equations

Functions and equations can be placed in a schematic in the same manner as ADS components, and some are especially well-suited to envelope simulations and behavioral model designs. There is also a set of functions that can be used only with FDDs, and a set of variables that can be used only with SDDs; they, too, can facilitate digital communication designs where the envelope simulator is applied.

To use the Variables and equations component:

1. From the **Data Items** palette, select **Var eqn** (Variables and equations) and place it in the Schematic window.

2. Edit it and select the **File Based** option from the Variable or Equation Entry Mode drop-down list.

3. In the Data Access Component field, type the Instance Name—or select it from the drop-down list—of the DAC that points to the file containing the data of interest.

   This makes it possible to use lists of data as component parameter values. Such data can also be used with n-state components to define the phase and amplitude of the various states. These components are especially efficient in envelope simulations, and dataset and list arrays simplify the entering of strings of data as parameter values.
Troubleshooting a Simulation

This section presents suggestions for using this simulator and improving the accuracy of results.

An Easy Way to Increase Accuracy

In general, accuracy is increased and aliasing reduced when the signal of greatest amplitude is entered as the Freq[1] parameter, and lower-amplitude signals are entered as Freq[2], Freq[3], and so on.

Improving Accuracy and Speed in Mixer IMD Measurements

Prior to the invention of the Krylov subspace solver (available in harmonic balance simulation), Circuit Envelope was the recommended choice for mixer IMD (intermodulation distortion) measurements. Now the Krylov option appears to provide faster solutions for this application. Nevertheless, when the envelope simulator is used, there are several ways to improve accuracy and speed. Also, Maximum order can have a profound effect on simulation results; in general, set Maximum order to a low value and increase it until the simulation produces marginal differences in results.

Using the Circuit Envelope Simulator to Analyze an Oscillator

This simulator can support the simulation of oscillator circuits and subsystems, through the use of the Oscillator Options area under the Env Params tab. The oscillator options work as follows:

- If Use OscPort is disabled, the envelope simulation proceeds as normal and any oscillations, if present, will build up from some initial value. The analysis frequencies and time step must be set to ensure that the oscillator’s instantaneous frequency is well within one of the envelope bandwidths being simulated. This can be either the baseband (DC) envelope or any of the fundamental, harmonic, or mixing-term analysis-frequency envelopes. Multiple oscillations can be simulated as well, as long as each one is within an analysis envelope.

- When the transient buildup is being simulated, a tickler voltage is usually needed in the same envelope as the actual oscillator frequency, to define an initial start-up value. This value can be set by a low-level transient source that
is disabled after the first few time samples. Alternatively, a noise source or resistor noise can also be used. For the oscillator control elements, the option Turn on all noise is enabled. This will cause devices with noise in the oscillator loop to generate the white noise that will provide the required tickler stimulus.

- If Use OscPort is enabled and a name is supplied in the OscPort name field, then a normal harmonic-balance oscillator simulation is automatically performed prior to the first time step (refer to Chapter 7, Harmonic Balance Simulation). The results of this simulation determine and set the analysis frequency value to the steady-state oscillator frequency. If the oscillation frequency does not change significantly during its buildup, this often allows larger time steps to be used. This is because the envelope bandwidth does not have to include any uncertainty as to the oscillator’s final frequency value.

- If Use OscPort is enabled, you may reset the oscillator voltage solution to zero by selecting Calculate oscillator startup transient, so that the transient buildup can be simulated. If this option is not enabled, the time-domain solution begins at the steady-state solution, the transient buildup time is skipped, and the oscillator can immediately start responding properly to any external modulation. Note that Use OscPort affects only the initial simulation, and is disabled once the envelope simulation begins.

Using the Circuit Envelope Simulator to Analyze Noise

Noise in envelope simulations can be added as random time voltages and currents. Noise can originate from circuit components such as resistors or noise sources. The full, nonlinear circuit equations are applied to this composite signal of random voltages and currents, so no small-signal assumptions about the relative size of the noise are required.

Setting the option Turn on all noise (under the Env Params tab) enables white noise from many devices in the circuit to be included in the simulation. The noise sources that are included in the circuit are also included in the simulation. They add white, Gaussian noise at all frequencies in the analysis envelope. The noise voltages will be complex for non-baseband envelope frequencies, generating both amplitude and phase equivalent noise.

Convolution Techniques Used in Circuit Envelope

In Circuit Envelope, the response within the envelope bandwidth (1/timestep) around each of the Harmonic Balance frequencies (including DC) needs to be represented in
Circuit Envelope Simulation

the time domain. The advantage here over a transient simulation is that the entire frequency response need not be represented in the time domain, and if the envelope bandwidth is relatively small, then a simple fit or a constant is sufficient to represent the response. And since a different constant could be used for each carrier, it maintains good accuracy at the exact Harmonic Balance frequencies.

A consequence of this method is that any components which are characterized in the frequency domain need to have their response converted to the time domain for the portions of their frequency response which fall within the envelope bandwidth.

By default, the Circuit Envelope simulator uses recursive convolution for this task, which first tests to see if the frequency response is described by a rational polynomial. If it is not, the simulator then tries to fit the frequency response to a rational polynomial (with no delay) around each carrier. If this fails, a poor fit warning is generated and the simulation will either proceed with the best fit or just use the center frequency constant, depending on the setting in the analysis control.

Impulse convolution can be used, but it is currently only available for certain components such as S1P_Eqn, Y1P_Eqn, or Z1P_Eqn. These components have an ImpDeltaFreq parameter, and an appropriate value must be entered for the parameter to use impulse convolution. However, recursive convolution has priority over impulse convolution, even for such components, if the frequency response is described by a rational polynomial.
Chapter 17: Transient and Convolution Simulation

The Transient/Convolution Simulation component, in the Simulation-Transient palette, enables you to:

- Perform a SPICE-type transient time-domain analysis on a circuit.
- Perform nonlinear transient analysis on circuits that include the frequency-dependent loss and dispersion effects of linear models. Such analyses are known as convolution analyses.

Refer to these topics:

- “Performing a Transient or Convolution Simulation” on page 17-2 has the minimum setup requirements for a transient simulation.
- “Example” on page 17-3 is a detailed setup for performing a transient simulation, using a Gilbert cell mixer as the example.
- “Transient/Convolution Simulation Overview” on page 17-7 is a brief explanation of transient and convolution simulations.
- “Troubleshooting a Simulation” on page 17-12 offers suggestions on how to improve a simulation.

Note: You must have the Transient simulator (included with the RFIC Designer Pro and Premier suites) and the Convolution simulator (included with the RFIC Designer Premier suite only) licenses to run these types of simulations. You may build the examples without the appropriate license, but cannot run the simulations.
Performing a Transient or Convolution Simulation

Start by creating your design, then add current probes and identify the nodes from which you want to collect data.

For a successful analysis:

- When selecting sources, you can use either frequency-domain or time-domain sources. Transient sources are under the Sources-Time Domain palette. They are identified by the small t in their names (for example, VtStep:Voltage Source: Step).

- Add the Trans component to the schematic. Double-click to edit it. Fill in the fields under the Time Setup tab:
  - Enter the start and stop times.
  - Enter the Max time step. This is the largest time step that will be used in the simulation. It should be small enough to adequately sample the highest frequency expected in the circuit.

- To achieve the most accurate model, select the Convolution tab and set the Max Frequency and max impulse sample points. The other parameters here are related to generating impulse responses for convolution analysis, but in general accept the defaults. For more information, refer to the sections “Convolution Analysis” on page 17-7 and “Setting Max Frequency and Other Convolution Parameters” on page 17-14.

- If frequency is not defined by a frequency source, select the Freq tab and set the fundamentals and order.

- The parameters under the Integration tab set truncation, integration techniques, and charge accuracy. For information on integration techniques, refer to “Integration Methods Used in Transient/Convolution Simulation” on page 17-10.

- The parameters under the Convergence tab are used to improve convergence. For more information, refer to “Solving Convergence Problems” on page 17-13.

- You can use previous simulation solutions to speed the simulation process. Refer to “Reusing Transient Simulation Solutions” on page 17-11.

- It is recommended that parameters not specifically mentioned here be left at the default values. For more information about each parameter, click Help in the Trans dialog box.
Example

Figure 17-1 illustrates the setup for a basic transient/convolution simulation.

Note  This design, TRAN1.dsn, is in the Examples directory under Tutorial/SimModels_prj. The results are in TRAN1.dds.

Figure 17-1. Setup for Transient/Convolution Simulation
Transient and Convolution Simulation

To perform a basic transient/convolution simulation:

1. From the Component Palette, choose **Sources-Time Domain** or **Sources-Freq Domain** and select appropriate sources. Place the source at the input of the component or circuit under test, then define input power and edit other parameters as required. In a circuit employing a mixer, provide a source for the local oscillator (LO).

   **Note** Transient sources, available in the Time Domain Sources palette, are available for these simulations, and are identified by the small “t” in their names (for example, VtStep:Voltage Source: Step). However, standard frequency sources (in the Freq Domain Source palette) can also be used.

2. Ensure that the inputs and outputs of nodes at which you want data to be reported are appropriately labeled.

3. If they are needed, select from among various transient measurement equations in the Transient Simulation palette, place these on the schematic, and edit them as required. (Refer to “Troubleshooting a Simulation” on page 4-9.) Their results can be plotted later in a Data Display window.

4. From the Component Palette, choose **Simulation-Transient**. Select and place the Tran simulation component on the schematic. You can run a basic simulation by editing the parameters **StopTime** and **MaxTimeStep** on the schematic.

   **Note** The Max time step value should be small enough to adequately sample the highest frequency expected in the circuit. The simulator may use a smaller timestep if needed but will never use a larger value.

5. To edit the frequency of the fundamental and any other frequencies to be considered, select the Freq tab. This frequency information is required only if the frequency is not set explicitly in the frequency domain sources.

6. The parameters that make it possible to obtain the most accurate model are **Max Frequency** and **Max impulse sample points**, under the Convolution tab of the simulation component. It is recommended that you leave the other parameters under this tab at their default settings and edit them only in special cases.
7. For details regarding all Transient parameters, double-click the simulation component to edit it, select the tab of interest, and click the Help button. Many parameters in this simulator apply only to special cases.

8. Launch the simulation. The data resulting from this simulation will be identified by TRAN1. The large trace is a plot of $V_{if}$ versus time, the output of the filter showing all harmonic components. The small sinusoid is a plot of $V_{load}$ following the filter. The filter output has been given enough time to approach a steady-state amplitude.

9. A plot of $fs(V_{if}, 4\text{ns}, 16\text{ns})$ shows a time-to-frequency transform from 4 to 16 ns for the mixer output before filtering. (The seven commas represent $fs$ parameters not used here.) Run the simulation longer to observe the noise floor drop.
10. Finally, an fs plot of the filter output (Vload) between 24 and 32 ns shows the response after steady-state amplitude has been approximated. Essentially only the bandpass frequency remains.
Transient/Convolution Simulation Overview

The transient and convolution simulators are SPICE-like in their operation. They solve a set of integro-differential equations that express the time dependence of the currents and voltages of the circuit under analysis. The result is a nonlinear analysis with respect to time and, possibly, a swept variable.

The main difference between the transient and convolution options lies in how each analysis characterizes the distributed and frequency-dependent elements of a circuit, as discussed below.

Transient Analysis

A transient analysis is performed entirely in the time-domain, and so is unable to account for the frequency-dependent behavior of distributed elements such as microstrip elements, S-parameter elements, and so on. Therefore, in a transient analysis, such elements must be represented by simplified, frequency-independent models such as lumped equivalents, transmission lines with constant loss and no dispersion, short circuits, open circuits, and the like. These assumptions and simplifications are usually very reasonable at low frequencies.

Convolution Analysis

A convolution analysis represents all the distributed elements in the frequency domain and hence accounts for their frequency-dependent behavior. The characterization of many RF and microwave distributed elements is best accomplished in the frequency domain, because the exact time-domain equivalents for these elements cannot always be obtained.

Convolution converts the frequency-domain information from all the distributed elements to the time domain, effectively resulting in the impulse response of those elements. The time-domain input signals at an element’s terminals are convolved with the impulse-response of the element to yield the output signals. Elements that have exact lumped equivalent models—including nonlinear elements—are characterized entirely in the time domain without using impulse responses.
 transient/Convolution Simulation Overview

**Note** In a convolution analysis, all elements are characterized by means of the full frequency-domain model, through the use of either an exact time-domain model or convolution. However, there may be minor differences between the results of a convolution simulation and the results of a transient simulation of the same circuit.

A convolution analysis requires both a convolution license and a transient license, and is performed whenever a convolution license is available. If the simplified approximate models are preferred, in this situation (for speed), set the Use Approximate Models When Available option to yes.

**Transient/Convolution Simulation Process**

The following steps describe how both the transient and convolution simulators operate:

1. The user specifies a time-sweep range, tolerances, and iteration limits.
2. A DC analysis is conducted to determine the system solution at zero time.
3. Inside the simulator, a breakpoint table is constructed to deal with frequency-domain-devices and data. Independent source waveforms frequently have sharp transitions that may not normally coincide with the time step calculated by the program. Such is the case with the piecewise linear sources. The breakpoint table contains a sorted list of all the transition points of the independent sources. During the simulation, whenever the next time point is sufficiently close to one of the breakpoints, the time step is adjusted to land exactly on the breakpoint. This prevents unnecessary time-step reductions in the vicinity of the transitions.
4. An internal control variable updates the current time, and the values of the independent sources are calculated at that time.
5. An attempt is made to solve the system of equations through numerical integration and a finite number of Newton-Raphson iterations. If the number of iterations exceeds Max iterations per time point, then the time step is reduced by a factor of Integration coefficient mu divided by 8. If this new time step is acceptable, the analysis is repeated from step 4. If Integration coefficient mu = 0, backward-Euler numerical integration is used. Otherwise, trapezoidal numerical integration is used.
6. Following convergence, the local truncation error is calculated. The default Trapezoidal integration method is used to estimate the error, unless Gear’s method is selected.

7. The time step interval is calculated. By default, the time step is computed for transient analysis by means of the truncation error estimate method.

8. The error tolerance is compared with the value in the Local truncation error over-est factor field (under the Integration tab). If the error is within acceptable limits, the results are stored and analysis continues at the next time point. Otherwise, the analysis is repeated at a smaller time step.

9. Steps 3 through 9 are repeated until the user-specified time-sweep range has been analyzed.
Integration Methods Used in Transient/Convolution Simulation

Like SPICE, this simulator uses the trapezoidal integration method described by the following equation as the default method for calculating derivatives at each time step in the simulation.

\[ x_{n+1} = \frac{2}{\Delta t} (x_{n+1} - x_n) - x'_n \]

For most circuits, this method will succeed. For those that do not, the simulator also supports Gear’s backward difference method:

\[ x_{n+1} = \alpha_0 x_{n+1} + \alpha_1 x_n + \alpha_2 x_{n-1} + \ldots + \alpha_k x_{n+1-k} \]

In this equation, the index \( k \) is called the order of the integration.

For most circuits, Gear’s method is no more accurate than the default trapezoidal integration technique. However, if a circuit analysis fails to converge, Gear’s method may succeed where trapezoidal integration fails. In particular, oscillator circuits and any circuit that is characterized by stiff state equations may benefit from Gear’s method.


If Time Step Control is set to TruncError and Max Gear order (under the Integration tab) is set to a number between two and six, the simulator will use Gear’s method along with an adaptive stepsize algorithm that picks the largest possible step size at each point in the simulation. For each time step, the order of Gear’s method will be chosen (up to the value of Max Gear order) to maintain accuracy with the largest possible time step. This potentially speeds up simulations with no loss in accuracy. If Gear Integration is selected with fixed timestep, then the integration will always be done at the fixed order given by Max Gear order.

The integration order at each time step is output to the dataset as the variable tranorder. This data is used by the fs() function, in data display, to do accurate interpolation of the data when an FFT is required. For the default trapezoidal integration, this will normally have a value of two, except at source-induced breakpoints where it will be one.
Reusing Transient Simulation Solutions

Transient simulations can be set to generate a harmonic balance solution that can then be used as an initial guess for a harmonic balance simulation. For example, in circuits such as dividers, harmonic balance cannot usually directly converge on a solution since multiple mathematical, but useless, solutions exist. By first running a Transient simulation and generating a harmonic balance solution file and then using this as an initial guess for a harmonic balance simulation, the harmonic balance simulation can then converge to the desired solution.

To save a solution to a file:

1. Add a transient component to your schematic if one is not present, and double-click to edit it.

2. Select the **Freq** tab. Fill in the frequency fields as you would for a harmonic balance simulation. The Frequency values can still be used, independent of this solution mode, to define the fundamental frequencies and _freqN variables used in sources. The Order and Maximum order information is used to determine the number of frequencies for which to compute a harmonic balance solution.

3. Enable **Write Solution** and enter a file name and extension. The default is the name of the project.

4. For best results, especially in multi-tone applications, enable **Apply Window**. This applies a Hanning window to the time domain data. For a good Fourier transform, the duration of the transient simulation must be longer than the period of the smallest specified frequency. A factor of 4 is often used here. It is also best to make sure that the transient simulation has reached an approximately steady state solution. This does not have to be precise, since this solution is only providing an initial guess value to the harmonic balance simulation. The start time of the transient simulation can be used to indicate when this approximate steady state solution has been reached. The internal Fourier transform will then not begin until after this start time.

5. To get the best results for the solution, the transient analysis should be run in a fixed timestep control mode.
Monte-Carlo Noise in the Time Domain

Noise in transient simulations can be added as pseudo-random voltages and currents at each timepoint. The noise has a Gaussian amplitude distribution. In the current release of ADS, noise can only originate from the V_Noise and I_Noise voltage and current noise sources. The noise is added to the signal, so there is no easy way to separate the signal from the noise. The full, nonlinear circuit equations are applied to this composite signal of random voltages and currents, so no small-signal assumptions about the relative size of the noise are required.

The Noise bandwidth parameter (under the Transient Noise tab) enables the generation of noise in transient analysis. Because transient analysis uses a variable timestep method, the noise must be limited to a bandwidth of less than $1/(2 \times \text{MaxTimeStep})$. If this parameter is left blank or specified as zero, no noise is generated.

The Noise scale parameter is used to multiplicatively scale all of the noise sources in the circuit. This can be useful when the noise levels are very low, as this allows them to be increased so they are visible above the numerical noise in the simulation. However, if the noise is increased too much, it can change the nonlinear operation of the circuit.

Troubleshooting a Simulation

This section presents suggestions for using this simulation tool and improving the accuracy of results.

Avoiding Simulation Errors in Transient Analysis

This section lists a variety of steps that can be taken to avoid errors in simulation.

1. Check the circuit’s schematic diagram carefully, and turn on the topology checker if it has been turned off. Consider using DC_Block and DC_Feed components where applicable.

2. Check the parameter Min time step. This parameter sets the smallest time step that the simulator is allowed to take, and should be smaller than the fastest rise time in the circuit. The default value of Min time step is $\text{Max time step}/10^{12}$. (Stop time is the last time point in the simulation.) The default value should be satisfactory.
3. Check to verify that the absolute and relative current and voltage tolerances (in the Options component) are not too small.

4. The models for some frequency-dependent devices have a high-frequency limit, beyond which they are not valid. Unless the very-high-frequency response of the device is important, the simulation results will still be valid.

5. If a lossy inductor model is included in the circuit, and the inductance has been set to zero, you may need to replace the lossy inductor with a resistor.

6. A lossy inductor model cannot be used with an initial condition. To solve this problem, replace the lossy inductor model with a lossless inductor in series with a resistor.

7. The simulator supports user-defined models that can have any impedance. However, it is easy for users to define nonphysical or noncausal components for which there is no correct answer. If a component has a constant reactance that does not vary with frequency (or has a nonzero reactance at DC), then the component is mathematically nonphysical. In these cases, the simulator will produce an answer that may not be physically realistic. To eliminate this problem, change the component’s definition.

8. Sometimes, in the case of user-defined devices, the simulator cannot handle certain types of time dependencies with guaranteed accuracy. These devices often work correctly, but the simulation results should be checked carefully.

9. The simulator cannot support S-parameter ports with zero impedances. To use a source with a zero impedance, use a simple voltage source instead.

**Solving Convergence Problems**

Nonconvergence is a numerical problem encountered by the simulator when it cannot reach a solution, within a given tolerance, after a given number of iterations. There is no single solution for solving convergence problems in transient and convolution analysis. Several ways to approach those problems are listed below.

- Make sure that you have specified an appropriate value for Max time step.
- Vary Integration coefficient mu in the case of high-Q circuits.
- Increase the time-point iteration limit, Max iterations per time step.
- If the circuit includes a GaAsFET, reduce the value of the transit time TAU in the device model itself.
• Adjust the current and voltage relative and absolute tolerances in the Options component. Note the following guidelines:

  The relative tolerance parameters have a greater effect on simulation speed and accuracy than the absolute tolerance parameters.

  Each order-of-magnitude change in accuracy (for example, from $10^{-3}$ to $10^{-4}$) will result in approximately three times as many time points in the simulation.

**Comparing Time-Domain and Steady-State Results**

Certain circuit elements such as microstrip, discontinuities, and so on, are represented by simplified models in a transient analysis. A few planar discontinuities are also treated as ideal short or open circuits. Therefore, results from transient or convolution simulation may differ from those for steady-state simulation on the same circuit, depending upon the types of elements used in the circuit. (A convolution simulation should yield results closer to those of a steady-state simulation.)

The frequency of operation also plays a major role. At low frequencies, simplified models or short-circuits may be valid approximations for certain dispersive models.

**Setting Max Frequency and Other Convolution Parameters**

In general, the synthesized impulse response accurately represents the frequency domain function over the frequency range specified by the convolution control parameter Max Frequency. The simulation techniques require negligible spectral energy outside this frequency range. An accurate solution is not guaranteed if this principle is not obeyed. Therefore, setting Max Frequency correctly is the most important aspect of any transient simulation using convolution-based devices.

Setting Max Frequency is similar to choosing the number of harmonics in a harmonic balance simulation or a time step in SPICE. In these cases the user must estimate the value of the parameter prior to the simulation. Examination of the results reveals whether the parameter was chosen correctly. The built-in estimation of Max Frequency is always a good starting point.

The remaining convolution-control parameters described below are best used with their default values for any causal device, such as a microstrip transmission line.

The Discrete convolution mode option, by causing a periodic extension of the frequency response, generally leads to the most accurate and efficient description of the device from DC to Max Frequency. Provided Max Frequency is set correctly, the
frequency response beyond Max Frequency is irrelevant, because no significant spectral energy exists beyond this value. The PWL Continuous option always leads to a low-pass response (which may be desirable in cases where a low-pass response is being modeled). The discrete mode is many times faster than the continuous mode.

**Delta frequency** defines the frequency spacing with which the convolution-based devices are sampled in the frequency domain. If Delta frequency is not specified, the simulator adaptively samples the frequency function until an appropriate value is determined. If you are unsure about the correct value of Delta frequency, simply allow the simulator to decide.

**Non-causal fcn imp response length** adjusts the length of the impulse response associated with the treatment of noncausal frequency responses (see discussion below).

**Smoothing window type** specifies the smoothing window to be applied to the time-domain impulse responses that are derived from noncausal frequency functions (such as Hilbert transforms). The window reduces ripple in the approximation caused by discontinuities in the frequency function. (Refer to “Dealing with Noncausal Frequency Responses” on page 17-16.)

**Max impulse sample points** places an upper limit on the allowed impulse-response length. It is mainly used when Max Frequency is specified but Delta Frequency is not. It is necessary to increase this value (default = 4096) if you specify a frequency response that requires a long impulse response.

**Relative impulse truncation factor** and **Absolute impulse truncation factor** are used to remove redundancy from the impulse responses. Setting them to a small value leads to longer impulse responses and a more accurate description of the frequency response of a convolution-based device. The simulator uses them to make decisions about the relative sizes of individual members of an impulse response.
Dealing with Noncausal Frequency Responses

A causal system has the property that if the input is zero for \( t < t_1 \), then the output is also zero for \( t < t_1 \). The value of \( t_1 \) is usually defined to be 0. Therefore, the impulse response of a causal system is zero for negative time. This may be stated in the frequency domain by saying that the real and imaginary parts of a causal frequency response are related by the Hilbert transform. Only causal frequency-domain transfer functions can be handled directly by means of transient simulation methods.

However, there are many ideal frequency functions that are noncausal but are extremely valuable in processing signals (such as Hilbert transforms). The transient simulator provides a way to use these functions in defining the frequency response of a nonlinear device.

A frequency response is considered noncausal if it has one of the following forms:

\[
R + jI \\
R(w) + j0 \\
R + jI(w)
\]

When you select Non-causal fcn imp response length, non-causal frequency functions are approximated by a digital filter that introduces a time delay in the impulse response. This time delay is sufficient to make the apparent impulse response causal. Non-causal fcn imp response length defines the length of this filter. The Smoothing window type can be selected to reduce Gibb's phenomenon, which may be present if the non-causal frequency response contains discontinuities. Increasing non-causal length to a more accurate description of the frequency function over the chosen band set by Max Frequency. However, increasing non-causal causes a longer delay through the filter and a longer simulation time. Only even values of non-causal are allowed. If an odd value is specified, the simulator adds 1 and issues a warning. The delay associated with the causal approximation is given by the equation:

\[
\frac{\text{noncausal}/2-1}{2\times\text{max Frequency}}.
\]

If a non-causal frequency response is specified which does not fit into one of the forms described above, the simulator will assume it is a causal function. This will almost certainly cause unexpected and erroneous results. Noncausal frequency responses should be avoided if possible.

The concept of causality is not an issue when using the harmonic balance simulator. All excitations and responses are sums of sinusoids and exist for all time. For this reason, any bounded frequency response can be simulated. When moving from harmonic balance to transient simulation, it must be remembered that non-causal
frequency functions cannot be simulated directly. Only a band-limited approximation to these functions can be simulated. This is particularly important when using SDD weighting functions.

Using Measured and Simulated S-Parameter Data

The ability to handle convolution based devices allows the user of measured or simulated S-parameter data to describe a wide variety of devices and circuits. A dataset or file containing the S-parameter values can be used to integrate the frequency response into a time-domain simulation. This adds a tremendous amount of flexibility to the number and types of devices and circuits which can be used in a simulation.

When S-parameter data is used, it is important that the frequency response be adequately sampled over the entire bandwidth to ensure negligible interpolation errors when the impulse response is being calculated. The Max Frequency parameter should never be set to a value which is greater than the maximum S-parameter data frequency. Doing so will lead to erroneous results as the available data would have to be extrapolated. S-parameter data must also extend all the way down to DC.
Transient and Convolution Simulation
Chapter 18: Using Circuit Simulators for RF System Analysis

The steady-state simulation of RF/IF subsystems in the frequency domain is achieved in Advanced Design System through the use of various circuit and system simulation components, as well as through a variety of measurement functions that can be applied to simulation data. Budget, spur (spurious signal), noise, and group delay data are typical objectives obtainable with these components, and a sweep analysis is a common way to obtain system response at a variety of frequencies, power levels, and other operating conditions.

For details refer to the following sections within this chapter, including a variety of examples:

- “Applicable Simulation Components” on page 18-2 lists the simulators that are best suited for RF system analysis.
- “Applicable Measurements” on page 18-4 lists the types of measurements that work well with RF system analysis.
- “Fundamentals of Using Circuit Simulation Components for System Analysis” on page 18-5 illustrates a typical RF system setup that can be used with circuit simulation.
- “Budget Analysis” on page 18-8 gives an overview of budget analysis plus several examples.
- “Using MixerIMT/MixerIMT2 Models in Spurious Signal Analysis” on page 18-26 describes how to use the Mixer IMT model for generating spurious signals and the approach to simulating spurious signals.
- “System Noise Analysis” on page 18-30 is an overview of contributors to system noise and how they are treated in a simulation.

Note The appropriate simulator licenses are required to run simulations examples, i.e. a Harmonic Balance simulation requires the Harmonic Balance simulator license (included in all Circuit Design suites except the RF Designer suite.) You may build the examples without the appropriate license, but will simply be unable to run the simulations.
Applicable Simulation Components

The circuit simulation components that can be used for system analysis are the AC, S-Parameter, Harmonic Balance, LSSP (including the use of the P2D Simulation component to generate power-dependent S-parameters), and XDB components. This chapter assumes that you are familiar with how to use those simulators to analyze circuits and display data. This section presents just a few of the RF system objectives that can be obtained.

By selecting the Harmonic Balance Simulation (HB) component in the Simulation-HB palette, you can achieve the following:

- Perform a budget analysis to determine the signal and noise performance for elements in an RF system network. This includes measuring system performance at an element’s input or output, and therefore finding the degree to which an element contributes to the degradation of system performance.
- Perform a sweep analysis to determine the network’s port-to-port performance with respect to a swept parameter such as frequency or power.
- Perform a spurious-signal analysis to determine the network’s spurious spectral tones, where all intermodulation products are due not only to mixer signal input and local oscillator mixing, but also to the nonlinearities of amplifiers.
- Use the Perform Budget simulation option to obtain currents and voltages at named nodes throughout the system. Use budget measurements to postprocess the resulting data.

By selecting the AC Simulation component (AC) in the Simulation-AC palette, you can achieve the following:

- Use the Enable AC Frequency Conversion option to do a small-signal analysis in systems containing freq.
- Perform a small-signal noise analysis at the IF at a variety of nodes.
- Use the Perform Budget simulation option to obtain currents and voltages at all pins of each element in the system. Use budget measurements to postprocess the resulting data.
By selecting the S_Param (SP) component, in the Simulation-S_Param palette, you can achieve the following:

- Determine linear scattering parameters (S-parameters).
- Use the Group delay option to analyze the group delay from the input to the output of the system.

By selecting the LSSP simulation component (LSSP), in the Simulation-LSSP palette, you can achieve the following:

- Perform a large-signal sweep of, for example, gain versus frequency and power, to determine the effect of gain compression on S-parameters.

By selecting P2D Simulation component (P2D), in the Simulation-LSSP palette, you can achieve the following:

- Create a system-level amplifier model by generating a power-dependent S-parameter file for a circuit-level amplifier design.
Applicable Measurements

From among the many measurements that are available, a variety of those specific to system analysis are found in their respective simulation palettes. Table 18-1 lists some of the system-applicable measurements that are available on the schematic page, and shows the palettes in which they can be found:

Table 18-1. Measurements for System Analysis

<table>
<thead>
<tr>
<th>Palette</th>
<th>Measurement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation-HB</td>
<td>BudLineariz</td>
<td>Linear budget simulation</td>
</tr>
<tr>
<td></td>
<td>Pspec</td>
<td>Power frequency spectrum</td>
</tr>
<tr>
<td></td>
<td>Vspec</td>
<td>Voltage frequency spectrum</td>
</tr>
<tr>
<td></td>
<td>Ip3in</td>
<td>Input third-order intercept point</td>
</tr>
<tr>
<td></td>
<td>Ip3out</td>
<td>Output third-order intercept point</td>
</tr>
<tr>
<td></td>
<td>CarrToIM</td>
<td>Ratio of carrier signal power to IMD power</td>
</tr>
<tr>
<td></td>
<td>SFDR</td>
<td>Spurious-free dynamic range</td>
</tr>
<tr>
<td></td>
<td>SNR</td>
<td>Signal-to-noise ratio</td>
</tr>
<tr>
<td>Simulation-LSSP</td>
<td>BudLineariz</td>
<td>Linear budget simulation</td>
</tr>
<tr>
<td></td>
<td>GainComp</td>
<td>Gain compression</td>
</tr>
<tr>
<td></td>
<td>PhaseComp</td>
<td>Phase compression</td>
</tr>
<tr>
<td>Simulation-S-Param</td>
<td>PwrGain</td>
<td>Power gain</td>
</tr>
<tr>
<td></td>
<td>MaxGain</td>
<td>Maximum available gain</td>
</tr>
<tr>
<td></td>
<td>NsPwrRefBW</td>
<td>Noise power in a reference bandwidth</td>
</tr>
</tbody>
</table>

These measurements and others are placed in the Schematic window and used in conjunction with a simulation.
**Fundamentals of Using Circuit Simulation Components for System Analysis**

*Figure 18-1* is a block diagram of a typical RF system. Such a system often consists of many interconnected linear and nonlinear “black-box” elements representing amplifiers, filters, mixers, modulators, demodulators, transmission lines, and radio links, as well as source and load matching elements. Each black box can in turn be composed of many circuit elements.

At the system simulation level, the signal transformation property of each black box is known, but the internal circuit characteristics of the elements need not be of concern.

*Figure 18-1. Block Diagram of a Typical RF System*

*Figure 18-2* shows an example RF system as represented on the schematic page. A variety of applicable simulation components and measurements are made available on the example schematic; turn them on and off as needed.

*Note*  This design, RF_SYS1.dsn, can be found in the examples directory under Tutorial/SimModels/networks. The results of the simulation can be found in RF_SYS1_spectra.dds.
Using Circuit Simulators for RF System Analysis

Example

The system depicted in the RF_SYS1.dsn uses a single upconversion stage and two downconversion stages. The RF frequency is 300 MHz, as defined in the P_1Tone source PORT1. The RF input power level is -10 dBm. This tone sums with 17.7 GHz from the first LO source, SRC1, to produce (among other tones) an IF of 18 GHz when mixed by the first upconversion mixer, MIX1. MIX1 and the other mixers in this example use the MixerIMT component, which relies on intermodulation tables to produce spurs. The parameter IMT_File references the table used for this component. For a discussion of IMT files as they are used here, refer to “Using MixerIMT/MixerIMT2 Models in Spurious Signal Analysis” on page 18-26.

The Chebyshev filter BPF1 (note the passband BWpass and quality factor Qu) then selects and passes the 18-GHz signal to amplifier AMP1, where the signal receives 20 dB of gain at a phase of 0 degrees. LINK1 applies both a transmitter and a receiver gain of 30 dB and adds the loss that would be incurred over a path length of 1 km. BPF2, another Chebyshev filter, receives the signal and passes it through to AMP2, which applies 30 dB of gain at 0 degrees phase.
MIX2 combines the 18-GHz signal with 16.5 GHz from the LO, SRC2, to produce (among other tones) a 1.5-GHz tone at the mixer’s output. This time the signal is passed through a much narrower (note the high Q) Butterworth filter to a low-gain amplifier, AMP3. From there the signal proceeds to a final downconversion stage, MIX3.

The signal input to MIX3 combines with 1.2 GHz from the LO at SRC3 to produce (among other tones) 300 MHz, the frequency originally inserted into the system. This is filtered by BPF4 (note the very high Q) and amplified by AMP4. Finally, a 50-ohm load at the output terminates the signal path.

Later we will revisit this design and the results of simulations on it.
Budget Analysis

Budget analysis determines the signal and noise performance for elements in the top-level design. Therefore, it is a key element of system analysis.

Budget measurements are performed upon data generated during a special mode of circuit simulation. AC and HB simulations are used in budget mode depending upon if linear or nonlinear analysis is needed for a system design. These measurements show the performance at the input and output pins of each element of the system at the top-level design. This enables the designer, for example, to adjust the gains or to reduce the nonlinearities of various components. These measurements can also indicate the degree to which a given component can degrade overall system performance.

Budget measurements include power gain, incident and reflected powers, noise figure, VSWR, and a variety of nonlinear measurements, such as SNR and gain compression.

There are various ways to obtain budget data:

- Use the Perform Budget simulation option (available in the HB and AC simulation dialog boxes). This option is required if budget measurements are to be used following a simulation (see below). Alternatively, the flag OutputBudgetIV can be set to Yes.

- Use the budget measurement components, available in the AC and HB simulation palettes. By placing one or more of the budget measurement components on the schematic and by selecting the required options, budget data can be generated.

- Add a budget path to your schematic using Simulate > Generate Budget Path. Budget data will be generated for the specified portion of the circuit. This is used in conjunction with other measurement components.

- Use Measurement Functions, available in data display windows as functions that can be input directly into an equation. First, the appropriate data must be referenced in the default dataset.

- Use the BudLinearization Component, available in the Harmonic Balance, LSSP, and XDB simulation palettes. This component, which must be used in conjunction with one of the harmonic balance simulators, provides information regarding the nonlinear effect of circuit elements.
Using the Perform Budget Simulation Option

Two simulators provide a budget simulation option:

- In the Harmonic Balance Simulation component (HB), select the **Params** tab, then select **Perform Budget simulation**.
- In the AC Simulation component, select the **Parameters** tab, then select **Perform Budget simulation**.

Using Budget Measurement Components

The budget measurement components are available in the AC and HB Simulation palettes, and must be used by selecting the required options. A budget data can be generated by placing one or more of these budget components in schematic.

The budget results at the terminal(s) of each element are sorted in ascending order of the component names. These component names are attached to the budget data as additional dependent variables.

Adding a Budget Path

You can generate budget results over a specified path in your circuit using **Simulate > Generate Budget Path**. You use this in conjunction with other budget measurement components, by replacing the pin with the name of the budget path. First specify the path, then modify your budget measurements.

To specify a path:

1. From the menu bar, choose **Simulate > Generate Budget Path**.
2. The names of the components in the circuit appear in two lists. Set the start of the path by selecting the name of a component in the left list.
3. Set the end of the path by selecting the name of a component in the right list.
4. Click **Generate**. To verify that the path is correct, click **Highlight** to highlight the path on the schematic. Click **Clear** to erase the highlighting.
5. If you are satisfied with the path, click **Close**.
6. Search the schematic for a measurement called BudPath, which was created when you invoked the command Generate Budget Path. You can change the name of the equation if desired. The default name is **budget_path**.
To modify your budget measurements:

1. Select a budget measurement and double-click to edit it.
2. Where the pin is specified in the budget expression, replace it with the name of the budget path measurement.

Using Budget Measurement Functions

The budget functions can also be entered by means of the Eqn component in the data display window. The Perform Budget simulation option must be selected prior to a simulation before measurement functions can be used following the simulation.

The budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

You can use a variety of budget-related measurement functions in the Equation entry field in the Data Display window. These include the following:

- `bud_freq`, `bud_gain`, `bud_gain_comp`, `bud_gamma`, `bud_ip3_deg`, `bud_nf`, `bud_nf_deg`, `bud_noise_pwr`, `bud_pwr_inc`, `bud_pwr_refl`, `bud_snr`, `bud_tn`, and `bud_vswr`.

**Note** For details on the above and other measurement functions, open a MeasEqn component dialog box and click Help.

Using the BudLinearization Component

This component is available in the Harmonic Balance, LSSP, and XDB Simulation palettes, and must be used in conjunction with those analyses. To use the BudLinearization component, place the component in the schematic and edit it to reference the simulation component to be used, as well as the circuit components that are to be linearized. If no component is specified, all the components in the top-level design are linearized one at a time.

The BudLinearization component first performs a regular harmonic balance simulation, and then looks at the DC operating point for each nonlinear system component in turn, linearizing the response of that component while the responses of the remaining nonlinear system components remain nonlinear. If a system component is in a subnetwork, the entire subnetwork will be linearized.
The results of BudLinearization analysis are sorted in ascending order of the names of the linearized components. LinearizedElementIndex, an integer independent variable, is attached to the data as an additional sweep variable. The first point corresponds to the results of the regular harmonic balance simulation when none of the components is linearized. Another variable, LinearizedElement, that contains the names of the linearized components is also generated.

**Note** The inclusion of the BudLinearization component in conjunction with a Harmonic Balance simulation causes \(N+1\) harmonic balance simulations to be run, where \(N\) is the number of nonlinear components to be linearized. Consequently, such a simulation will take \(N+1\) times as long.

**Budget Examples**
This section includes these budget examples:

- “Calculating Gain and Noise Figure” on page 18-12 shows how to determine the power gain and noise figure budgets for a typical RF System. It uses the AC simulator and measurement functions.
- “Calculating Spurious Signals and TOI” on page 18-15 shows how to use the Harmonic Balance simulator to analyze a PCS receiver.
- “Obtaining Budget Incident Power and Gain” on page 18-20 illustrates how to use budget measurement functions and how to display results.
- “Obtaining Group Delay Data” on page 18-23 uses the S-parameter simulator and linear sweep to calculate group delay data.
Calculating Gain and Noise Figure

*Figure 18-3* illustrates one way to obtain budget gain and noise figure data.

**Note** The design Linear_Budget.dsn is in the Examples directory under Com_Sys/Linear_Budget_prj. The results are in SchematicMeasurements.dds.

**Hint** To make it easy to observe the gains and losses contributed by individual components along the signal path, label them so that a List plot sorts them alphanumerically. In this case they have been labeled b1_BPF1, b2_AMP1, and so on.
This example uses the AC Simulation component, with parameters set as follows:

- Under the Noise tab, Calculate noise has been enabled. The noise nodes (the input of the first filter and output of the final filter) has been labeled “Vin” and “Vout” and have been added to the list of Nodes for noise parameter calculation. The Noise contributors mode is set to Sort by name.
- Under the Parameters tab, both Enable AC Frequency Conversion and Perform Budget simulation have been selected.
- Under the Frequency tab, the frequency is set to 1.960 GHz.

Note: In nonlinear noise analyses, it is recommended that the Options component be used to establish a global simulation temperature of 16.85 degrees Celsius. This can be done by editing Temp=16.85 in the Schematic window, or by selecting the Misc tab and editing Simulation temperature to that value.

Add budget measurements to the schematic:

- From the Simulation-AC palette, add BdGain (budget gain), BudNF (budget noise figure), BudNFd (budget noise figure degradation), and BudPwrI (budget incident power).
- Double-click the budget noise figure degradation component to edit it. Edit the equation by replacing “term2” with “b6” and changing “vout” to “Vout”. Note that b6 is the name of the output termination component.

To display all of the Budget measurements at once, open the data display SchematicMeasurements.dds. This will include gain, incident power, noise figure and noise figure degradation by component, both in tabular and graphic form. Ensure that the default dataset name is set to the name of the design you have simulated. The results of the simulation appear as follows:
The first listing shows the losses and gains, in dB, contributed by the various components. For example, the first amplifier has a nominal insertion loss (IL) of 5 dB at 1.960 GHz, but here the figure is –5.239 dB—the result of reflection and thermal loss. Try lowering the reflection parameters $S_{11}$ and $S_{22}$ to approximate ideal values (simply add zeros) to observe the bud_gain figure approach 5 dB. The second listing shows the incident power at the input of each component. The third and fourth listings show the noise figure and the noise figure degradation at pin 1 of each component, respectively. For more discussion of noise analysis, refer to “System Noise Analysis” on page 18-30.

Budget measurements may also be performed entirely within data display window by adding equations that operate on the current and voltage data provided by the AC simulation. The data display DataDisplayMeasurements show an alternative way of using the budget functions, instead of adding measurements to a schematic, after a simulation you can get the same calculations by adding equations to the data display page.

**Calculating Spurious Signals and TOI**

*Figure 18-4* illustrates traces of the spectral tones (spurious signals) for various nodes of the design RF_SYS1.dsn. For a detailed discussion of mixing products, refer to “Using Mixer1MT/Mixer1MT2 Models in Spurious Signal Analysis” on page 18-26.
Figure 18-4. Spectral Data for RF_SYS1.dsn
Figure 18-5 illustrates a setup for obtaining third-order intercept.

Note: The design PCS_Rx_TOI_test.dsn is in the Examples directory under Com_Sys RF_System_prj. The results are in PCS_Rx_TOI_test.dds.

Receiver Third-Order Intercept Point and Carrier-to-Intermodulation Distortion Simulations

Two sources at the input define two closely adjacent tones, as can be found under typical conditions of communications interference. (A P_nTone component could also be used.) The source at the top of the schematic, identified as PORT1, can be used to test the degree to which the outgoing signal leaks back into the IF stage.
Note CarrToIM and IP3out measurement components have been used to define those measurements for the lower and upper intermodulation products (selected by means of the mixing indices vector) that will be passed by the filters. IP3out must reference 50 ohms. Also, although there are six apparent source tones in the design (at 1880, 1880.3, 1960, 1960.3, 2048, and 84 MHz), only five tones are independent.

A plot of the IF output, VOUT_IF, appears as follows:

The mixing products of interest center around 6 MHz, as determined by the final filters. Note that only one of the third-order intermodulation products (at 6.06 MHz) is safely below about –75 dB. The spur at 5.97 MHz indicates that the filter's bandwidth needs to be adjusted.

The listing below is a plot of the Mix data output, which produces mixing indices vectors. Highlighted are the coefficients of those frequency components whose product resulted in 6.030 MHz (0 * 1880 + 1 * 94 - 1 * 88 + 1 * 0.03 + 0 * 1960 = 6.03 MHz).
The listing below, of the four equations on the schematic, shows the relationships, in dB, of the various products. Note that the difference between the carrier and the lower intermodulation product (the second and first columns, respectively) is approximately 26 dB.

<table>
<thead>
<tr>
<th>freq</th>
<th>M(1)</th>
<th>M(2)</th>
<th>M(3)</th>
<th>M(4)</th>
<th>M(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001 MHz</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.003 MHz</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0.008 MHz</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>0.02 MHz</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.08 MHz</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>0.94 MHz</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>7.94 MHz</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>.1</td>
</tr>
<tr>
<td>9.97 MHz</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>15.87 MHz</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>.2</td>
<td>1</td>
</tr>
</tbody>
</table>

The fundamental of the product in the first column is defined by

\[ 0 \times 94 + 1 \times 94 - 1 \times 88 + 0 \times 0.03 + 0 \times 1960 = 6 \text{ MHz} \]

and its intermodulation product by

\[ 0 \times 1880 + 1 \times 94 - 1 \times 88 - 1 \times 0.03 + 0 \times 1960 = 5.97 \text{ MHz} \]

The fundamental of the product in the second column is defined by

\[ 0 \times 1880 + 1 \times 94 - 1 \times 88 + 1 \times 0.03 + 0 \times 1960 = 6.03 \text{ MHz} \]

and its intermodulation product by

\[ 0 \times 1880 + 1 \times 94 - 1 \times 88 + 2 \times 0.03 + 0 \times 1960 = 6.06 \text{ MHz} \]
Obtaining Budget Incident Power and Gain

Figure 18-6 illustrates the use of the bud_gain, bud_snr and bud_pwr_inc measurement functions.

Note: The design IQ_mod_bud.dsn is in the Examples directory under Com_Sys/MultiChan_NL_Budget_prj. The results are in IQ_BudgetSchematic.dds and IQ_Budget.dds.

It is similar to the preceding example, with the exceptions that only a single tone is inserted at the input, and the transmitter tone has been replaced by a terminating resistor.
To display all of the Budget measurements at once, open the data display IQ_BudgetSchematic. This will include frequency plan, gain, signal-to-noise ratio and incident power in tabular and graphic form. Ensure that the default dataset name is set to the name of the design you have simulated. The results of the simulation appear as shown here:

<table>
<thead>
<tr>
<th>Component</th>
<th>f (GHz)</th>
<th>Gain (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fo_0101</td>
<td>1.00010</td>
<td>0.452-10</td>
</tr>
<tr>
<td>fo_0102</td>
<td>1.00101</td>
<td>0.371</td>
</tr>
<tr>
<td>fo_0103</td>
<td>1.00203</td>
<td>0.353</td>
</tr>
<tr>
<td>fo_0104</td>
<td>1.00304</td>
<td>0.361</td>
</tr>
<tr>
<td>fo_0105</td>
<td>1.00405</td>
<td>0.372</td>
</tr>
<tr>
<td>fo_0106</td>
<td>1.00506</td>
<td>0.381</td>
</tr>
<tr>
<td>fo_0107</td>
<td>1.00607</td>
<td>0.392</td>
</tr>
<tr>
<td>fo_0108</td>
<td>1.00708</td>
<td>0.401</td>
</tr>
</tbody>
</table>

Gain from Port 1 to the input of each component in the "a" path

<table>
<thead>
<tr>
<th>Component</th>
<th>f (GHz)</th>
<th>Gain (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fo_0101</td>
<td>1.00010</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0102</td>
<td>1.00101</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0103</td>
<td>1.00203</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0104</td>
<td>1.00304</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0105</td>
<td>1.00405</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0106</td>
<td>1.00506</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0107</td>
<td>1.00607</td>
<td>110.8408</td>
</tr>
<tr>
<td>fo_0108</td>
<td>1.00708</td>
<td>110.8408</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio from Port 1 to the input of each component in the "a" path

Incident Power Budget

The first listing shows the fundamental frequencies for plan 1 and the power gain from the input port to pin 1 of each component. The second listing (top-left corner) shows the signal-to-noise ratio at pin 1 of each component. The last listing (bottom)
Using Circuit Simulators for RF System Analysis

shows the incident power at the input of each component at the fundamental frequencies in plan 1 through the system. In this example, note the imbalance at the power combiner input due to the nonlinearity of b3_AMP2.

**Note**  Certain measurements (such as bud_gain), although they derive data for a single tone, will output results for all harmonics.

The data display IO_Budget shows an alternative way of using the budget functions. The budget measurements are performed within data display window by adding equations that operate on the current and voltage data provided by the HB simulation.
Obtaining Group Delay Data

Figure 18-7 illustrates the use of the S-parameter simulator in obtaining group delay data. For more information about group delay, refer to "Calculating Group Delay" on page 6-6, and "Group Delay" on page 6-12.

**Note**  The design Linear_Sweep.dsn is in the Examples directory under Com_Sys/RF_Sys_prj. The results are in Linear_Sweep.dds.

In the S-parameter Simulation component a linear sweep centers closely around the center frequency of the first filter. Under the Parameters tab, the Group delay option has been selected. In addition, Enable AC Frequency Conversion has been selected, and S-parameter freq. conv. port has been set to 1, the input port.

**Note**  The frequency conversion port must be the number of the input port.

---

Figure 18-7. Using the S-parameter Simulator to Obtain Group Delay Data

The following is a plot, in dB, of S(2,1), showing the response of BPF2.
Using Circuit Simulators for RF System Analysis

Below, a plot of the function phase(S(2,1)) shows wrapped phase versus frequency. This is essentially a compressed view of the response, as the display is normalized to accommodate the ±180-degree variations without unnecessary repetition.

Finally, a plot of the function delay(2,1) (the delay at the output port with respect to the signal at the input port) shows the shift in phase in the filter’s passband (below).
Using Circuit Simulators for RF System Analysis

Using MixerIMT/MixerIMT2 Models in Spurious Signal Analysis

System spurious signal analysis determines the spurious spectral tones at the system output. In the mixer IMT (intermodulation table) model, all intermodulation products are due to mixer signal input and local oscillator (LO) mixing.

The following discusses how spurious signals are generated, the system spurious-signal simulation approach, and spurious signal simulation topology restrictions.

Within an RF subsystem, spurious signals are generated by mixers, nonlinear amplifiers, and spectrally impure oscillators.

In mixers, spurious signals are due to the harmonic mixing of RF and local oscillator (LO) input signals. For single-tone RF and LO signals, the spurious signal frequencies are the \( N \cdot RF \) plus \( M \cdot LO \) harmonic products, where \( N \) and \( M \) are integers. For multitone RF and LO signals, the spurious-signal frequencies include not only the primary \( N \cdot RF \) and \( M \cdot LO \) harmonic products for each RF tone combined with each LO tone individually, but also the cross-modulation products between multiple RF tones and LO tones.

Mixer spurious-signal generation (\( N \cdot RF \) plus \( M \cdot LO \)) is typically defined in the RF industry by means of .imt files. An .imt file is used to define a mixer’s spurious-signal generation properties as a function of RF and LO tone mixing order, as well as of RF and LO signal power levels.

In nonlinear amplifiers, spurious signals are due to the nonlinear distortion of the input RF signal. This distortion results in both the harmonics of RF tones and the cross-modulation frequencies of the input RF tones.

In oscillators, spurious signals are due to harmonics of the fundamental oscillation frequency, as well as to unwanted spurious modulation sidebands at the oscillator output.

All spurious signals may be defined by a frequency-domain representation. Each RF signal is represented mathematically as the summation of RF spectral tones. Each tone is individually represented by the triplet of numbers: frequency, power (dBm), and phase (degrees).
Every mixer has an associated RF input signal (RFin), an LO signal, and an RF output signal (RFout). A MixerIMT or MixerIMT2 component in conjunction with an associated .imt file produces spurious output tones at the following frequencies:

\[ RF_{out} = N \cdot RF_{in} \pm M \cdot LO; \ N = 0, 1, \ldots 15 \]
\[ M = 0, 1, \ldots 15 \]
\[ N + M \leq 15 \]

Each .imt file contains information about the relative power level of output spurious tones with respect to the level of the fundamental mixing tone. An example .imt file, for a double-balanced mixer, is shown below.

```plaintext
]$ DBL1.IMT
! @(#) $Source: /sr/src/gemini105/templates/dbl1.imt $Revision: 1.2 $Date: 1997/06,
! Intermodulation table for double balanced mixer $i
! Signal Level (dBm) LO Level (dBm)
-10
!
<table>
<thead>
<tr>
<th>M x LO (Horizontal)</th>
<th>N x Signal (Vertical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15</td>
<td></td>
</tr>
<tr>
<td>99 26 35 39 50 41 53 49 51 45 65 55 75 65 85 99</td>
<td></td>
</tr>
<tr>
<td>24 0 35 13 40 24 45 28 49 35 55 45 65 55 99</td>
<td></td>
</tr>
<tr>
<td>73 73 74 76 47 64 64 69 69 65 75 75 85 85 99</td>
<td></td>
</tr>
<tr>
<td>67 64 69 50 77 47 74 44 74 45 75 75 85 99</td>
<td></td>
</tr>
<tr>
<td>86 90 86 88 88 85 86 85 90 85 85 99</td>
<td></td>
</tr>
<tr>
<td>90 80 90 71 90 68 90 65 88 65 99</td>
<td></td>
</tr>
<tr>
<td>90 90 90 90 90 90 90 90 90 90 99</td>
<td></td>
</tr>
<tr>
<td>90 90 90 90 90 90 90 90 90 90 99</td>
<td></td>
</tr>
</tbody>
</table>

The .imt file is valid for a specific reference-power level for RFin, PRFREF (dBm), and a reference LO power level, PLOREF, both of which are included in the .imt file. If the input power levels differ from the signal and LO levels specified in the .imt file, interpolation is performed. The valid ranges for interpolation are RFin \leq PRFREF + 3, and PLOREF - 10 \leq PLO \leq PLOREF + 3.
Note: The simulator does not check whether the RF and LO drive levels are exceeded. It is your responsibility to understand the signal levels within your system so as to not to set up a simulation in which drive levels are either too high or too low for the mixer intermodulation table you have selected.

The output spurs from a mixer vary with RF\textsubscript{in} drive power and with LO drive power. Consider a mixer with an .imt file and with a conversion gain ConvGain, in dB. Then the mixer output \( N \cdot M \) IM product levels are as follows:

Let

\[ S (\text{source}) = \text{RF}_{\text{in}} \text{ drive level, in dBm} \]
\[ L = \text{LO}_{\text{in}} \text{ drive level, in dBm} \]
\[ P_{\text{IM}} = \text{the level of the output } N \cdot M \text{ IM product, in dBm} \]
\[ \text{IMT}[N][M] = \text{the .imt file's value for the } N \cdot \text{RF by } M \cdot \text{LO intermodulation product, where} \]
\[ N = \text{harmonic number of the } \text{RF}_{\text{in}} \text{ signal and is the row index to the .imt file} \]
\[ M = \text{harmonic number of the LO signal and is the column index to the .imt file} \]

Therefore,

if \( S = P_{\text{RFREF}} \) and \( L = P_{\text{LOREF}} \), then

\[ P_{\text{IM}} = S + \text{ConvGain (dB)} - \text{IMT}[N][M] \]

If \( N \neq +1 \) or \(-1\) and \( M 
eq 0 \), then

\[ P_{\text{IM}} = S + \text{ConvGain (dB)} - \text{IMT}[N][M] + |N - 1| \cdot (S - P_{\text{RFREF}}) + |M| \cdot (L - P_{\text{LOREF}}) \]
If $N = +1$ or $-1$ and $M$ is nonzero, then

$$P_{IM} = S + \text{ConvGain}(\text{db}) - \text{IMT}[1][M] + |M| \cdot (L - P_{LOREF})$$

These relationships state the following:

- A decrease in RF drive power by $k$ dB below the reference results in an increase of suppression of any $N \cdot M$ product by $k \cdot (|N| - 1)$ dB (that is, a decrease in the level of the IM product).
- A decrease in LO drive power by $k$ dB below the reference results in an increase of suppression of any $N \cdot M$ product by $k \cdot |M|$ dB (that is, a decrease in the level of the IM product).

These IM suppression formulas are most accurate when $S = P_{RFREF}$ and $L = P_{LOREF}$. However, they give reasonable results for $S \leq P_{RFREF} + 3$, and for $(P_{LOREF} - 10) \leq L \leq (P_{LOREF} + 3)$. 
System Noise Analysis

There are three major contributors to system noise: passive-element thermal noise, active-element noise, and oscillator phase noise. (For a discussion of oscillator phase noise, refer to Chapter 9, Harmonic Balance for Oscillator Simulation.) The system noise response is simulated by the program under small-signal conditions. A linear analysis of system noise gives a reasonable representation even when the signal is well into compression, provided the signal-to-noise ratio is not too low.

System thermal and active noise simulation uses a noise-wave model that accounts for the effects of element mismatches.

System phase noise is described as the phase-noise level versus the oscillator offset frequency. System phase noise is simulated by sequentially combining the phase-noise characteristics of consecutive oscillators from the system input port to the system output port.

The combined active noise, thermal noise, and phase noise at the system outputs can be observed in terms of noise power in dBm versus frequency. This total noise power can be combined with the output signal so that the system output signals can be observed with the system noise added.

Note  In a noise analysis the following assumptions are made with respect to the terminating resistances that are implicitly connected to the system network’s input and output ports:

(1) The signal source (assumed to be connected to the system network input, or Port 1) is assumed to have a resistance of 50 ohms at a standard physical temperature, $T_0$, of 290 K. This source resistance provides noise power at a noise density of $-174$ dBm/Hz into the system. This source must be a power source.

(2) The system network outputs (any port other than Port 1) are assumed to be terminated with 50 ohms at absolute zero physical temperature, 0 K. This termination resistance does not contribute any noise to the noise measured from the system. This termination must be a Term component.

When a noise analysis is requested, it is recommended that you use an Options component and set the global temperature to 16.85 degrees Celsius.
Chapter 19: Parameter Sweeps and Sweep Plans

Generally, sweeps of individual parameters can be performed most efficiently from within many of the simulator dialog boxes themselves. The ability to step through a series of values automatically is incorporated into all the standard simulation components. Sweeps can be performed at both the circuit and the system level.

The following parameters are typical candidates for sweeping:

- Signal frequency, amplitude, or power
- Bias voltage or current
- Resistance
- Signal path attenuation
- Impedance
- Ambient temperature
- Most component parameters

However, it is possible to combine sweeps of several parameters into a hierarchical sweep plan. By using parameter sweeps, you can do the following:

- Find the bias voltage that yields the best mixer conversion gain.
- Find the load impedance that yields the lowest harmonic distortion.
- Simulate a load-pull measurement.
- Simulate the effects of process variations and temperature on circuit performance.

By selecting the ParamSweep and Sweep Plan components from any of the simulator palettes, you can sweep a variety of parameters and construct a series of sweep plans for special purposes.
Conducting Sweeps

Sweeps of frequencies can be conducted from within many of the simulators themselves, using options available under the Sweep tab where applicable. To sweep a parameter such as power, for example, select a ParamSweep component from a simulator palette and edit it. It is necessary to ensure that frequency and power variables to be swept are defined appropriately in a source component, such as a P_1Tone component (available in the Sources-Freq Domain library).

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and results combined into a single dataset on the local machine. For details on setting up remote and local machines for remote processing, refer to these appendices:

- Appendix E, Using Remote Simulation in the Installation on PC Systems manual

Using Options under the Simulator Sweep Tab

A variety of simulation options allow you to conduct a simulation for only a single parameter at a single value, or to sweep a parameter over a defined range, either linear or logarithmic. They also allow you to select a named sweep plan that you can define.

Select the Sweep tab in various simulation components to do the following:

- Define a parameter to sweep (use a Var eqn component to define it)
- Select a sweep type
- Select start, stop, and step sizes
- Select a sweep plan

A parameter entered into the Parameter to sweep field will appear on the schematic in quotes. To display a parameter to sweep so you can edit it directly on the schematic, do the following:

1. Select the Display tab.
2. Select SweepVar.
3. Select **Display parameter on schematic**, then click **OK**.

You can then define that parameter directly on the schematic, taking care to place the definition in double quotes.

**Using Sweep Components**

In addition to the sweeps that are provided within various simulators, ParamSweep and SweepPlan components are available in each simulation palette. Place and edit these like other components. Also, ensure that the frequency and power variables to be swept are defined appropriately in a source component (for example, a P_1Tone component).

Use a ParamSweep to select a named parameter to sweep, and use a SweepPlan to order the way in which various simulators are invoked. To use a SweepPlan component that you have defined, select **Use sweep plan** (under the Sweep tab of the simulation component) and enter the name of the SweepPlan component.

---

**Note**  
The placement of sweep components within a circuit or system design does not affect the order in which parameters are swept. Similarly, the order in which the sweeps are automatically numbered does not determine the order in which they are executed. The order of execution is determined by the order in which one sweep calls another, as determined by the value of the parameter SweepPlan. The simulation component calls the first sweep plan to be conducted, whatever it is named.
Basic Procedures

This section presents the following example sweep scenarios:

- “Using ParamSweep to Sweep Two Parameters” on page 19-4
- “Using SweepPlans to Perform Fine and Coarse Sweeps” on page 19-8

Note: The appropriate simulators are required to run the following simulations, i.e. a Harmonic Balance simulation requires the Harmonic Balance simulator (included with all Circuit Design suites except the RF Designer suite.). You may build the examples with the appropriate license, you will simply be unable to run the simulations.

Using ParamSweep to Sweep Two Parameters

Figure 19-1 illustrates an example setup that uses a ParamSweep component (available in all the simulation palettes) to sweep two parameters. In this case, the result is a curve-tracer display of collector current versus $V_{ce}$ for different values of $I_{bb}$.

Note: This design, Curve_Tracer.dsn, is in the Examples directory under MW_Ckts/ LNA_pr. The results re in Curve_Tracer.dds.

Although it is not necessary for the current design, such a simulation can be used to characterize a device whose I-V relationships are unknown. SRC1 establishes collector-to-emitter voltage $V_{ce}$. SRC2 establishes base current $I_{bb}$. The current probe, Probe1, measures collector current $I_{cc}$. Note that Probe1 has been named Icc and that Icc.1, the current at this point, will be plotted later.

The procedure for using the ParamSweep component, in conjunction with a simulator sweep and an equation, is outlined as follows.

Note: The following steps describe the design under discussion. Modify the details to suit your particular needs.
To use the ParamSweep component to sweep two parameters:

1. Use a VarEqn component (available from Component Palette List > Data Items > Var eqn) to define two variables—an “inner” and an “outer” variable. The inner variable is swept over its full range each time the outer variable is stepped.

2. Use a DC Simulation component to define the parameter to sweep, the inner variable.

3. Use a ParamSweep component to establish a sweep plan for the outer variable.

The following illustrates this procedure in detail.
Parameter Sweeps and Sweep Plans

**Define Inner and Outer Variables**

1. Edit the VarEqn component on the schematic.
2. In the Select Parameter field, ensure that the following variables are defined by an equation:
   - VCE = 0 V
   - IBB = 0 A

   **Note** The simulator requires that all variables be initialized. The above voltage values would be used if no sweep were in effect.

   These equations are written in the field on the right of the box, with Variable or Equation Entry Mode set to Name=Value.

3. Click **OK**.

**Define the Inner Variable in the Simulation Component**

1. Edit the DC Simulation component.
2. In the Parameter to sweep field, ensure that VCE is entered. VCE is the inner variable, which has been established in the Var/Eqn component.

   **Note** Variables entered into this field will appear in quotes on the schematic. If you enter a variable directly on the schematic (in this case, as the right-hand side of the SweepVar statement in the DC component), you must surround the variable with double quotes.

3. Ensure that the following parameters are set:
   - Sweep Type = Linear
   - Start/Stop is selected.

4. Click **Apply**, then **OK**.
Establish a Sweep Plan for the Outer Variable

1. Edit the ParamSweep component:

2. Ensure that the following parameters are set, and make them visible on the schematic:
   - Parameter to sweep = IBB. This establishes Idc at SRC2.
   - Sweep Type = Linear
   - Start/Stop is selected
   - Start = 20 \( \mu \)A
   - Stop = 100 \( \mu \)A
   - Step-size = 10 \( \mu \)A

3. Select the Simulations tab, and ensure that DC1 is entered in the Simulation 1 field.

Launch the Simulation and Display Data

1. Launch the simulation. The following is a plot of Icc.i:

Information such as that depicted above is useful in deciding whether a device is suitable for a given application where the limiting factors are available voltage and current.
Parameter Sweeps and Sweep Plans

Using SweepPlans to Perform Fine and Coarse Sweeps

Figure 19-2 illustrates an example setup for using two SweepPlan components (available in all the simulation palettes) to perform a fine sweep of noise figure versus RF frequency, followed by a coarse sweep.

Note: This design, SweptRF_NF.dsn, is in the Examples directory under RFIC/Mixers_prj. The results re in SweptRF_NF.dds.

This simulation uses a Harmonic Balance Simulation component (HB) to call a sweep plan, and that sweep plan, in turn, calls a second sweep plan. A simulation component cannot call more than one sweep plan.

Note: The following steps describe the design under discussion. Alter the details to suit your particular needs.

To use two SweepPlan components to perform a fine and a coarse sweep:

1. Use a VarEqn component (available from Component Palette List > Data Items > Var eqn) to define all frequencies.

2. Using two SweepPlan components (available in all the simulation palettes), establish two sweep plans—one for a fine sweep and one for a coarse sweep.

3. Use a Harmonic Balance Simulation component (HB) to define the parameter to sweep and reference a sweep plan.
The following illustrates this procedure in detail.

Figure 19-2. Example Setup for Using Two Sweep Plans

Noise figure can be simulated without a large-signal tone at the RF input. With only a single large-signal tone (the LO), the simulation runs in 15 seconds on an HP 8500 Series 6100 workstation.
Parameter Sweeps and Sweep Plans

Define Frequencies

1. Edit the VarEqn component on the schematic.

2. In the Select Parameter field, confirm that the following variables are defined by an equation:
   • \( \text{Fif} \) (intermediate frequency) = 70 MHz. This is the noise frequency defined in the Harmonic Balance Simulation component (HB) (see below).
   • \( \text{FrF} \) (RF input frequency) = 1 GHz. This is the input frequency defined in the Harmonic Balance Simulation component (HB) (see below).
   • \( \text{Flo} = \text{FrF} - \text{Fif} \). Flo is the frequency assigned to the P_1Tone component at the LO input of the mixer.

\[ \text{Note} \quad \text{The simulator requires that all variables be initialized. Although a value of 0 MHz or 0 GHz is sufficient to establish units for a local variable, leaving realistic units on the schematic allows a meaningful simulation to be conducted even when those variables are not swept.} \]

These equations are written in the field on the right of the box, with Variable or Equation Entry Mode set to Name=Value.

3. Click **Apply**, then **OK**.
Establish Two Sweep Plans

1. From any simulation palette, select the SweepPlan component and place two of these in the Schematic window.

2. Edit the first SweepPlan (Plan1) as follows:
   - Sweep Type = Linear
   - Start = 300 MHz
   - Stop = 1 GHz
   - Step = 350 MHz
   - Select Next Sweep Plan, and enter Plan2 (to be established in the next step) in the field below the option.

3. Edit the second SweepPlan (Plan2) as follows:
   - Sweep Type = Linear
   - Start = 1 GHz
   - Stop = 7 GHz
   - Step = 3 GHz
   - Because this is the last SweepPlan in this series, do not select Next Sweep Plan or enter a plan name.
Parameter Sweeps and Sweep Plans

**Define Sweep Frequencies and Sweep Plan in the Simulation Component**

1. Under the **Sweep** tab of the Harmonic Balance Simulation component (HB), establish the following:
   
   • Parameter to sweep = Frf

   **Note** Variables entered into this field will appear in quotes on the schematic. If you enter a variable directly on the schematic (in this case, as the right-hand side of the SweepVar statement in the DC component), you must surround the variable with double quotes.

   • Sweep Type = Linear
   
   • Use sweep plan is selected, and the entry in the field is Plan1. You can use the pulldown menu at the right of the field to select the name of a SweepPlan component.

   Other sweep parameter entries will be overridden by values in the SweepPlan component.

   • Finally, because this is a noise analysis, also select **Nonlinear noise** to enable the analysis.

2. Click **Apply**, then **OK**.

**Launch the Simulation and Display Data**

1. Launch the simulation. The following are Rectangular and List plots of noise figure
Additional Examples

An additional example of a swept-parameter simulation is found in the examples directory, in RFIC/Mixers_prj/networks. IMDLOSwp.dsn uses a ParamSweep component to sweep LO power in conjunction with harmonic balance simulation.
Recommendations and Tips

This section presents suggestions for using sweeps and improving the accuracy of results.

Ensuring that Sweep Results are Displayed Correctly

Parameter sweeps in the time domain are remarkable only in the way that they affect the adaptive time-step algorithm, which may have possible negative effects when results are displayed in the Data Display window. If the Transient simulator is allowed to use its adaptive time-step algorithm and Max time step is not specified by the simulation component, the simulator will probably produce results that have irregular data intervals. This does not matter when you are simulating only versus time and not sweeping — that is, when you are attempting to produce only a single trace for each nodal waveform, not a family of traces.

The potential difficulty arises when the Data Display server attempts to display a family of traces, each having different numbers of trace points and irregular spacings. Simply stated, the data cannot be displayed. The Data Display server must have a rectangular array of data. This means that all subtraces must have the same number of points, and that the spacing between the points must be the same as that between the corresponding points in other subtraces. The spacing can be irregular, as long as the distribution of spacing along the x-axis is the same for all traces. The only way to be sure that the Data Display server receives a neatly formatted, rectangular array of data is to specify a value for Max time step. This is especially true of Monte Carlo simulations, which are essentially statistical parameter sweeps.
Controlling the Amount of Data Sent to the Dataset

Time-domain simulations generally take more time and produce more nodal data than do comparable frequency-domain simulations. Hundreds (or even thousands) of time points may be required to simulate the behavior of the circuit accurately. This makes it especially important to minimize the number of time points and swept parameter values. You can control the amount and kind of data sent to the dataset using the Output tab of each analysis controller. For details, refer to “Selectively Saving and Controlling Simulation Data” on page 1-10.

Using Sweeps in Monte Carlo Analysis

It is possible to use sweeps in Monte Carlo statistical analyses with the transient simulator, to the extent that histograms can be generated. Yield percentage analyses are not possible except through the following indirect method.

---

Note: Monte Carlo analyses are enabled through the use of the Yield and YieldOptim components in the Optim/Stat/Yield palette (under their Parameters tab, enable Random variables to dataset, and establish seed values as appropriate).

Yield percentages can be estimated if the pass/fail yield criterion can be expressed in terms of node voltages and currents at individual time points.

Instead of plotting a histogram, plot the cumulative distribution function (CDF) using the cdf() operator and equations. The CDF increases monotonically from 0 to 1. Insert two markers onto the CDF trace at the limits of the pass/fail criterion. Then use the delta marker mode to find the difference between them. The yield percentage is the difference between the two markers, multiplied by 100. For example, suppose the response of a circuit to a 1.0-V step with a 100-psec risetime is being plotted. The maximum risetime desired from the circuit is 300 psec.

Ask for 100 Monte Carlo trials. Select the time point where the output voltage must be at least 90% (perhaps $t = 350$ psec). Then plot the CDF multiplied by 100. Insert a marker on the 90% value and another on the maximum value, then examine the difference. This difference is approximately the yield.

Monte Carlo statistical analyses can be regarded as a special type of parameter sweep in which the parameter values are assigned randomly, rather than as a succession of ascending values.
Parameter Sweeps and Sweep Plans
Chapter 20: Working with Data Files

Data files enable you to take data from sources outside of Advanced Design System and apply it to projects in ADS. There are many applications for using data files, but a common purpose is to define the behavior of a specific component or create a “black box” component. Some examples are:

• Using S-parameters to define the behavior of a linear “black box” component representing an attenuator, filter, or small-signal transistor. The S-parameters, which are saved in a file, are used in conjunction with a component like the S2P.

• Storing sets of transistor model parameters in separate files, and accessing them automatically through the course of a simulation to define the behavior of the transistor.

• Defining the behavior of an amplifier by using the S2DAmplifier component and data saved in an .s2d file. The data defines the behavior of the amplifier during a simulation.

There are other methods of developing user-defined models, such as using SDDs, FDDs, equation-based components, or the Model Builder. This chapter focuses on how to use the various types of data files to define component behavior. This chapter also describes the format requirements of these files.
Types of Data Files

A data file is simply data in an ASCII text file, but there are several formats to choose from. When determining the type of data file to choose:

- The format you choose may depend on the type of data you have and where you want to use the data. The table below lists supported file formats and examples of where they are used.

- You can use a DataAccessComponent to access the data from any data file regardless of format and use it with any component that accepts file-based parameters. In this case, you need to make sure there is a logical relationship between the data and how you intend to use it. For more information, refer to “Using Data Files, Datasets, and Data Access Components” on page 20-5.

<table>
<thead>
<tr>
<th>File Type</th>
<th>Description</th>
<th>Usage Examples</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SnP Touchstone Format</td>
<td>Small signal S, H, Y, Z, or G-parameters. May optionally include noise data (for 2-ports only). Where n is the number of ports from 1 to 99.</td>
<td>n-port S-parameter file (SnP) components in the Data Items library.</td>
<td>“SnP Format” on page 20-6.</td>
</tr>
<tr>
<td>MDIF Formats</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMT</td>
<td>Table of mixer intermodulation products between the LO and signal that relates to your mixer IM output level to signal input level</td>
<td>MixerIMT in the System-Amps &amp; Mixers library.</td>
<td>“IMT Format” on page 20-68.</td>
</tr>
<tr>
<td>File Type</td>
<td>Description</td>
<td>Usage Examples</td>
<td>Details</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>TIM</td>
<td>Time-domain data</td>
<td>TimeFile item in Times Sources and OutFile item in the Sinks library.</td>
<td>“TIM, BINTIM Formats” on page 20-70.</td>
</tr>
<tr>
<td>BNTIM</td>
<td>Time-domain data in binary format</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSCR</td>
<td>file of general, discrete data points</td>
<td>Components that accept file-based parameters, link via the DAC.</td>
<td>“DSCR Format” on page 20-74.</td>
</tr>
<tr>
<td>Generalized MDIF</td>
<td>Use in place of any specific MDIF</td>
<td>AmplifierS2D, AmplifierP2D, or any other MDIF example listed above.</td>
<td>“Generalized MDIF” on page 20-76.</td>
</tr>
<tr>
<td>Probability Density format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDF</td>
<td>User-specified probability density functions for arbitrary distributions that are not correlated.</td>
<td></td>
<td>“PDF Format” on page 20-79.</td>
</tr>
<tr>
<td>Cadence Alta Group SPW Formats</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCIG</td>
<td>Time-domain voltage data</td>
<td>TimeFile item in Times Sources and OutFile item in the Sinks library.</td>
<td>“ASCSIG and SIG Formats” on page 20-81.</td>
</tr>
<tr>
<td>SIG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITIfile Format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITI</td>
<td>General data format supported by network analyzers. Capable of storing multiple packages of multi-dimensional data</td>
<td>n-Port S-parameter file (SnP) components in the Data Items Library.</td>
<td>“CITIfile Data Format” on page 20-83.</td>
</tr>
<tr>
<td>Model Files</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model MDIF</td>
<td>Model parameters for nonlinear devices</td>
<td></td>
<td>“Writing Model Files” on page 20-103.</td>
</tr>
</tbody>
</table>

For information about a particular component, refer to the Circuit Components or Signal Processing Components manual, or click Help when editing component parameters.
Working with Data Files

Making a Data File

You can create a data file using these methods:

- Manually type the data into a text file using any text editor, being sure to follow the formatting guidelines for the type of data file that you want.

- Use the Instrument Server. The Instrument Server enables you to take measured data from an instrument (such as S-parameters from a network analyzer) and copy the data into a text file. The Instrument Server has additional capabilities for transferring data; refer to the Instrument Server or Data Display manual.

- Perform a P2D simulation. The results of the simulation are saved to a file in P2D format, which can then be used with the AmplifierP2D. For more information, refer to Chapter 14, P2D Simulation.

Saving a Data File

When saving a data file, save it as an ASCII text file.

- A data file does not require a particular extension, but you may want to use the File Type from the table above for identification purposes. If you choose a different extension, you must provide this information to the component you intend to use through the File parameter of the component.

You can save the data file:

- In the project where it is used under the <prj>/data directory. This is the default location if no path is provided to the component using the file.

- Any location, if you provide the full file path to the component. You provide this information using the File parameter of the component.

For more information about the File parameter, place the component of interest on a schematic, double-click to edit it, and click the Help button at the bottom of the dialog box.
Using Data Files, Datasets, and Data Access Components

Both data files and datasets can contain data that you want to use with a component. You can use either one, depending on your situation:

- You may want the S-parameters from a simulated design. In this case, use the dataset and a DAC to link the data to the component you want to use.
- You may have S-parameter specifications from a component sheet. In this case, type them into a data file.

You also want to consider the method of linking your data with the component of interest:

- You have a data file, like an .s2d, and you want to use it with a component that can read such a file, like an AmplifierS2D. No DAC is needed.
- You have a dataset or data file, but the component you want to use doesn’t read the data directly. Use a DataAccessComponent to link the data file and component. The component you choose must have the file-based option under the Parameter Entry Mode or the parameter AllParams. For example, the BJ TM1 model has the parameter AllParams; the R component has the Parameter Entry Mode. For instructions on how to use a DAC, refer to the component documentation in Circuit Components or the User’s Guide.

Examples

You can find designs that use different data files in the Examples directory under Data_comp_prj and DataAccess_prj.

Instructions for using a particular type of file with a component that is designed to read the file (like an .snp file and SnP component) can be found in the remaining reference sections.
Working with Data Files

SnP Format

These files contain small-signal G-, H-, S-, Y-, or Z-network parameters described by frequency-dependent linear network parameters for 1- to 10-port components. The 2-port component files can also contain frequency-dependent noise parameters. This data file format is also known as Touchstone format.

An .snp file can be used with an SnP component to model the behavior of a linear model using S-parameters. The file contains the S-parameters, the component is placed within the schematic.

This section describes:

• Choosing an .snp file for use with an Snp component
• An overview of the SnP file
• The basic SnP format
• Adding noise to a 2-port Snp file
• The basic SnP format applied to G-, H-, S-, Y-, and Z-parameters, plus examples of each

Linking an .snp File to an SnP Component

To link a file to the component:

1. Add an SnP component to your schematic. It can be found in the Data Items library.
2. Select the File parameter. Ensure that the Parameter Entry Mode is set to Network Parameter File Name.
3. In the File Name field, enter the name of the file you want to use:
   • You can type the name directly in the field
   • Click Data files list to locate a file in the current project (or any files located based on the setting of the DATAFILES variable in de_sim.cfg).
   • Click Browse to locate a file outside the current project
   • Click Copy template to select an example file that you can customize
4. After you select a file, click Edit if you want to view the file or change its contents.
For instructions on how to set the remaining parameters, click Help in the open component dialog box.

Overview

SnP (also known as Touchstone) data files are ASCII text files in which data appears line by line, one line per data point, in increasing order of frequency. Each line of data consists of a frequency value and one or more pairs of values for the magnitude and phase of each S-parameter at that frequency. Values are separated by one or more spaces, tabs or commands. Comments are preceded by an exclamation mark (!). Comments can appear on separate lines, or after the data on any line or lines.

Recommendations for filenames are:

1-port filename.s1p
2-port filename.s2p

Up to 10 ports can be defined.

You can specify the following parameters in an .snp file:

S = Scattering parameters
Y = Admittance parameters
Z = Impedance parameters
H = Hybrid-h parameters
G = Hybrid-g parameters

The following sections discusses the content and format of network parameter files as input for circuit analysis.
Basic SnP File Format

The following example shows the general format for component data files. It consists of:

- An option line
- Data lines
- Comments

The Option Line

The option line, specifying the frequency units and the normalizing impedance, precedes the data lines.

```
  option line— # <frequency unit> <parameter> <format> <R n>
                  <data line>
                  ...
                  <data line>
```

where

- `#` = The delimiter that tells the program you are specifying these parameters
- `frequency unit` = The set of units desired (GHz, MHz, KHz, Hz)
- `parameter` = The parameter desired (S, Y or Z for S1P components; S, Y, Z, G, or H for S2P components; S for S3P or S4P components)
- `format` = The format desired (DB for dB-angle, MA for magnitude angle, or RI for real-imaginary). This does not apply to noise parameters. (Refer to “Adding Noise Parameters to an SnP File” on page 20-12.)
- `R n` = The reference resistance in ohms, where n is a positive number of ohms (the real impedance to which the parameters are normalized).
In summary, the option line should read:

For .s1p files:  
#    GHZ    S    MA    R 50

For .s2p files:  
#    [HZ/KHZ/MHZ/1000]    [S/Y/Z]    [MA/DB/RI]    [R n]

For .s3p/.s4p files:  
#    [HZ/KHZ/MHZ/1000]    [S]    [MA/DB/RI]    [R n]

where square brackets [...] indicate optional information; .../.../... indicates that you select one of the choices; and, n is replaced by a positive number.

**Default Option Line**

The default option line for component data files is:

#    GHZ    S    MA    R 50

**Option Line Examples**

Frequency in GHz, S-parameters in real-imaginary format, normalized 100 ohms:

#    GHz    S    RI    R 100

Frequency in KHz, Y-parameters in real-imaginary format, normalized 100 ohms:

#    KHz    Y    RI    R 100

Frequency in Hz, Z-parameters in magnitude-degree format, normalized to 1 ohm:

#    Hz    Z    MA    R 1

Frequency in KHz, H-parameters in real-imaginary format normalized to 1 ohm:

#    KHz    H    RI    R 1

Frequency in Hz, G-parameters in magnitude-degree, format normalized to 1 ohm:

#    Hz    G    MA    R 1
Data Lines

Data lines contain the data of interest. A special format is used for 2-port data files where all of the network parameter data for a single frequency is listed on one line. The order of the network parameters is:

N11, N21, N12, N22

For 3-port or higher data files, the network parameters appear in the file in a matrix form, each row starting on a separate line. A maximum of four network parameters (with 2 real numbers for each) appear on any line. The remaining network parameters are continued on as many additional lines as are needed.

The following sections describe the data-line format for single and multi-port components.

Data-line Formats

When you type the data below the option line, the columns need not line up precisely like those shown. The syntax for entering data is as follows:

1-port Component

Magnitude-Angle format:

(Column: f Mag Ang)

f | S11| <S11

2-port Component

Magnitude-Angle format:

f | S11| <S11 | S21| <S21 | S12| <S12 | S22| <S22

Real-Imaginary format:

f Re{S11} Im{S11} Re{S21} Im{S21} Re{S12} Im{S12} Re{S22} Im{S22}

dB-Angle format:

f 20log10|x11| <x11 20log10|x21| <x21 20log10|x12| <x12 20log10|x22| <x22
where

\[
\begin{align*}
  x & \quad = \quad S/ Y/ Z/ H/ G \\
  f & \quad = \quad \text{frequency}
\end{align*}
\]

**Note**  For each s1p and s2p file format, the data must be on one line.

---

**3-port Component**

Magnitude-Angle format:

(Column: \( f \quad \text{Mag} \quad \text{Ang} \quad \text{Mag} \quad \text{Ang} \quad \text{Mag} \quad \text{Ang} \)

\[
\begin{array}{lllllll}
  f & |S11| & \angle S11 & |S12| & \angle S12 & |S13| & \angle S13 \\
  |S21| & \angle S21 & |S22| & \angle S22 & |S23| & \angle S23 \\
  |S31| & \angle S31 & |S32| & \angle S32 & |S33| & \angle S33 \\
\end{array}
\]

**4-port Component**

Magnitude-Angle format:

(Column: \( f \quad \text{Mag} \quad \text{Ang} \quad \text{Mag} \quad \text{Ang} \quad \text{Mag} \quad \text{Ang} \quad \text{Mag} \quad \text{Ang} \)

\[
\begin{array}{lllllllll}
  f & |S11| & \angle S11 & |S12| & \angle S12 & |S13| & \angle S13 & |S14| & \angle S14 \\
  |S21| & \angle S21 & |S22| & \angle S22 & |S23| & \angle S23 & |S24| & \angle S24 \\
  |S31| & \angle S31 & |S32| & \angle S32 & |S33| & \angle S33 & |S34| & \angle S34 \\
  |S41| & \angle S41 & |S42| & \angle S42 & |S43| & \angle S43 & |S44| & \angle S44 \\
\end{array}
\]

where

\[
\begin{align*}
  f & \quad = \quad \text{Frequency} \\
  \text{Mag} & \quad = \quad \text{Magnitude of S-parameter Sij} \\
  \text{Ang} & \quad = \quad \text{Angle of S-parameter Sij}
\end{align*}
\]

**Adding Comments to Data Files**

You can document your data files by preceding a comment with the exclamation mark (!) on any line. A comment can be the only entry on a line or can follow the data on any line.
Adding Noise Parameters to an SnP File

Noise parameters can be included in SnP 2-port data files. Noise data can follow G-, H-, S-, Y-, or Z-parameters described for each frequency. The x values are data.

Each line of a noise parameter has the following five entries:

```
x1  x2  x3  x4  x5
```

where

```
x1 = Frequency in units. The first point of noise data must have a frequency less than the frequency of the last S-parameter frequency
x2 = Minimum noise figure in dB
x3 = Source reflection coefficient to realize minimum noise figure (MA)
x4 = Phase in degrees of the reflection coefficient (MA)
x5 = Normalized effective noise resistance. The system simulator requires this parameter to meet physical requirements. If the user-supplied x5 value is less than allowed for this requirement, then the system simulator will force this x5 value to the lowest physical limit.
```

**Note**  The frequencies for noise parameters and network parameters need not match. The only requirement is that the lowest noise-parameter frequency be less than or equal to the highest network-parameter frequency. This allows the file processor to determine where network parameters end and noise parameters begin.

The source reflection coefficient and effective noise resistance are normalized to the same resistance as specified for the network parameters.
Example File Containing Noise Data

This is an example of a data file with noise data:

```plaintext
! NEC710
# GHZ S MA R 50
 2  .95  -26  3.57  157  .04  76  .66  -14
 22 .60  -144 1.30  40  .14  40  .56  -85
! NOISE PARAMETERS
  4  .7   .64  69 .38
 18 2.7 .46  -33 .40
```

Applying the SnP Format, and Examples

In this section are formatting references and examples for:

- G-parameter files
- H-parameter files
- S-parameter files
- Y- and Z-parameter files

Guidelines

- The optimum source reflection coefficient and the normalized effective noise resistance are assumed to be with respect to the normalizing resistance value (appearing after the R keyword) on the header line.
- The frequencies for noise parameters and G-, H-, S-, Y-, or Z-parameters (network parameters) do not have to match. The only requirement is that the lowest noise parameter frequency be less than or equal to the highest network parameter frequency. This allows the file processor to determine where G-, H-, S-, Y-, or Z-parameters end and noise parameters begin.
Working with Data Files

**G-parameter Files**

G-parameter files (Hybrid-g parameters) use MA or RI format. They are strictly 2-port files. G-parameter measurements are:

- **G11**: input admittance (port 2 open)
- **G22**: output impedance (port 1 shorted)
- **G21**: forward voltage gain (port 2 open)
- **G12**: reverse current gain (port 1 shorted)

**G-parameter MA and RI File Formats**

```
# frequency_unit G MA R impedance
```

<table>
<thead>
<tr>
<th>#</th>
<th>frequency_unit</th>
<th>G</th>
<th>MA</th>
<th>R</th>
<th>impedance</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>reG11</td>
<td>imG11</td>
<td>reG21</td>
<td>imG21</td>
<td>reG12</td>
</tr>
</tbody>
</table>

**G-parameter File Example**

```
! symbol freq-unit parameter-type data-format keyword impedance-ohms
# KHZ G MA R 1
2 .95  -26  3.57  157  .04  76  .66  -14
3 .93  -40  3.53  147  .05  69  .65  -20
4 .89  -52  3.23  136  .06  62  .63  -26
```
**H-parameter Files**

H-parameter files (Hybrid-h parameters) use MA or RI format. They are strictly 2-port files. H-parameter measurements are:

- H11  input impedance (port 2 shorted)
- H22  output admittance (port 1 open)
- H21  forward current gain (port 2 shorted)
- H12  reverse voltage gain (port 1 open)

**H-parameter File Example**

```
! symbol freq-unit parameter-type data-format keyword impedance-ohms
# KHZ H MA R I
! freq  magH11 angH11 magH21 angH21 magH12 angH12 magH22 angH22
2  .95  -26  3.57  157  .04  76  .66  -14
3  .93  -40  3.53  147  .05  69  .65  -20
4  .89  -52  3.23  136  .06  62  .63  -26
```
S-parameter Files

S-parameter files (scattering parameters) can have MA, RI, or DB format for files with 1 to 99 ports.

S-parameter 1-port MA, RI, and DB File Formats

```
# frequency_unit S MA R impedance
freq  magS11  angS11

# frequency_unit S RI R impedance
freq  reS11  imS11

# frequency_unit S DB R impedance
freq  dbS11  angS11
```

S-parameter 2-port MA, RI, and DB File Formats

```
# frequency_unit S MA R impedance
freq  magS11  angS11  magS21  angS21  magS12  angS12  magS22  angS22

# frequency_unit S RI R impedance
freq  reS11  imS11  reS21  imS21  reS12  imS12  reS22  imS22

# frequency_unit S DB R impedance
freq  dbS11  angS11  dbS21  angS21  dbS12  angS12  dbS22  angS22
```

S-parameter 3-port MA, RI, and DB File Formats

```
# frequency_unit S MA R impedance
freq  magS11  angS11  magS12  angS12  magS13  ! 1st row
  magS21  angS21  magS22  angS22  magS23  ! 2nd row
  magS31  angS31  magS32  angS32  magS33  ! 3rd row

# frequency_unit S RI R impedance
freq  reS11  imS11  reS12  imS12  reS13  ! 1st row
  reS21  imS21  reS22  imS22  reS23  ! 2nd row
  reS31  imS31  reS32  imS32  reS33  ! 3rd row

# frequency_unit S DB R impedance
freq  dbS11  angS11  dbS12  angS12  dbS13  ! 1st row
  dbS21  angS21  dbS22  angS22  dbS23  ! 2nd row
  dbS31  angS31  dbS32  angS32  dbS33  ! 3rd row
```
S-parameter 4-port MA, RI, and DB File Formats

# frequency_unit S MA R impedance
! 1st row
freq magS11 angS11 magS12 angS12 magS13 angS13 magS14 angS14
   magS21 angS21 magS22 angS22 magS23 angS23 magS24 angS24 ! 2nd row
   magS31 angS31 magS32 angS32 magS33 angS33 magS34 angS34 ! 3rd row
   magS41 angS41 magS42 angS42 magS43 angS43 magS44 angS44 ! 4th row

# frequency_unit S RI R impedance
freq reS11 imS11 reS12 imS12 reS13 imS13 reS14 imS14 ! 1st row
   reS21 imS21 reS22 imS22 reS23 imS23 reS24 imS24 ! 2nd row
   reS31 imS31 reS32 imS32 reS33 imS33 reS34 imS34 ! 3rd row
   reS41 imS41 reS42 imS42 reS43 imS43 reS44 imS44 ! 4th row

# frequency_unit S DB R impedance
freq dbS11 angS11 dbS12 angS12 dbS13 angS13 dbS14 angS14 ! 1st row
   dbS21 angS21 dbS22 angS22 dbS23 angS23 dbS24 angS24 ! 2nd row
   dbS31 angS31 dbS32 angS32 dbS33 angS33 dbS34 angS34 ! 3rd row
   dbS41 angS41 dbS42 angS42 dbS43 angS43 dbS44 angS44 ! 4th row

S-parameter 1-port File Example

! symbol freq-unit parameter-type data-format keyword impedance-ohms
# MHZ S MA R 50
! freq magS11 angS11 (commented header line)
  2.000 0.894 -12.136
  3.000 0.893 -18.179
  4.000 0.891 -24.193

S-parameter 5- to 99-port File Formats

These file formats appear in a matrix form similar to the 3- and 4-port files, except that only four S-parameters (with 2 real numbers for each) can appear on a given line. Therefore, the remaining S-parameters in that row of the S-matrix continue on the next line of the file.

Each row of the S-matrix must begin on a new line of the file. The first line of the first row of the S-matrix begins with the frequency value.
Working with Data Files

S-parameter 10-port File Example (at One Frequency)

```
# frequency_unit S MA R impedance
freq magS11 angS11 magS12 angS12 magS13 angS13 magS14 angS14 ! 1st row
magS15 angS15 magS16 angS16 magS17 angS17 magS18 angS18
magS19 angS19 magS1,10 angS1,10
magS21 angS21 magS22 angS22 magS23 angS23 magS24 angS24 ! 2nd row
magS29 angS29 magS2,10 angS2,10
magS31 angS31 magS32 angS32 magS33 angS33 magS34 angS34 ! 3rd row
magS35 angS35 magS36 angS36 magS37 angS37 magS38 angS38
magS39 angS39 magS3,10 angS3,10
magS41 angS41 magS42 angS42 magS43 angS43 magS44 angS44 ! 4th row
magS45 angS45 magS46 angS46 magS47 angS47 magS48 angS48
magS49 angS49 magS4,10 angS4,10
magS51 angS51 magS52 angS52 magS53 angS53 magS54 angS54 ! 5th row
magS55 angS55 magS56 angS56 magS57 angS57 magS58 angS58
magS59 angS59 magS5,10 angS5,10
magS61 angS61 magS62 angS62 magS63 angS63 magS64 angS64 ! 6th row
magS65 angS65 magS66 angS66 magS67 angS67 magS68 angS68
magS69 angS69 magS6,10 angS6,10
magS71 angS71 magS72 angS72 magS73 angS73 magS74 angS74 ! 7th row
magS75 angS75 magS76 angS76 magS77 angS77 magS78 angS78
magS79 angS79 magS7,10 angS7,10
magS81 angS81 magS82 angS82 magS83 angS83 magS84 angS84 ! 8th row
magS85 angS85 magS86 angS86 magS87 angS87 magS88 angS88
magS89 angS89 magS8,10 angS8,10
magS91 angS91 magS92 angS92 magS93 angS93 magS94 angS94 ! 9th row
magS95 angS95 magS96 angS96 magS97 angS97 magS98 angS98
magS99 angS99 magS9,10 angS9,10
!10th row
magS10,1 angS10,1 magS10,2 angS10,2 magS10,3 angS10,3 magS10,4 angS10,4
magS10,5 angS10,5 magS10,6 angS10,6 magS10,7 angS10,7 magS10,8 angS10,8
magS10,9 angS10,9 magS10,10 angS10,10
```
Linear 1-port (.s1p) File Example

# GHZ S RI R 50.0
1.00000000 0.9488 -0.2017
1.50000000 0.9077 -0.3125
2.00000000 0.8539 -0.4165
2.50000000 0.7884 -0.5120
3.00000000 0.7124 -0.5978
3.50000000 0.6321 -0.6546
4.00000000 0.5479 -0.7013
4.50000000 0.4701 -0.7380
5.00000000 0.3904 -0.7663
5.50000000 0.3302 -0.7778
6.00000000 0.2702 -0.7848
6.50000000 0.2041 -0.7890
7.00000000 0.1389 -0.7878
7.50000000 0.0894 -0.7849
8.00000000 0.0408 -0.7789
8.50000000 0.0134 -0.7649
9.00000000 0.0654 -0.7471
9.50000000 0.1094 -0.7319
10.0000000 0.1518 -0.7140

Linear 2-port (.s2p) File Example

# GHZ S RI R 50.0
1.0000 0.3926 -0.1211 -0.0003 -0.0003 -0.0021 -0.0021 0.3926 -0.1211
2.0000 0.3517 -0.3054 -0.0096 -0.0298 -0.0298 0.3517 -0.3054
10.000 0.3419 0.3336 -0.0134 0.0379 -0.0134 0.0379 0.3419 0.3336

!Noise params
1.0000 2.0000 -0.1211 -0.0003 -0.4
2.0000 2.5000 -0.3054 -0.0096 -0.45
3.0000 3.0000 -0.6916 -0.6933 -0.5
4.0000 3.5000 -0.3756 0.4617 -0.55
5.0000 4.0000 0.3880 0.6848 -0.6
6.0000 4.5000 0.0343 0.0383 -0.65
7.0000 5.0000 0.6916 0.6933 -0.7
8.0000 5.5000 0.5659 0.1000 -0.75
9.0000 6.0000 0.4145 0.0307 -0.8
10.0000 6.5000 0.3336 0.0134 -0.85
Working with Data Files

**Linear 3-port (.s3p) File Example**

```
#   GHZ S MA R 50.0
! POWER DIVIDER, 3-PORT
5.00000 0.24254 136.711 0.68599 -43.3139 0.68599 -43.3139
   0.68599 -43.3139 0.08081 66.1846 0.28009 -59.1165
   0.68599 -43.3139 0.28009 -59.1165 0.08081 66.1846
6.00000 0.20347 127.652 0.69232 -52.3816 0.69232 -52.3816
   0.69232 -52.3816 0.05057 52.0604 0.22159 -65.1817
   0.69232 -52.3816 0.22159 -65.1817 0.05057 52.0604
7.00000 0.15848 118.436 0.69817 -61.6117 0.69817 -61.6117
   0.69817 -61.6117 0.02804 38.6500 0.16581 -71.2358
   0.69817 -61.6117 0.16581 -71.2358 0.02804 38.6500
```

**Linear 4-port (.s4p) File Example**

```
#   GHZ S MA R 50
5.00000 0.60262 161.240 0.40611 -42.2029 0.42918 -66.5876 0.53640 -79.3473
   0.40611 -42.2029 0.60262 161.240 0.53640 -79.3473 0.42918 -66.5876
   0.42918 -66.5876 0.53640 -79.3473 0.60262 161.240 0.40611 -42.2029
   0.53640 -79.3473 0.42918 -66.5876 0.40611 -42.2029 0.60262 161.240
6.00000 0.57701 150.379 0.40942 -44.3428 0.41011 -81.2449 0.57554 -95.7731
   0.40942 -44.3428 0.57701 150.379 0.57554 -95.7731 0.41011 -81.2449
   0.41011 -81.2449 0.57554 -95.7731 0.57701 150.379 0.40942 -44.3428
   0.57554 -95.7731 0.41011 -81.2449 0.40942 -44.3428 0.57701 150.379
7.00000 0.50641 136.693 0.45378 -46.4151 0.37845 -99.0918 0.62802 -114.196
   0.45378 -46.4151 0.50641 136.693 0.62802 -114.196 0.37845 -99.0918
   0.37845 -99.0918 0.62802 -114.196 0.50641 136.693 0.45378 -46.4151
   0.62802 -114.196 0.37845 -99.0918 0.45378 -46.4151 0.50641 136.693
```
Y- and Z-parameter Files

Immittance parameters are specified in MA or RI format, where the ‘#’ line has ‘Y’ for admittance and ‘Z’ for impedance. Both are normalized to the reference resistance.

Y- (Z-) Parameter 1-port MA and RI File Formats

# frequency_unit Y MA R impedance
freq magY11 angY11

# frequency_unit Y RI R impedance
freq reY11 imY11

Y- (Z-) Parameter 2-port MA and RI File Formats

# frequency_unit Y MA R impedance
freq magY11 angY11 magY21 angY21 magY12 angY12 magY22 angY22

# frequency_unit Y RI R impedance
freq reY11 imY11 reY21 imY21 reY12 imY12 reY22 imY22
Working with Data Files

**Y- (Z-) Parameter 3-port MA and RI File Formats**

```
freq  magY11  angY11  magY12  angY12  magY13  angY13  ! 1st row
      magY21  angY21  magY22  angY22  magY23  angY23  ! 2nd row
      magY31  angY31  magY32  angY32  magY33  angY33  ! 3rd row

# frequency_unit  Y  RI  R  impedance
freq  reY11  imY11  reY12  imY12  reY13  imY13  ! 1st row
      reY21  imY21  reY22  imY22  reY23  imY23  ! 2nd row
      reY31  imY31  reY32  imY32  reY33  imY33  ! 3rd row
```

**Y- (Z-) Parameter 4-port MA and RI File Formats**

```
freq  magY11  angY11  magY12  angY12  magY13  angY13  magY14  angY14  ! 1st row
      magY21  angY21  magY22  angY22  magY23  angY23  magY24  angY24  ! 2nd row
      magY31  angY31  magY32  angY32  magY33  angY33  magY34  angY34  ! 3rd row
      magY41  angY41  magY42  angY42  magY43  angY43  magY44  angY44  ! 4th row

# frequency_unit  Y  MA  R  impedance
freq  reY11  imY11  reY12  imY12  reY13  imY13  reY14  imY14  ! 1st row
      reY21  imY21  reY22  imY22  reY23  imY23  reY24  imY24  ! 2nd row
      reY31  imY31  reY32  imY32  reY33  imY33  reY34  imY34  ! 3rd row
      reY41  imY41  reY42  imY42  reY43  imY43  reY44  imY44  ! 4th row
```
Y- (Z-) Parameter 3-port File Example

! symbol freq-unit parameter-type data-format keyword impedance-ohms
# GHz Y MA R l
! freq magY11 angY11 magY12 angY12 magY13 angY13 ! 1st line
! magY21 angY21 magY22 angY22 magY23 angY23 ! 2nd line
! magY31 angY31 magY32 angY32 magY33 angY33 ! 3rd line

<table>
<thead>
<tr>
<th>freq</th>
<th>magY11</th>
<th>angY11</th>
<th>magY12</th>
<th>angY12</th>
<th>magY13</th>
<th>angY13</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>83.122</td>
<td>0.008</td>
<td>8.5e-04</td>
<td>-86.740</td>
<td>0.007</td>
<td>-98.037</td>
</tr>
<tr>
<td></td>
<td>0.046</td>
<td>-12.740</td>
<td>0.005</td>
<td>36.580</td>
<td>0.049</td>
<td>171.554</td>
</tr>
<tr>
<td></td>
<td>0.046</td>
<td>177.588</td>
<td>0.004</td>
<td>-152.638</td>
<td>0.050</td>
<td>0.134</td>
</tr>
<tr>
<td>8</td>
<td>79.068</td>
<td>0.016</td>
<td>0.002</td>
<td>-84.015</td>
<td>0.014</td>
<td>-102.924</td>
</tr>
<tr>
<td></td>
<td>0.049</td>
<td>-23.015</td>
<td>0.006</td>
<td>52.828</td>
<td>0.051</td>
<td>164.123</td>
</tr>
<tr>
<td></td>
<td>0.048</td>
<td>175.827</td>
<td>0.005</td>
<td>-139.640</td>
<td>0.052</td>
<td>-0.004</td>
</tr>
<tr>
<td>12</td>
<td>73.501</td>
<td>0.025</td>
<td>0.003</td>
<td>-81.736</td>
<td>0.023</td>
<td>-109.374</td>
</tr>
<tr>
<td></td>
<td>0.058</td>
<td>-36.736</td>
<td>0.009</td>
<td>58.596</td>
<td>0.058</td>
<td>152.007</td>
</tr>
<tr>
<td></td>
<td>0.055</td>
<td>169.129</td>
<td>0.007</td>
<td>-136.047</td>
<td>0.059</td>
<td>-5.349</td>
</tr>
<tr>
<td>18</td>
<td>65.138</td>
<td>0.036</td>
<td>0.004</td>
<td>-71.761</td>
<td>0.033</td>
<td>-119.900</td>
</tr>
<tr>
<td></td>
<td>0.059</td>
<td>-54.761</td>
<td>0.013</td>
<td>72.274</td>
<td>0.052</td>
<td>137.118</td>
</tr>
<tr>
<td></td>
<td>0.052</td>
<td>162.979</td>
<td>0.010</td>
<td>-121.976</td>
<td>0.055</td>
<td>-6.677</td>
</tr>
</tbody>
</table>
Y- (Z-) Parameter 5- to 99-port File Formats

These file formats appear in a matrix form similar to 3- and 4-port files. Only four Y-, or Z-parameters (with 2 real numbers for each) can appear on a given line; therefore, the remaining parameters in that row of the matrix continue on the next line of the file. Each row of the Y-matrix must begin on a new line of the file. The first line of the first row of the Y-matrix begins with the frequency value. The actual Y- (Z-) parameter value is obtained by dividing (multiplying) the file entry with the reference resistance.

Y- (Z-) Parameter 10-port File Example (at one frequency)

```plaintext
# frequency_unit Y MA R impedance
freq magY11 angY11 magY12 angY12 magY13 angY13 magY14 angY14 ! 1st row
magY15 angY15 magY16 angY16 magY17 angY17 magY18 angY18
magY19 angY19 magY1,10 ngY1,10

magY21 angY21 magY22 angY22 magY23 angY23 magY24 angY24 ! 2nd row
magY25 angY25 magY26 angY26 magY27 angY27 magY28 angY28
magY29 angY29 magY2,10 angY2,10

magY31 angY31 magY32 angY32 magY33 angY33 magY34 angY34 ! 3rd row
magY35 angY35 magY36 angY36 magY37 angY37 magY38 angY38
magY39 angY39 magY3,10 angY3,10

magY41 angY41 magY42 angY42 magY43 angY43 magY44 angY44 ! 4th row
magY45 angY45 magY46 angY46 magY47 angY47 magY48 angY48
magY49 angY49 magY4,10 angY4,10

magY51 angY51 magY52 angY52 magY53 angY53 magY54 angY54 ! 5th row
magY55 angY55 magY56 angY56 magY57 angY57 magY58 angY58
magY59 angY59 magY5,10 angY5,10

magY61 angY61 magY62 angY62 magY63 angY63 magY64 angY64 ! 6th row
magY65 angY65 magY66 angY66 magY67 angY67 magY68 angY68
magY69 angY69 magY6,10 angY6,10

magY71 angY71 magY72 angY72 magY73 angY73 magY74 angY74 ! 7th row
magY75 angY75 magY76 angY76 magY77 angY77 magY78 angY78
magY79 angY79 magY7,10 angY7,10

magY81 angY81 magY82 angY82 magY83 angY83 magY84 angY84 ! 8th row
magY85 angY85 magY86 angY86 magY87 angY87 magY88 angY88
magY89 angY89 magY8,10 angY8,10

magY91 angY91 magY92 angY92 magY93 angY93 magY94 angY94 ! 9th row
```
magY95 angY95 magY96 angY96 magY97 angY97 magY98 angY98
magY99 angY99 magY9,10 angY9,10
!10th row
magY10,1 angY10,1 magY10,2 angY10,2 magY10,3 angY10,3 magY10,4 angY10,4
magY10,5 angY10,5 magY10,6 angY10,6 magY10,7 angY10,7 magY10,8 angY10,8
magY10,9 angY10,9 magY10,10 angY10,10
S2D Format

The small- or large-signal S-parameter (.s2d) file is a system input file you create from an S-parameter file, inserting the MDIF format. In the .s2d file, you describe small-signal data, optional noise data, optional nonlinear data, and optional intermodulation data. You can describe nonlinearity as a function of drive power or (using GCOMP Data Items) with some combination of third-order intercept point, 1dB gain compression, saturated power, and gain compression at saturation.

The general .s2d file format is:

```
REM MDIF Basic Syntax Example
VAR TEMP =xxx
BEGIN ACDATA ! required AC data block
....
.... ACDATA block data lines ....
.... END
BEGIN NDATA! optional noise data block
....
.... NDATA block data lines ....
....
END;
VAR TEMP=YYY ! optional
BEGIN ACDATA
....
.... block data lines ....
....
END
BEGIN GCOMP1 ! or GCOMP2 thru GCOMP7 ! optional
.... data goes here
END
```
BEGIN IMTDATA! optional
....data goes here
END

Guidelines for .s2d

• Basic MDIF syntax contains four reserved words:
  • VAR begins an independent variable definition line, in the form
    VAR <name> = <value>.
  • BEGIN <blockname> signals the beginning of a data block.
  • END signals the conclusion of a data block.
  • REM or the comment symbol (!) at the beginning of a line signifies a
    comment.
  • VAR statements are used to specify multidimensional data, i.e. 2 or more
    independent variables. The value of a VAR statement can be a number or an
    alphanumeric string.
  • Comments can be inserted on any line within the file and must be preceded by a
    comment symbol (!).
  • Multiple sets of data (ACDATA, NDATA, GCOMP, IMTDATA) can be used with
    VAR statements before or after any data set.
  • The file data set is made up of data blocks, each separated by BEGIN and END
    statements. Ten different types of data blocks are allowed:
  • Only one of the GCOMPi blocks can be used per file.

    ACDATA     lists small-signal noise parameters vs. frequency (required)
    NDATA      lists noise parameters vs. frequency (optional)
    GCOMP1     lists output 3rd order intercept (IP3) (optional)
    GCOMP2     lists output 1dB gain compression power (IDBC) (optional)
    GCOMP3     lists output IP3 and 1DBC (optional)
    GCOMP4     lists output IP3 and output saturation power (PS) and gain
                compression at saturation (GCS) (optional)
    GCOMP5     lists output 1DBC, PS, and GCS (optional)
Working with Data Files

GCOMP6  lists output IP3, 1DBC, PS, and GCS (optional)
GCOMP7  lists S21 as a function of input power at a single frequency (optional)
IMTDATA used for a 2-port frequency converter to give the single-tone intermodulation table (optional)

- One or more spaces or tabs separate entries on a line. Names can be mixed case—the file reader will store them as lowercase.
The ACDATA Block

The ACDATA block allows you to specify the small-signal characteristics of the 2-port.

- General format:
  
  BEGIN ACDATA
  # AC (.......)! this is the option line
  % ....! this is the format line
  ..... data goes here
  END

- Option line syntax:

  #AC( freq-unit parm-type parm-format R xx FC m b )

  freq-unit: HZ, KHZ, MHZ, or GHZ sets the frequency units
  options:

  parm-type: S, Z, Y, H, or G sets the 2-port network parameter option
  options:

  parm-format: MA, DB, RI, VDB sets the format for the four network parameters
  options:
  
  MA— for magnitude and angle (deg.)
  DB— for 20log10(MA) and angle (deg.)
  RI— for real and imaginary
  VDB— for n11 and n22 as VSWR and angle (deg.) and n12 and n21 as DB and angle (deg.)
  R xx— where xx = normalization resistance
  FC mbs— for input to output frequency conversion such that Fout = m•Fin + b where m is for frequency multiplication, b is for frequency translation
Working with Data Files

- **Example:**

  
  ```
  #AC (MHZ S MA R 50 FC 0 30)
  
  would set the frequency units to MHZ, 2-port parameters to S, 2-port parameter format to magnitude and angle, reference resistance to 50 ohms, frequency conversion to 30 MHZ of constant output frequency.
  
  FC  1  0     Sets the output frequency equal to the input frequency   
                 ( Fout =1 • Fin + 0 )
  
  FC -1  30    Sets the output frequency to 30 minus the input frequency  
                 ( Fout =-1 • Fin + 30 )
  
  - **Format line:**

  ```
  % F n11x n11y n21x n21y n12x n12y n22x n22y
  
  This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred.

  ```
  F       = frequency data column
  n11x, n11y = the 11 data pair for the 2 port data matrix
  n21x, n21y = the 21 data pair
  n12x, n12y = the 12 data pair
  n22x, n22y = the 22 data pair
  
  where these data pairs belong to the 2-port characterization matrix:

  ```
  \[
  \begin{bmatrix}
  n_{11} & n_{12} \\
  n_{21} & n_{22}
  \end{bmatrix}
  \]
• Examples

The following examples illustrate the ACDATA block with various data options:

2-port with 50-ohm S-parameters:

BEGIN ACDATA
# AC ( GHZ S DB R 50 FC 1 0 )
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-db S11-deg S21-db S21-deg S12-db S12-deg S22-db S22-deg
1.0000 -15 45 -8 25 -20 -15 -12 10
2.0000 -16 25 -9 30 -20 -15 -12 20
3.0000 -17 10 -10 35 -20 -15 -11 30
END

2-port frequency converter with variable RF freq, fixed LO freq, variable IF freq, fixed LO power, and RF-to-IF 2-port 50-ohm S-parameters:

BEGIN ACDATA
# AC( GHZ S DB R 50 FC -1 4 )! this gives an
! IF = 4 - RF
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-db S11-deg S21-db S21-deg S12-db S12-deg S22-db S22-deg
1.0000 -12 0 -8 0 -20 0 -13 0
2.0000 -13 0 -9 0 -20 0 -12 0
3.0000 -14 0 -10 0 -20 0 -13 0
END
Working with Data Files

The NDATA Block
The NDATA block allows you to specify the small-signal noise characteristics of the 2-port.

• General format:

  BEGIN NDATA
  # AC ( ....... ) ! this is the option line
  % .... ! this is the format line
  ..... data goes here
  END

• Option line syntax:

  #AC( freq-unit parm-type parm-format   R xx )

freq-unit options: HZ, KHZ, MHZ, or GHZ sets the frequency units
parm-type options: S only, source reflection coefficient
parm-format options: MA, DB, RI sets the format for optimum source match
  MA - for magnitude and angle (deg.)
  DB - for 20log10(MA) and angle (deg.)
  RI - for real and imaginary
  R xx - where xx = normalization resistance for the source match and noise resistance
• Format line syntax:

%  F   nfmin  n11x  n11y  rn

This line gives the order for the data in the lines to follow. All of the keywords
shown must be given in the format line. While the order of keywords is arbitrary,
the order shown is preferred.

F       = frequency data column
nfmin   = for minimum noise figure data
n11x, n11y = for the optimum source reflection coefficient for minimum
noise figure
rn       = for the equivalent normalized input noise resistance of the
2-port. The system simulator requires this parameter to meet
physical requirements. If the user-supplied rn value is less
than allowed for this requirement, then the system simulator
will force this rn value to the lowest physical limit.

For more information on noise figure, refer to “Noise Analysis” on page 6-14.

The following is an example of the NDATA block:

BEGIN  NDATA
#  ACDATA  (  GHz  S  RI  R 50  )
%  F   nfmin  n11x  n11y  rn
1.0000   2.0000   -0.1211  -0.0003   .4
2.0000   2.5000   -0.3054  -0.0096   .45
3.0000   3.0000   -0.6916  -0.6933   .5
END
Working with Data Files

The GCOMP1 Block
The GCOMP1 data block for the .s2d data file allows you to specify the 2-port 3rd order output intercept (IP3). No option line is used.

- General format:
  ```
  BEGIN GCOMP1
  %  IP3 ! this is the format line, required
  .....  data goes here
  END
  ```

- Format line (required):
  ```
  %  IP3
  ```

- Example:
  ```
  BEGIN   GCOMP1
  %  IP3
  25  ! this sets the output IP3 to 25 dBm
  END
  ```
The GCOMP2 Block

The GCOMP2 data block for the .s2d data file allows you to specify the 2-port output power at 1 dB gain compression. No option line is used.

- General format:
  
  BEGIN GCOMP2
  
  % 1DBC ! this is the format line, required.
  
  ..... data goes here
  
  END

- Format line (required):
  
  % 1DBC

- Example:

  BEGIN   GCOMP2
  
  % 1DBC

  15 ! this sets the output power for 1 dB gain

  ! compression at 15 dBm

  END
Working with Data Files

The GCOMP3 Block

The GCOMP3 data block for the .s2d data file allows you to specify the 2-port output IP3 and 1DBC simultaneously. No option line is used.

- General format:
  BEGIN GCOMP3
  % 1DBC   IP3 ! this is the format line
  ..... data goes here
  END

- Format line:
  % 1DBC   IP3

  This line gives the order for the data in lines to follow. All keywords shown must be given in the format line; keyword order is arbitrary.

- Example:
  BEGIN   GCOMP3 ! includes IP3 and 1DBC
  %  IP3   1DBC
  25   15
  END
The **GCOMP4** Block

The GCOMP4 data block for the .s2d data file allows you to specify the 2-port output IP3, output power at saturation (PS), and the gain compression at saturation (GCS). No option line is used.

- **General format:**

```
BEGIN GCOMP4
%  IP3   PS   GCS ! this is the format line
.....  data goes here
END
```

- **Format line syntax:**

```
%  IP3  PS  GCS
```

This line gives the order for the data in the lines to follow. All keywords shown must be given in the format line; keyword order is arbitrary.

- **Example:**

```
BEGIN   GCOMP4! output 3rd order intercept occurs at 25 dBm
%  IP3  PS  GCS! output saturation occurs at 20 dBm
  25   20  5! with 5 dB of gain compression
END
```
The GCOMP5 Block

The GCOMP5 data block for the *.s2d data file allows you to specify the 2-port output 1DBC, output power at saturation (PS), and the gain compression at saturation (GCS). No option line is used.

- General format:
  BEGIN GCOMP5
  % 1DBC PS GCS! this is the format line
  ..... data goes here
  END

- Format line syntax:
  % 1DBC PS GCS

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line; keyword order is arbitrary.

- Example:
  BEGIN GCOMP5! output 1 dB gain compression occurs at 15 dBm
  % 1DBC PS GCS! output saturation occurs at 20
  15 20 5! dBm with 5 dB of gain compression
  END
The GCOMP6 Block

The GCOMP6 data block for the *.s2d data file allows you to specify the 2-port output IP3, 1DBC, output power at saturation (PS), and the gain compression at saturation (GCS). No option line is used.

- General format:

  BEGIN GCOMP6
  % IP3 1DBC PS GCS ! this is the format line
  ..... data goes here
  END

- Format line:

  % IP3 1DBC PS GCS

  This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line; keyword order is arbitrary.

- Example:

  BEGIN GCOMP6! output 3rd order intercept
  ! occurs at 25 dBm
  % IP3 1DBC PS GCS! output 1 dB gain compression
  ! occurs at 15 dBm
  25 15 20 5 ! output saturation occurs
  ! at 20 dBm with 5 dB of gain compression.
  END
The GCOMP7 Block

The GCOMP7 data block for the *.s2d data file allows you to specify the 2-port gain compression characteristic by listing the gain and phase as a function of input power in tabular form. Only one frequency point may be specified.

- General format:

```
BEGIN GCOMP7
# AC ( ....... )! this is the option line
% ....! this is format line 1
...... frequency goes here
% ....! this is format line 2
...... data goes here
END
```

- Option line syntax:

```
# AC( freq-dim  parm-type  power-nit  parm_format  R xx )
freq-dim options: HZ, KHZ, MHZ, or GHZ sets the frequency units
parm-type options: S only
power_dim options: DBM only
parm_format options: MA, DB, RI sets the format for the S21
MA-for magnitude and angle (deg.)
DB-for 20\log_{10}( MA) and angle (deg.)
RI-for real and imaginary
R xx-where xx = reference resistance for s-parameters
```

- Format line:

There are two format lines:

```
% F format line 1
% P I N n21x, n21y format line 2
```
where F indicates that the following data point is the frequency (only one frequency can be specified)

PIN is the input power

n21x, n21 y are the S21 data pair

These lines give the order for the data in the lines to follow. All of the keywords shown must be given in the format line. The order of these keywords is arbitrary.

• Example:

BEGIN   GCOMP7! here, the gain compression
          ! characteristic of the 2 port
          ! is described by actual data of
          ! S21 vs. input power
          #  AC( GHZ  S  DBM  RI   R  50.0 )
          %  F
          5.!. Only one frequency point can be specified
          % PINN21xN21y
            -202.6848-0.5367
            -102.5448-0.5031
             2.4521-0.4732
          ! Up to 101 sets of data versus
          ! power can be specified

END
Working with Data Files

Complete .s2d File Example

BEGIN ACDATA
! This line and all lines starting with a comment (!) character are
! Do not have a blank line, or comments as the first line in this file
! ACDATA is a required block of data
! The following is the OPTION line
#AC( GHZ   S RI R 50.0    FC 1 15 )
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S, Z, Y, H, or G; Default is S
! Optional Selection: MA, DB, RI, VDB; Default is MA
! Optional Selection: R xx; where xx=reference resistance; Default R 50.0
! Optional Selection: FC x1 x2 is for frequency conversion: Fout=x1*Fin
! + x2
!       Default is Fout = Fin
% F   n11x n11y n21x n21y n12x n12y n22x n22y
! The above line is the format line showing the order in the following
data lines
! The columns of data can be in any order
1.0000  0.3926  -0.1211  -0.0003  -0.0021  -0.0003  -0.0021  0.3926
  -0.1211
2.0000  0.3517  -0.3054  -0.0096  -0.0298  -0.0096  -0.0298  0.3517
  -0.3054
3.0000  0.0430  -0.5916  -2.6933  -0.1433  -0.5933  -0.1433  0.0430
  -0.5916
4.0000  0.4071  -0.2756   2.4617   0.6234   0.3617   0.4234  0.4071
  -0.2756
5.0000  0.2041   0.2880   2.6848  -0.5367   0.3848  -0.4367  0.2041
  0.2880
6.0000  0.5666   0.0343   2.0383  -0.7437   0.0383  -0.7437  0.5666
  0.0343
BEGIN NDATA

! This is an optional block of data
! AC( GHZ RI S R 50.0 )
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S, Z, Y, H, or G; Default is S
! Optional Selection: MA, DB, or RI; Default is MA
! Optional Selection: R xx; where xx=reference resistance; Default R 50.0
% f nfmin n11x n11y rn
! The above line is the format line showing the order in the following data lines
! The columns of data can be in any order

| 1.0000 | 2.0000 | 0.3926 | -0.1211 | .4  |
| 2.0000 | 2.5000 | 0.3517 | -0.3054 | .45 |
| 3.0000 | 3.0000 | 0.0430 | -0.5916 | .5  |
| 4.0000 | 3.5000 | 0.4071 | -0.2756 | .55 |
| 5.0000 | 4.0000 | 0.2041 |  0.2880 | .6  |
| 6.0000 | 4.5000 | 0.5666 |  0.0343 | .65 |
| 7.0000 | 5.0000 | 0.0430 |  0.6916 | .7  |
| 8.0000 | 5.5000 | 0.3059 |  0.5659 | .75 |
| 9.0000 | 6.0000 | 0.3071 |  0.4145 | .8  |
Working with Data Files

10.0000    6.5000    0.3419    0.3336    .85
END
BEGIN    GCOMP6
! This is an optional block of data
% IP3 1DBC PS GCS
25 15 20 5
END

! In place of a GCOMP1 block of data there can be either of the
! following blocks
! GCOMP1, GCOMP2, GCOMP3, GCOMP4, GCOMP5, GCOMP6, GCOMP7
! An example of each of these are given below
! BEGIN    GCOMP1
! % IP3
! 25
! END
! BEGIN    GCOMP2
! % 1DBC
! 15
! END
! BEGIN    GCOMP3
! % IP3 1DBC
! 25 15
! END
! BEGIN    GCOMP4
! % IP3 PS GCS
! 25 25 5
! END
! BEGIN    GCOMP5
! % 1DBC PS GCS
! 15 25 5
! END
! BEGIN   GCOMP7
! #  AC( GHZ  S  DBM  RI   R  50.0 )
!! Optional Selection:  HZ, KHZ, MKZ, or GHZ; Default is GHZ
!! Optional Selection:   MA, DB, or RI;     Default is MA
!! Optional Selection:   R xx;  where xx = reference resistance;  
Default R 50.
! %  F
!    5.
!! Only one frequency point can be specified
! %  PIN   N21x   N21y
! -20    2.6848   -0.5367
! -10    2.5448   -0.5031
!  0     2.4521   -0.4732
!! An unlimited number of sets of data versus power can be specified
! END
BEGIN  IMTDATA
! Intermodulation table for double balanced mixer #1
! Signal Level (dBm)  LO Level (dBm)
# -10  7
! M x LO ( Horizontal )   N x Signal (Vertical )
% 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
15
!
  99  26  35  39  50  41  53  49  51  42  62  51  60  47
 77  50
  24  0  35  13  40  24  45  28  49  33  53  42  60  47  63
 73  74  74  70  71  64  69  64  69  62  74  62  72  60
 67  64  69  50  77  47  74  44  74  47  75  44  70
 86  90  86  88  88  85  86  85  90  85  85  85
 90  80  90  71  90  68  90  65  88  65  85
 90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  87  90  90  90
 99  95  99  95  99  95  99  95
 90  95  90  90  90  99  90
Working with Data Files

\[
\begin{array}{cccccccc}
99 & 99 & 99 & 99 & 99 & 99 \\
90 & 99 & 90 & 95 & 90 \\
99 & 99 & 99 & 99 \\
90 & 99 & 90 \\
99 & 99 \\
99 \\
\end{array}
\]

END

P2D Format

The large-signal S-parameter (.p2d) file is a system input file you create from an S-parameter file, inserting the MDIF format.

You use the .p2d file to characterize the component by a complete set of large-signal S-parameters, accounting for power dependence of S21, S11, S22, and S12, plus optional small-signal data, optional noise data, and optional intermodulation data.

A .p2d file can also be created via simulation by placing a P2D simulation component from the LSSP Simulation Library in the design.

The general file format is:

\[
\begin{array}{l}
\text{REM MDIF Basic Syntax Example} \\
\text{VAR TEMP = xxx} \\
\text{BEGIN ACDATA} \\
\text{....} \\
\text{.... ACDATA block data lines ....} \\
\text{.... small and large-signal data goes here} \\
\text{....} \\
\text{.... for large-signal data with up to 101 sets of power data per frequency} \\
\text{END} \\
\text{BEGIN NDATA} \\
\text{....}
\end{array}
\]

20-46  P2D Format
.... NDATA block data lines ....
....
END

VAR .... new varname and value
BEGIN ... new block name
....
.... block data lines ....
....
END
BEGIN IMTDATA
.... data goes here
END

Guidelines for .p2d

1. Basic MDIF syntax contains four reserved words. VAR begins an independent variable definition line, in the form VAR <name> = <value>. BEGIN <blockname> signals the beginning of a data block, and END signals the conclusion of a data block. A line beginning with REM or the comment symbol (!) will be assumed as comments.

2. VAR statements are used to specify multidimensional data, i.e. 2 or more independent variables. The value of a VAR statement can be a number of an alphanumeric string. Typical VAR statements are:

   VAR TEMP = 30
   VAR MeasId = NEC 710 Lot 45G
   VAR Date = Fri Apr 8 11:15:53 PDT 1988

3. Multiple sets of data (ACDATA, NDATA, IMTDATA) can be used with VAR statements used before or after any data set.
4. The file data set is made up of data blocks, each separated by BEGIN and END statements. Three different types of data blocks are allowed:

- **ACDATA** = lists small and large-signal parameters vs. frequency and power (required)
- **NDATA** = lists noise parameters vs. frequency (optional)
- **IMTDATA** = used for a 2-port frequency converter to give the single-tone intermodulation table (optional)
The ACDATA Block

The ACDATA block allows you to specify the small- and large-signal characteristics of the 2-port.

- General format:
  
  BEGIN ACDATA
  
  # AC ( ....... ) ! this is the option line
  % .... ! this is a format line
  ..... small-signal data goes here
  % F ! this is a format line
  ... first large-signal data frequency
  % P1 P2 ..... ! this is a format line
  ... large-signal data at first frequency
  % F ! next large-signal frequency
  ... second large-signal data frequency
  % P1 P2 ....
  ... large-signal data at second frequency
  ...
  ... additional sets of large-signal frequency and data
  ...
  END
• Option line:

```
#AC( unit parm_type parm_format R xx FC m b )
```

desc: freq_unit options: HZ, KHZ, MHZ, or GHZ sets the frequency unit
parm_type options: S  Only S-parameters are allowed
parm_format options: MA, DB, RI, VDB  sets the format for the four network
parameters
MA-for magnitude and angle (deg.)
DB-for 20•log10( MA) and angle (deg.)
RI-for real and imaginary
VDB-for n11 and n22 as VSWR and angle (deg.) and n21 and n12 as DB and
angle (deg.)
R xx where xx = normalization resistance
FC m b for input to output frequency conversion where
Fout = m•Fin + b;
m is for frequency multiplication
b is for frequency translation
• Small-signal format line:

%  F  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. The order of these keywords is arbitrary. The order shown is preferred.

F    = for frequency data
n11x, n11y = the 11 data pair for the 2 port data matrix
n21x, n21y = the 21 data pair
n12x, n12y = the 12 data pair
n22x, n22y = the 22 data pair

\[
\begin{bmatrix}
  n11 & n12 \\
  n21 & n22
\end{bmatrix}
\]

If the parameter type = S, and the parameter format = DB, then n11x = |S11| in dB, and n11y = angle of S11 in degrees.

• Large-signal format line:

%  F

This line precedes the frequency for the following large-signal data.

%  P1  P2  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y

This line precedes the large-signal data. The large-signal data may have up to 101 sets of power data per frequency.

All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred.

P1    = the power (dBm) incident at port 1 with port 2 terminated in R ohms, for the measurement of S11 and S21
P2    = the power (dBm) incident at port 2 with port 1 terminated in R ohms, for the measurement of S22 and S12
In the following data, a value of 1000 (1.e3) for the P2 data indicates that the S22 and S12 data are for small-signal measurement.

\[
\begin{align*}
n_{11x}, n_{11y} & = \text{the 11 data pair for the 2 port data matrix} \\
n_{21x}, n_{21y} & = \text{the 21 data pair} \\
n_{12x}, n_{12y} & = \text{the 12 data pair} \\
n_{22x}, n_{22y} & = \text{the 22 data pair}
\end{align*}
\]

If the parameter type = S, and the parameter format = DB, then \(n_{11x} = |S_{11}|\) in dB, and \(n_{11y} = \text{angle of } S_{11} \text{ in degrees.}\)
ACDATA Block Examples

2-port with 50-ohm S-parameters:

BEGIN ACDATA
\# AC( GHZ S DB R 50 FC 1.0 0.0 )
\% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-db S11-deg S21-db S21-deg S12-db S12-deg S22-db S22-deg
1.0000 -15 45 8 25 -20 -15 -12 10
2.0000 -16 25 9 30 -20 -15 -12 20
3.0000 -17 -10 10 35 -20 -15 -11 30
\% F
1.00
\% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.000 ...
-5.000 5.000 ... large-signal S-parameter data here ....
5.020 15.000 ...
\% F
2.00
\% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.000 ...
-5.000 5.000 ... large-signal S-parameter data here ...
5.020 15.000 ...
END

2-port frequency converter with variable RF freq, variable LO freq, fixed IF freq, fixed LO power, and RF-to-IF 2-port 50-ohm S-parameters:

BEGIN ACDATA
\# AC( GHZ VDB R 50 FC 0.2 ) ! this gives a constant IF of 0.2 GHz
\% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-vswr S11-deg S21-db S21-deg S12-db S12-deg S22-vswr S22-deg
1.0000 1.2 0 -8 0 -20 0 1.3 0
Working with Data Files

2.0000  1.3  0  -9  0  -20  0  1.2  0
3.0000  1.4  0  -10  0  -20  0  1.3  0
% F
1.00
% P1  P2  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
-15.000  -5.00  ...
-5.000  5.00  ...  large-signal S-parameter data here ....
5.020  15.00  ...
% F
2.00
% P1  P2  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
-15.000  -5.00  ...
-5.000  5.00  ...  large-signal S-parameter data here ...
5.020  15.00  ...
END

2-port frequency converter with variable RF freq, fixed LO freq, variable IF freq, fixed LO power, and RF-to-IF 2 port 50-ohm S-parameters:

BEGIN  ACDATA
# AC( GHZ  DB  R 50  FC -1 4 ) ! this gives an IF = 4 - RF
% F  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
! RF-freq S11-dB S11-deg S21-dB S21-deg S12-dB S12-deg S22-dB S22-deg
1.0000  -12  0  -8  0  -20  0  -13  0
2.0000  -13  0  -9  0  -20  0  -12  0
3.0000  -14  0  -10  0  -20  0  -13  0
% F
1000.00
% P1  P2  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
-15.000  -5.00  ...
-5.000  5.00  ...  large-signal S-parameter data here ....
5.020  15.00  ...

END
% F
2000.00
% P1 P2  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
-15.000  -5.00   ...
-5.000   5.00   ... large-signal S-parameter data here ...
  5.020   15.00   ...
END
The NDATA Block

The NDATA block allows the user to specify the small-signal noise characteristics of the 2-port.

- General format:
  
  \[
  \text{BEGIN NDATA} \\
  \# \ AC \ ( \ ....... ) \ ! \ this \ is \ the \ option \ line \\
  \% \ .... \ ! \ this \ is \ the \ format \ line \\
  \ldots \ data \ goes \ here \\
  \text{END}
  \]

- Option line:

  \[
  \# \ AC( \ \text{freq} \_\text{unit} \ \text{parm} \_\text{type} \ \text{parm} \_\text{format} \ \ R \ xx )
  \]

  freq_unit options: HZ, KHZ, MHZ, or GHZ sets the frequency units

  parm_type options: S sets the source noise match type to optimum source reflection coefficient

  parm_format options: MA, DB, RI sets the format for the optimum source match

  - MA for magnitude and angle (deg.)
  - DB for 20 \*\log_{10} (MA) and angle (deg.)
  - RI for real and imaginary

  R xx; where xx = normalization resistance for the source match and \( r_n \)
• Format line:

% F nfmin n11x n11y rn

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred

\[F\] = for frequency data
\[nfmin\] = for minimum noise figure data
\[n11x, n11y\] = for the optimum source impedance for minimum noise figure
\[rn\] = for the equivalent input normalized noise resistance. The system simulator requires this parameter to meet physical requirements. If the user-supplied rn value is less than allowed for this requirement, then the system simulator will force this rn value to the lowest physical limit.
Working with Data Files

**The IMTDATA Block**

The IMTDATA data block allows you to specify for a 2-port frequency converter the single tone output intermodulation levels with respect to the fundamental output tone.

- General format:

  ```
  BEGIN
  reference_signal_power  reference_LO_power
  
  ....... IMT data goes here
  
  END
  ```

This data is given for specific LO and RF input power levels. However, during a system simulator spurious signal analysis, the data is adjusted for the actual converter input signal and LO power levels.
Example

BEGIN IMTDATA
! Intermodulation table for double balanced mixer #1
! Reference Signal Level (dBm)  Reference LO Level (dBm)
#  -10                        7
!  M x LO  (Horizontal)   N x Signal (Vertical)
% 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
!

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>26</td>
<td>35</td>
<td>39</td>
<td>50</td>
<td>41</td>
<td>53</td>
<td>49</td>
<td>51</td>
<td>45</td>
<td>65</td>
<td>55</td>
<td>75</td>
<td>65</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>35</td>
<td>13</td>
<td>40</td>
<td>24</td>
<td>45</td>
<td>28</td>
<td>49</td>
<td>35</td>
<td>45</td>
<td>65</td>
<td>55</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>73</td>
<td>74</td>
<td>70</td>
<td>71</td>
<td>64</td>
<td>69</td>
<td>64</td>
<td>69</td>
<td>65</td>
<td>75</td>
<td>75</td>
<td>85</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>64</td>
<td>69</td>
<td>50</td>
<td>77</td>
<td>47</td>
<td>74</td>
<td>44</td>
<td>74</td>
<td>45</td>
<td>75</td>
<td>55</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>90</td>
<td>86</td>
<td>88</td>
<td>88</td>
<td>85</td>
<td>86</td>
<td>85</td>
<td>90</td>
<td>85</td>
<td>85</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>80</td>
<td>90</td>
<td>71</td>
<td>90</td>
<td>68</td>
<td>90</td>
<td>65</td>
<td>88</td>
<td>65</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>87</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>95</td>
<td>99</td>
<td>95</td>
<td>99</td>
<td>95</td>
<td>99</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>95</td>
<td>90</td>
<td>95</td>
<td>90</td>
<td>99</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

END
Working with Data Files

In the IMT table:

- The vertical row number, \( N \), (0, 1, to 15) indicates the harmonic of the signal used in deriving the spurious output signal.
- The horizontal column number, \( M \), (0, 1, to 15) indicates the harmonic of the local oscillator used in deriving the spurious output signal.
- In row 2, column 4, the data is 13. This means that for an input signal at \(-10 \text{ dBm}\) input, with an LO drive of +7 dBm, an output spurious signal will occur at \(3^\text{LO} + 1^\text{signal}\), with a level that is 13 dB below the fundamental output signal.
- If the input signal differs from the \(-10 \text{ dBm}\) reference power level listed at the top of the table by \(X\) dB, then the number in the table is adjusted by adding \((N-1) \times X\) dB to it. This manner of adjustment is good for input power levels up to 5 dB greater than the reference signal power.
- If the local oscillator signal differs from the +7 dBm reference power level listed at the top of the table by \(X\) dB, then the number in the table is adjusted by adding \(M \times X\) dB to it. This manner of adjustment is good for local oscillator power levels from the reference level minus 10 dB to the reference level plus 3 dB.
- The data values must fall in the range of 0 to 99. Numbers outside this range will cause an error.
- The “#IMT ( )” is optional. Signal and LO reference power levels can be listed without this.
- IMT data can be in square or triangular format.
Example .p2d File

BEGIN ACDATA
! This an optional block of data
! This line and all lines starting with a comment (!) character are ignored
! Do not have a blank line, or comments as the first line in this file
! ACDATA is a required block of data
! The following is optional small-signal data and must precede the
large-signal data
# AC( MHZ  S RI  R 50.0 FC 1 3700 )
! The above line is the OPTION line
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S; S required
! Optional Selection: MA, DB, RI, VDB; Default is MA
! Optional Selection: R xx; where xx = reference resistance; Default R 50.0
! Optional Selection: FC x1 x2 is for frequency conversion: Fout = x1*Fin + x2
! Default is Fout = Fin

%  F   n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
! The above line is the format line showing the order in the following data
lines
! The columns of data can be in any order
250.00 -0.150 0.129 7.888 -22.178 -9.0e-11 1.8e-09 -0.217 -0.502
260.00 -0.145 0.127 -4.885 -23.525 1.4e-10 -5.36e-09 0.036
270.00 -0.140 0.126 -17.420 -6.3e-11 6.0e-11 0.285 0.481
280.00 -0.136 0.125 -22.473 -6.5e-10 -1.9e-10 0.273 0.530
290.00 -0.131 0.123 -22.652 5.045 -6.6e-10 1.5e-10 0.610 0.105
300.00 -0.126 0.121 -17.682 15.564 -1.0e-10 4.0e-10 0.454 -0.415
310.00 -0.121 0.120 -7.838 23.060 1.5e-10 -3.2e-10 -0.085 -0.584
320.00 -0.116 0.118 5.489 24.349 8.3e-11 -7.3e-10 -0.534 -0.124
330.00 -0.112 0.116 17.637 17.846 3.0e-10 1.5e-10 0.414 0.400
340.00 -0.107 0.115 23.851 6.301 3.3e-10 -1.5e-10 0.144 0.586
350.00 -0.102 0.113 23.407 6.383 3.5e-11 1.3e-10 0.578 0.234

! This following is the large-signal data and must be listed
! an unlimited number of sets of frequency data can be specified,
! with an unlimited number of sets of power data at each frequency
% F
250.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.00 -0.150 0.129 7.733 -21.772 -1.1e-10 3.2e-10 -0.216
-0.497
-5.000 5.00 -0.150 0.129 7.335 -20.725 1.1e-10 -3.0e-10 -0.214
-0.484
5.020 15.00 -0.150 0.129 5.519 -15.857 -2.5e-11 -1.0e-09 -0.203
-0.426
% F
260.00

P2D Format 20-61
### Working with Data Files

```plaintext
%  P1  P2  n1x  n1y  n21x  n21y  n12x  n12y  n22x  n22y
-15.000 -5.000  -0.145  0.127  -4.800  -23.077  6.4e-10  5.7e-10  -0.531
-0.034
-5.000  5.000  -0.145  0.127  -4.577  -21.923  1.5e-10  3.0e-10  -0.518
-0.030
5.020  15.000 -0.145  0.127  -3.524  -16.594  8.4e-11  9.9e-10  -0.457
-0.012

%  F
350.00
%  P1  P2  n1x  n1y  n21x  n21y  n12x  n12y  n22x  n22y
-15.000 -5.000  -0.102  0.113  22.951  -6.264  2.3e-11  8.3e-11  0.572
  0.233
-5.000  5.000  -0.102  0.113  21.764  -5.952  -5.0e-10  6.5e-10  0.558
  0.230
5.020  15.000 -0.102  0.113  16.176  -4.466  -4.1e-10  3.7e-10  0.493
  0.217
END
```

```plaintext
BEGIN  NDATA
! This is an optional block of data
# AC( MHZ  RI  S  R  50.0 )
! Optional Selection:  HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection:  S, Z, Y, H, or G; Default is S
! Optional Selection:  MA, DB, or RI;  Default is MA
! Optional Selection: R xx; where xx = reference resistance; Default R 50.0
! An unlimited number of lines of frequency data can be specified
%  F   nmin  n1x  n1y  rn
! The above line is the format line showing the order in the following data lines
! The columns of data can be in any order
250.00  0.874  0.131  150.374  0.048
260.00  0.880  0.127  149.640  0.049
270.00  0.886  0.124  148.895  0.050
280.00  0.892  0.121  148.136  0.050
290.00  0.897  0.118  147.363  0.051
300.00  0.903  0.115  146.574  0.051
310.00  0.909  0.111  145.769  0.052
320.00  0.915  0.108  144.944  0.053
330.00  0.921  0.105  144.098  0.053
340.00  0.927  0.101  143.230  0.054
350.00  0.933  0.098  142.335  0.055
END
```

```plaintext
BEGIN  IMTDATA
```

---

20-62  P2D Format
<table>
<thead>
<tr>
<th>Signal Level (dBm)</th>
<th>LO Level (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td># -10</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M x LO (Horizontal)</th>
<th>N x Signal (Vertical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
<td></td>
</tr>
</tbody>
</table>

```
99 26 35 39 50 41 53 49 51 42 62 51 60 47 77 50
24 0 35 13 40 24 45 28 49 33 53 42 60 47 63
73 73 74 70 71 64 69 64 69 62 74 62 72 70
67 64 69 50 77 47 74 44 74 47 75 44 70
86 90 86 88 88 85 86 85 90 85 85 85
90 80 90 71 90 68 90 65 88 65 85
90 90 90 90 90 90 90 90 90 90 90 90
99 95 99 95 99 95 99 95 99 95 99 95
90 95 90 90 90 99 90
99 99 99 99 99 99 99
90 99 90
99 99
99
```

END
Example .p2d File with Annotation

BEGIN ACDATA

# AC( GHZ RI )                              -- option line
% F  N11X  N11Y  N21X  N12X  N12Y  N22X  N22Y
  3.0 -8.04 -133. 5.85  49. -19.54 -108. -8.78 -128.     -- small-signal format line
  -- small-signal data

% F
  3.0

% P1  P2  N11X  N11Y  N21X  N21Y  N12X  N12Y  N22X  N22Y

-5.0  -5.0  8.  -132. 6.  50.  -20.  -105.  -8.8  -129.     -- large-signal format line

-- header format line
-- large-signal data

  4.0  4.0  8.  -132. 5.7  51.  -20.5 -105.  -8.8  -129.
  6.0  6.0  8.  -132. 5.7  51.  -20.5 -105.  -8.8  -129.
  8.0  8.0  8.  -132. 5.5  51.  -20.5 -105.  -8.8  -129.
 10.0 10.0  7.9 -132. 5.1  51.  -20.5 -105.  -8.8  -129.
 12.0 12.0  7.9 -131. 4.2  50.  -20.5 -105.  -8.7  -129.
 14.0 14.0  7.8 -131. 3.  50.  -20.  -105.  -8.7  -129.
 16.0 16.0  7.7 -131. 1.6  50.  -17.5 -107.  -8.7  -129.

END

20-64  P2D Format
Format:

A. BEGIN ACDATA
B. #AC (GHZ S MA R 50.0 FC x1 x2)
C. %F N11X N11Y N21X N21Y N12X N12Y N22X N22Y

<data line>
...
<data line>
D. %F
<data line>
E. %P1 P2

<data line>
...
<data line>
END

F. BEGIN NDATA
G. #AC (GHZ S MA R 50.0)
H. %F NFMIN N11X N11Y RN

<data line>
...
<data line>
I. END
Notes:

A. BEGIN alternating current data statement
B. Option line
C. Format line specifying small-signal frequency data
D. Format line specifying set of frequency data
E. Format line specifying set of power data
F. BEGIN noise data statement
G. Option line
H. Format line specifying up to n lines of noise data
I. END statement

where

• BEGIN statements

BEGIN ACDATA Begin alternating current data
BEGIN NDATA Begin noise data
• Option lines

ACDATA: #AC ([HZ/ KHZ/ MHZ/ GHZ] [S/ Z/ Y/ H/ G] [MA/ DB/ RI] [R n] [FC x1 x2])
NDATA: #AC ([HZ/ KHZ/ MHZ/ GHZ] [MA/ DB/ RI] [R n])

where

# = Delimiter; tells the program you are specifying these parameters
AC = Alternating current
HZ/ KHZ/ MHZ/ GHZ = Hertz/kiloHertz/megaHertz/gigaHertz
S/ Z/ Y/ H/ G = Scattering/impedance/admittance/hybrid-h/hybrid-g parameters
MA/ DB/ RI = Magnitude-angle/decibel/real-imaginary
R n = Reference resistance value, in ohms
FC = Frequency conversion keyword
x1, x2 = Numbers defining frequency-in/frequency-out relationship. FOUT = x1 FIN + x2
### IMT Format

The intermodulation product table (.imt) file is a user-defined table of mixer intermodulation (IM) products between the LO and signal that relates your mixer IM output level to the signal input level.

The .imt file is used to define the MIX2 component in system analysis and is also used in the mixer-spurious analysis. (See also the System Tools booklet for further details on mixer-spurious analysis.)

The `zbl1.imt` and `zbl2.imt` files are immediately available in the simulator and do not require a path to be used.

### Example

The following table is for small-signal input conditions for which the IM level is fixed by the type of mixer and LO drive. This table is an example of a characteristic mixer at a specific LO drive level. This table does not give IM results for multi-tone signal input.

In the following example, the IM product for 5*LO + 1*SIGNAL is 24 dB below the fundamental output. The IM product for 2*LO + 3*SIGNAL is 69 dB below the fundamental output.

```plaintext
BEGIN IMT_DATA
!Intermodulation table for double balanced mixer #1
!Reference signal Level (dBm) Reference LO Level (dBm)
#IMT (-107)
!M x LO (Horizontal) N x Signal (Vertical)
% 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
99 26 35 39 50 41 53 49 51 45 65 55 65 55 99 99 !0
24 0 35 13 40 24 45 28 49 35 55 45 85 99 99 99 !2
73 73 74 70 71 64 69 64 69 65 75 75 99 99 99 99 !3
67 64 69 50 77 47 74 44 74 45 75 55 99 99 99 99 !4
86 90 86 88 88 85 86 85 90 85 85 99 99 99 99 99
90 80 90 71 90 68 90 65 88 65 99 99 99 99 99 99
90 90 90 90 90 90 90 90 90 90 99 99 99 99 99 99
```

---

20-68  IMT Format
TIM, BINTIM Formats

The .tim and .bintim files are signal data files in MDIF format. They contain time-domain waveform data for defining the signals associated with certain sources.

The general .tim file format is:

```
BEGIN TIMEDATA
# T (SEC V R xx)
% t v
<data line>
...
<data line>
END
```

For binary (.bintim) files, the BEGIN line is preceded by a line indicating the number of data points, n:

```
NUMBER OF DATA n
```

The <data line> in a .bintim file is just a binary dump of all the waveform (t, v) data.

Guidelines for .tim

1. An exclamation point (!) at the beginning of a line makes it a comment line. Characters following the ! are ignored by the program.
2. The TIMEDATA data block is required.
TIMEDATA Block

- The BEGIN statement:

```
BEGIN TIMEDATA ! Begin time-domain waveform data
```

- Option line:

```
#T ([SEC/ MSEC/ USEC/ NSEC/ PSEC] V/ MV/ DATA R xx )
```

where

- `#` = Delimiter that tells the program you are specifying these parameters
- `T` = Time
- `SEC/MSEC/USEC/NSEC/PSEC` = Your choice of Seconds, Milliseconds, Microseconds, Nanoseconds, or Picoseconds
- `V/MV/DATA` = Your choice of Volts or Millivolts of data noninterpolation
- `R` = Reference resistance, default is 50.0
- `xx` = User-specified value for reference resistance

- Format line:

```
% t v
```

where

- `%` = Delimiter that tells the program you are specifying these parameters
- `t` = time
- `v` = voltage

By design of the program, the syntax `t` and `v` in the Format line are completely arbitrary. These values can be whatever you prefer. For example, an option line such as:

```
% time voltage
```

can be used. If used this way, these names would correspond to variables appearing in a dataset file (.ds file) if the .tim file were imported using the instrument server.

- The TIMEDATA data requirements are as follows:
  - A value for `t=0` is not required.
• The signal is assumed to be time periodic with the time period equal to maximum time minus minimum time.

• The DATA option line value specifies that the voltage values are to be interpreted as discrete voltage levels with no interpolation between consecutive voltage points, with each voltage level held constant until the next voltage point.

**Example .tim Files**

**Example 1**

```
BEGIN  TIMEDATA
# T ( USEC  V  R 50 )
% t  v
  0.0 -1.0
  2.0  1.0
  4.0  2.0
  8.0  3.0
 10.0  3.0
 14.0  0.0
 18.0 -1.0
 24.0 -2.0
 28.0  0.0
 32.0 -1.0
END
```

**Example 2**

```
BEGIN  TIMEDATA
# T ( USEC DATA R 50 )
% t  v
  0.0 -1.0
  2.0  1.0
  4.0  2.0
  8.0  3.0
 10.0  3.0
 14.0  0.0
 18.0 -1.0
 24.0 -2.0
 28.0  0.0
 32.0 -1.0
END
```
This example file results in a time periodic voltage versus time with time period 32 \( \mu \text{sec} \). The first example is interpreted as a piece-wise linear voltage description, the second example is interpreted as a voltage state description.

Figure 20-1. Time Periodic Voltage vs. Time with Time Period 32 \( \mu \text{sec} \).
DSCR Format

The discrete data file consists of an array of data arranged in rows and columns. The values available for each parameter are arranged in columns. Following the BEGIN DSCR DATA line is the % format line which specifies the names of dependent variables. The first column is always treated as a string; other columns are real, integer or string, depending on the first row of data.

The first column, under the heading Index in the example below, contains entries used to identify each row in the file. These entries can be either an integer or an alphanumeric identifier, and can be thought of as a list of specification numbers (or part numbers). For example, the data file data/stdvalues15.dscr is arranged as follows:

REM     stdvalues15.dscr
BEGIN DSCRDATA
% INDEX  A12  A13
  1  1000   1000
  2  1000   1200
  3  1000   2200
  4  1200   1000
  5  1200   1200
  6  1200   2200
END DSCRDATA

Selecting a Row

A row of data can be selected by specifying its row index (starting from 0). In this example, the file lists two columns of values labeled A12 and A13. By specifying 2 as the row number, the values 1000 and 2200 are selected for A12 and A13, respectively.
Using the File with a DAC

To use the data within the file, you must link the file to the component of interest. You reference a discrete data file in this way by using a DAC:

1. Place a DataAccess Component data item in your design. the DAC is located in the Data Items palette.
2. Double-click the DAC to edit it. In the File field, specify the name of the discrete data file. Set Type to Discrete and InterpMode to Index Lookup.
3. You will want to use the INDEX variable from the discrete data file to access rows of data. INDEX, which is an implicit independent variable, must be specified by its integer cardinality. To do this, select iVar and set iVar to 1 by typing @1. @ is required to suppress the double quotes (recall that the first column in the discrete data file is treated as a string). 1 identifies it as the innermost independent variable. To set the value (in this case, the row index), select iVal and enter INDEX-1. Recall that rows are indexed beginning with zero.

The other variable name, value pairs {iVarK, iValK} K = ...10, are only needed for multi-dimensional data, i.e., if the file has outer independent variables such as VAR SAMPLE VALUE = 3 for each discrete block.
4. Place the component whose parameter values should come from the data file. Double-click the component.
5. Under Parameter Entry Mode, select File Based.
6. Under Data Access Component Instance, enter the ID of the desired DAC data item.
7. Under Dependent Parameter Name, enter the name of a dependent variable in the discrete data file (In the sample above, the dependent variable names are A12 and A13.).

Example

For an example of using a discrete data file, refer to amp1.dsn in the Examples directory, under Tutorials/dataAccess_prj.
Working with Data Files

Generalized MDIF

The generalized MDIF provides a generic MDIF format for unifying the various specific MDIF formats, and overcome some limitations of other formats. The generic format enables diverse applications to use a common data I/O interface, so long as the intent is to access/save multidimensional (multiple independent vs dependent variables) data.

The general format is as follows:

```
VAR var1Name(var1Type) = var1Value
VAR var2Name(var2Type) = var2Value
...
VAR varNName(varNType) = varNValue
BEGIN blockName
# attrib1Name(attrib1Type) = attrib1Value
# attrib2Name(attrib2Type) = attrib2Value..
...
# attribMName(attribMType) = attribMValue
% bVar1Name(bVar1Type) bVar2Name(bVar2Type) ....
% bVar1Name(bVar1Type) ...
% ...
% bVarQName(bVarQType) ... bVarPName(bVarPType)
bVar1Value bVar2Value ...
bVarQValue ...
% bVarPValue
...
END
```

where var*Type can be the token:

- 0 or int
- 1 or real
- 2 or string

---

20-76 Generalized MDIF
Types attrib*Type, bVar*Type can be one of the above as well as:

3 or complex
4 or boolean
5 or binary
6 or octal
7 or hexadecimal
8 or byte16

The variable names above constitute a name-space uniquely identified by the string blockName which is either:

- alphanumeric: all bVar*Name block variables are dependent, except bVar1Name, which is usually the most rapidly changing (innermost) independent variable.

or

- DSCR(blockName): all bVar*Name block variables are dependent, and there is an indexing implicit independent variable.

A string type variable’s value must be surrounded by “”.

The block data (bVar*Value) lines must follow the pattern (order, number and number of lines) of the format (%) lines. A variable’s value cannot be split across lines. Although there is no line length limit specified, MDIF file readers may choose to truncate at some finite length. This would likely result in a file read error, or, if the file had been carefully crafted, truncated names and/or string-type values.

Real numbers may have case-insensitive scale factor suffixes as follows:

f = 1e-15, p = 1e-12, n = 1e-9, u = 1e-6, mil = 2.54e-5, m = 1e-3, k = 1e3, g = 1e9, t = 1e12

E.g.: 15mA = 15e-3, 30KHz = 30e3

Note that there should be no space between the number and the suffix, and extra characters are ignored. Unrecognized suffixes result in 1.0. The above is not totally consistent with rest of ADS.
Example 1

This has 3 indepVars: v1, v2, v3(innermost) and
4 depVars: dv1(integer), dv2(real), dv3(string) and
dv4(hexadecimal), but is read in as a string
! There are 2 data nodes, with an attribute “att3”

VAR v1(0) = 1
VAR v2(1) = 2.2
BEGIN blk1
   # att3(2) = “Hello, World”
   % v3(1) dv1(0) dv2(1) dv3(2) dv4(hexadecimal)
   7.789.9999”line 1” 0xabc
   8.891.11”line 2” 0x123
END

VAR v1(0) = 2
VAR v2(1) = 3.2
BEGIN blk1
   # att3(2) = “Hello, Universe”
   % v3(1) dv1(0) dv2(1) dv3(2) dv4(hexadecimal)
   8.79uF10.9999mA”line 1” 0xff
   9.810uF11.11mA”line 2” 0xdef
END

Example 2

Created Tue Mar 9 13:39:19 1999
Data Acquired Tue Mar 9 13:38:34 1999

BEGIN NDATA_noise
   % freq(real) Sopt(complex) NFmin(real) Rn(real) PortZ[1](real)
   1e+09 0.098481 0.017365 1 5 50
   2e+09 0.18794 0.068404 2 10 50
   3e+09 0.25981 0.15 3 15 50
   4e+09 0.30642 0.25712 4 20 50
   5e+09 0.32139 0.38302 5 25 50
   6e+09 0.3 0.51962 6 30 50
   7e+09 0.23941 0.65778 7 35 50
   8e+09 0.13892 0.78785 8 40 50
   9.543e+09 -0.014122 0.911 9.5445 46.166 50
END

20-78    Generalized MDIF
PDF Format

This format is for user-specified Probability Density Functions (PDF). PDF is used for arbitrary distributions that are not correlated. The two methods available accommodate:

- Situations where the spread of statistical data is proportional to the nominal value (as in a percent tolerance).
- Situations where the spread is independent of the nominal value (an absolute tolerance).

Probability density functions are represented as vertices of piecewise linear segments. These value/ordinate pairs are stored in a textual data file in a prescribed format. The nominal value is also stored.

User-specified probability density files have an extension .pdf and use an MDIF file format. Only a single distribution definition is allowed in each .pdf file.

Guidelines for .pdf

- In addition to the nominal value, there must be a minimum of three value/ordinate data pairs.
- For tolerance representation, all value data and the nominal value must have the same algebraic sign.
- The ordinate data associated with the most negative and most positive value data must be zero.
- All ordinate data must be non-negative.
- At least one non-end ordinate data must be non-zero.
A sample file for tolerance PDF data is shown next.

REM   An example probability density function description file
REM   Data listed represent a triangular density
REM with a +/- 10% variation
REM
BEGIN TOLERANCE
  %NOMINAL
    100
  %VALUE  ORDINATE
    90   0.0
    100  1.0
    110  0.0
END TOLERANCE

Note   The total area under your PDF does not have to equal one as in the strict
definition of a PDF—the simulator will automatically scale your PDF to meet this
condition.

Interpretation of PDF data

Interpretation of user-supplied PDF data is piece-wise linear with respect to
value/ordinate pairs. The data preparation previously described enables the program
to supply a properly qualified variate.

To realize a statistical variable which obeys the user-supplied PDF, a uniform variate
on the interval 0 to 1 is used as an input to a function which is the inverse of the
cumulative distribution function (CDF). The CDF is formed by integrating the PDF
from its most negative value to its most positive value, with the following conditions:

- \( F_X(-\infty) \), the CDF at minus infinity = 0
- \( F_X(\infty) \), the CDF at infinity = 1

Applying this uniform \([0,1]\) variate to the inverse of the CDF results in a statistical
variable having the user-specified probability density function.

PDFs may be used with:

- Yield analysis, with or without post-production tuning
- Yield optimization (design centering)
ASCSIG and SIG Formats

These files contain time-domain waveform data and are signal data files in SPW format. Advanced Design System can interface with the Cadence Alta Group SPW format through the use of data files in SPW format. Both ASCII, .ascsig and binary, .sig file formats are supported.

The SPW version 3.0 data file format is fully supported for real double data and partially supported for complex double data.

Guidelines for .ascsig

• The SPW version 3.0 data file format must be used.
• Comments can only be included on the one line following the $USER_COMMENT statement.

Example .ascsig Files

There are two example, one uses real values and the second uses complex numbers.

File 1: Real double-data format

$SIGNAL_FILE 9
$USER_COMMENT

$COMMON_INFO
SPW Version=3.0

Sampling Frequency=1
Starting Time=0
$DATA_INFO
Number of points = 6
Signal Type = Double
$DATA
1.00000000000000000000
1.00000000000000000000
Working with Data Files

- 1.000000000000000000000
- 1.000000000000000000000
1.000000000000000000000
1.000000000000000000000

File 2: Complex double-data format

$SIGNAL_FILE 9
$USER_COMMENT

$COMMON_INFO
SPW Version=3.0

Sampling Frequency=1
Starting Time=0
$DATA_INFO
Number of points=10
Signal Type=Double
Complex Format =Real_Imag
$DATA
1.000000000000000000000+j1.000000000000000000000
1.000000000000000000000+j1.000000000000000000000
-1.000000000000000000000+j1.000000000000000000000
-1.000000000000000000000+j1.000000000000000000000
-1.000000000000000000000+j1.000000000000000000000
1.000000000000000000000+j1.000000000000000000000
1.000000000000000000000+j1.000000000000000000000
1.000000000000000000000+j1.000000000000000000000
-1.000000000000000000000+j1.000000000000000000000

20-82 ASCSIG and SIG Formats
CITIfile Data Format

This section describes the CITIfile format definitions of key terms, and file examples. It also includes:

• Keyword reference
• File guidelines
• Instructions for converting between disk formats
• Device-specific definitions
• File name requirements

Overview

CITIfile is a standardized data format that is used for exchanging data between different computers and instruments. CITIfile stands for Common Instrumentation Transfer and Interchange file format.

This standard is a group effort between instrument and computer-aided design program designers. As much as possible, CITIfile meets current needs for data transfer, and it is designed to be expandable so it can meet future needs.

CITIfile defines how the data inside an ASCII package is formatted. Since it is not tied to any particular disk or transfer format, it can be used with any operating system, such as DOS or UNIX, with any disk format, such as DOS or HFS, or with any transfer mechanism, such as by disk, LAN, or GPIB.

By careful implementation of the standard, instruments and software packages using CITIfile are able to load and work with data created on another instrument or computer. It is possible, for example, for a network analyzer to directly load and display data measured on a scalar analyzer, or for a software package running on a computer to read data measured on the network analyzer.
Working with Data Files

Data Formats
There are two main types of data formats: binary and ASCII. CITIfile uses the ASCII text format. Although this format requires more space than binary format, ASCII data is a transportable, standard type of format which is supported by all operating systems. In addition, the ASCII format is accepted by most text editors. This allows files to be created, examined, and edited easily, making CITIfile easier to test and debug.

File and Operating System Formats
CITIfile is a data storage convention designed to be independent of the operating system, and therefore may be implemented by any file system. However, transfer between file systems may sometimes be necessary. You can use any software that has the ability to transfer ASCII files between systems to transfer CITIfile data. Refer to “Converting Between Disk Formats” on page 20-95 for more information.

The descriptions and examples shown here demonstrate how CITIfile may be used to store and transfer both measurement information and data. The use of a single, common format allows data to be easily moved between instruments and computers.
CITIfile Definitions

This section defines: package, header, data array, and keyword.

Package

A typical CITIfile package is divided into two parts:

- The header is made up of keywords and setup information.
- The data usually consists of one or more arrays of data.

The following example shows the basic structure of a CITIfile package:

```
CITIFILE A.01.00
NAME MEMORY
VAR FREQ MAG 3
DATA S RI
BEGIN
  -3.54545E-2,-1.38601E-3
  0.23491E-3,-1.39883E-3
  2.00382E-3,-1.40022E-3
END
```

When stored in a file there may be more than one CITIfile package. With the Agilent 8510 network analyzer, for example, storing a memory all will save all eight of the memories held in the instrument. This results in a single file that contains eight CITIfile packages.
Working with Data Files

**Header**

The header section contains information about the data that will follow. It may also include information about the setup of the instrument that measured the data. The CITIfile header shown in the first example has the minimum of information necessary; no instrument setup information was included.

**Data Array**

An array is numeric data that is arranged with one data element per line. A CITIfile package may contain more than one array of data. Arrays of data start after the `BEGIN` keyword, and the `END` keyword follows the last data element in an array.

A CITIfile package does not necessarily need to include data arrays. For instance, CITIfile could be used to store the current state of an instrument. In that case the keywords `VAR`, `BEGIN`, and `END` would not be required.

When accessing arrays via the DAC (DataAccessComponent), the simulator requires array elements to be listed completely and in order.

Example: S[1,1], S[1,2], S[2,1], S[2,2]

**Keywords**

Keywords are always the first word on a new line. They are always one continuous word without embedded spaces. A listing of all the keywords used in version A.01.00 of CITIfile is shown in “CITIfile Keyword Reference” on page 20-91.
CITIfile Examples

The following are examples of CITIfile packages.

Display Memory File

This example shows an Agilent 8510 display memory file. The file contains no frequency information. Some instruments do not keep frequency information for display memory data, so this information is not included in the CITIfile package. Note that instrument-specific information (#NA = network analyzer information) is also stored in this file.

    CITIFILE A.01.00
    #NA VERSION HP8510B.05.00
    NAME MEMORY
    #NA REGISTER 1
    VAR FREQ MAG 5
    DATA S RI
    BEGIN
     -1.31189E-3,-1.47980E-3
     -3.67867E-3,-0.67782E-3
     -3.43990E-3,0.58746E-3
     -2.70664E-4,-9.76175E-4
     0.65892E-4,-9.61571E-4
    END
Working with Data Files

**Agilent 8510 Data File**

This example shows an 8510 data file, a package created from the data register of an Agilent 8510 network analyzer. In this case, 10 points of real and imaginary data were stored, and frequency information was recorded in a segment list table.

```
CITIFILE A.01.00
#NA VERSION HP8510B.05.00
NAME DATA
#NA REGISTER 1
VAR FREQ MAG 10
DATA S[1,1] RI
SEG_LIST_BEGIN
SEG 1000000000 4000000000 10
SEG_LIST_END
BEGIN
  0.86303E-1,-8.98651E-1
  8.97491E-1,3.06915E-1
  -4.96887E-1,7.87323E-1
  -5.65338E-1,-7.05291E-1
  8.94287E-1,-4.25537E-1
  1.77551E-1,8.96606E-1
  -9.35028E-1,-1.10504E-1
  3.69079E-1,-9.13787E-1
  7.80120E-1,5.37841E-1
  -7.78350E-1,5.72082E-1
END
```
Agilent 8510 3-Term Frequency List Cal Set File

This example shows an 8510 3-term frequency list cal set file. It shows how CITIfile may be used to store instrument setup information. In the case of an 8510 cal set, a limited instrument state is needed to return the instrument to the same state that it was in when the calibration was done.

Three arrays of error correction data are defined by using three `DATA` statements. Some instruments require these arrays be in the proper order, from $E[1]$ to $E[3]$. In general, CITIfile implementations should strive to handle data arrays that are arranged in any order.

```plaintext
CITIFILE A.01.00
#NA VERSION HP8510B.05.00
NAME CAL_SET
#NA REGISTER 1
VAR FREQ MAG 4
DATA E[1] RI
DATA E[2] RI
DATA E[3] RI
#NA SWEEP_TIME 9.999987E-2
#NA POWER1 1.0E1
#NA POWER2 1.0E1
#NA PARAMS 2
#NA CAL_TYPE 3
#NA POWER_SLOPE 0.0E0
#NA SLOPE_MODE 0
#NA TRIM_SWEEP 0
#NA SWEEP_MODE 4
#NA LOWPASS_FLAG -1
#NA FREQ_INFO 1
#NA SPAN 1000000000 3000000000 4
#NA DUPLICATES 0
#NA ARB_SEG 1000000000 1000000000 1
```
When an instrument's frequency list mode is used, as it was in this example, a list of frequencies is stored in the file after the VAR_LIST_BEGIN statement. The unsorted frequency list segments used by this instrument to create the VAR_LIST_BEGIN data are defined in the #NA ARB_SEG statements.
## CITIfile Keyword Reference

Table 20-1 lists keywords, definitions, and examples.

### Table 20-1. CITIfile Keywords and Definitions

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Example and Explanation</th>
</tr>
</thead>
</table>
| CITIFILE  | Example: CITIFILE A.01.00  
Identifies the file as a CITIfile and indicates the revision level of the file. The CITIFILE keyword and revision code must precede any other keywords.  
The CITIFILE keyword at the beginning of the package assures the device reading the file that the data that follows is in the CITIfile format. The revision number allows for future extensions of the CITIfile standard.  
The revision code shown here following the CITIFILE keyword indicates that the machine writing this file is using the A.01.00 version of CITIfile as defined here. Any future extensions of CITIfile will increment the revision code. |
| NAME      | Example: NAME CAL_SET  
Sets the current CITIfile package name. The package name should be a single word with no embedded spaces. Some standard package names:  
RAW_DATA: Uncorrected data.  
DATA: Data that has been error corrected. When only a single data array exists, it should be named DATA.  
CAL_SET: Coefficients used for error correction.  
CAL_KIT: Description of the standards used.  
DELAY_TABLE: Delay coefficients for calibration. |
### Table 20-1. CITIfile Keywords and Definitions (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Example and Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VAR</strong></td>
<td>Example: <code>VAR FREQ MAG 201</code></td>
</tr>
<tr>
<td></td>
<td>Defines the name of the independent variable (<code>FREQ</code>); the format of values in a <code>VAR_LIST_BEGIN</code> table (<code>MAG</code>) if used; and the number of data points (201).</td>
</tr>
<tr>
<td><strong>CONSTANT</strong></td>
<td>Example: <code>CONSTANT name value</code></td>
</tr>
<tr>
<td></td>
<td>Lets you record values that do not change when the independent variable changes.</td>
</tr>
<tr>
<td>#</td>
<td>Example: <code>#NA POWER1 1.0E1</code></td>
</tr>
<tr>
<td></td>
<td>Lets you define variables specific to a particular type of device. The pound sign (#) tells the device reading the file that the following variable is for a particular device.</td>
</tr>
<tr>
<td></td>
<td>The device identifier shown here (NA) indicates that the information is for a network analyzer. This convention lets you define new devices without fear of conflict with keywords for previously defined devices. The device identifier can be any number of characters.</td>
</tr>
<tr>
<td><strong>SEG_LIST_BEGIN</strong></td>
<td>Indicates that a list of segments for the independent variable follows. Segment Format: <code>segment type start stop number of points</code></td>
</tr>
<tr>
<td></td>
<td>The current implementation supports only a signal segment. If you use more than one segment, use the <code>VAR_LIST_BEGIN</code> construct. CITIfile revision A.01.00 supports only the <code>SEG</code> (linear segment) segment type.</td>
</tr>
<tr>
<td><strong>SEG_LIST_END</strong></td>
<td>Sets the end of a list of independent variable segments.</td>
</tr>
<tr>
<td><strong>VAR_LIST_BEGIN</strong></td>
<td>Indicates that a list of the values for the independent variable (declared in the <code>VAR</code> statement) follows. Only the <code>MAG</code> format is supported in revision A.01.00.</td>
</tr>
</tbody>
</table>

---

20-92   CITIfile Data Format
Table 20-1. CITIfile Keywords and Definitions (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Example and Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR_LIST_END</td>
<td>Sets the end of a list of values for the independent variable.</td>
</tr>
<tr>
<td>DATA</td>
<td>Example: DATA $[1,1]$ RI</td>
</tr>
</tbody>
</table>

Defines the name of an array of data that will be read later in the current CITIfile package and the format that the data will be in. Multiple arrays of data are supported by using standard array indexing as shown above. CITIfile revision A.01.00 supports only the RI (real and imaginary) format, and a maximum of two array indexes.

Commonly used array names include:

- **S** S parameter
- **E** Error Term
- **Voltage** Voltage
- **VOLTAGE_RATIO** a ratio of two voltages (A/R)
CITIfile Guidelines

The following general guidelines aid in making CITIfiles universally transportable:

**Line Length.** The length of a line within a CITIfile package should not exceed 80 characters. This allows instruments which may have limited RAM to define a reasonable input buffer length.

**Keywords.** Keywords are always at the beginning of a new line. The end of a line is as defined by the file system or transfer mechanism being used.

**Unrecognized Keywords.** When reading a CITIfile, unrecognized keywords should be ignored. There are two reasons for this:

- Ignoring unknown keywords allows new keywords to be added, without affecting an older program or instrument that might not use the new keywords. The older instrument or program can still use the rest of the data in the CITIfile as it did before. Ignoring unknown keywords allows “backwards compatibility” to be maintained.
- Keywords intended for other instruments or devices can be added to the same file without affecting the reading of the data.

**Adding New Devices.** Individual users are allowed to create their own device keywords through the # (user-defined device) mechanism. (Refer to “CITIfile Keywords and Definitions” on page 20-91 for more information.) Individual users should not add keywords to CITIfiles without using the # notation, as this could make their files incompatible with current or future CITIfile implementations.

**File Names.** Some instruments or programs identify a particular type of file by characters that are added before or after the file name. Creating a file with a particular prefix or ending is not a problem. However in general an instrument or program should not require any such characters when reading a file. This allows any file, no matter what the filename, to be read into the instrument or computer. Requiring special filename prefixes and endings makes the exchange of data between different instruments and computers much more difficult.
Converting Between Disk Formats

Most current Agilent Technologies instruments use disks formatted in the Logical Interchange Format (LIF). Some instruments also use DOS-formatted disks. CITI files created on one file system (LIF, DOS, HFS, etc.) may be transferred to other file systems. This is useful for designers using test equipment in addition to ADS to read/write CITI files.

HFS

Several LIF and DOS utilities are available for HP-UX workstations. The HP-UX utilities lifcp and doscp can transfer CITI files to and from LIF and DOS disks. Using lifcp and doscp are similar; using lifcp is described below. Several other LIF and DOS utilities are also available. Consult the manuals for these utilities for more detailed information. Listing the contents of a LIF disk when using HP-UX would be similar to the following example:

```
lifls  /dev/rdsk/ls0.0
```

The device name used will depend on how your system was configured. Copying a CITI file named `DD_FILED1` from a LIF disk to HFS would be similar to the following example:

```
lifcp  /dev/rdsk/ls0.0:DD_FILED1  DD_FILED1
```

To copy a standard HFS ASCII file to a LIF disk:

```
lifcp  DD_FILED1  /dev/rdsk/ls0.0:DD_FILED1
```
When used on an HFS disk, The HP-UX program RMB/UX (Rocky Mountain BASIC for HP-UX) has the ability to write a CITI file in either as a standard HFS ASCII file, or as a LIF volume file. The LIF volume file is the default. This type of file is not directly readable when using the HP-UX operating system, and the copy commands listed above will not work correctly.

BASIC program writers are encouraged to detect when writing to an HFS disk, and to use the standard HFS format. The program examples CITIWRITE and CITIDOALL show how this can be done. However CITI files stored in the LIF volume format can still be transferred to LIF disks, or converted to standard HFS files. To copy a LIF volume file named DD_FILED stored on an HFS disk and move it to a LIF disk:

```
lifcp  DD_FILED1:WS_FILE  /dev/rdsk/1s0.0:DD_FILED1
```

To copy the LIF volume file DD_FILED1 to a standard HFS file named NEWFILE:

```
lifcp  DD_FILED1:WS_FILE  NEWFILE
```

DOS

Utilities are available for DOS machines that enable them to transfer files to and from a LIF formatted disk. Many of these programs are menu-driven, and are available from the following companies: HP, Oswego, Meadow Soft Works, and Innovative Software Systems.
CITIfile Device-Specific Definitions

CITIfile is a generic definition of a data storage format for any type of computer or instrument. However each type of device may need to define certain conventions for itself. This section describes the device-specific keywords and conventions for current implementations.

Network Analyzer (#NA) Definitions

Data Grouping  Data arrays of the same type, obtained during a single measurement operation, are stored in a single CITIfile package. For example, all error correction arrays are stored in the same CITIfile package, and all parameters acquired during an s-parameter measurement operation are stored in the same CITIfile package.

A CITIfile package is as described in the main CITIfile documentation: the CITIFILE keyword, followed by a header section, usually followed by one or more arrays of data.

Network Analyzer Keywords  The definition of CITIfile allows for statements that are specific to a certain type of device. Table A-2 lists the currently defined commands for the #NA (network analyzer) keyword.

Table 20-2. Network Analyzer Keyword Commands

<table>
<thead>
<tr>
<th>Statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>#NA ARB_SEG x y p</td>
<td>A list segment, as entered by the user.</td>
</tr>
<tr>
<td></td>
<td>xx = start value</td>
</tr>
<tr>
<td></td>
<td>y = stop value</td>
</tr>
<tr>
<td></td>
<td>p = number of points.</td>
</tr>
<tr>
<td>#NA REGISTER nn</td>
<td>Register in instrument that the current data package was stored in.</td>
</tr>
<tr>
<td></td>
<td>nn = number of register.</td>
</tr>
<tr>
<td>#NA SWEEP_TIME tt</td>
<td>The sweep time of the analyzer.</td>
</tr>
<tr>
<td></td>
<td>tt = time in seconds.</td>
</tr>
<tr>
<td>#NA POWER1 pp</td>
<td>Power level of signal source #1.</td>
</tr>
<tr>
<td></td>
<td>pp = power in dBm.</td>
</tr>
<tr>
<td>#NA POWER2 pp</td>
<td>Power level of signal source #2.</td>
</tr>
<tr>
<td></td>
<td>pp = power in dBm.</td>
</tr>
</tbody>
</table>
### Table 20-2. Network Analyzer Keyword Commands (continued)

<table>
<thead>
<tr>
<th>Statement</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| #NA PARAMS aa                 | Bitmap of valid parameters for a calibration. Bit positions 1-8 represent the following:  
  Bit #1 = $S_{11}$  
  Bit #2 = $S_{21}$  
  Bit #3 = $S_{12}$  
  Bit #4 = $S_{22}$  
  Bit #5 = user1  
  Bit #6 = user2  
  Bit #7 = user3  
  Bit #8 = user4  
  A bit equal to one means that the calibration is valid for that parameter; a zero means that the calibration is not valid for that parameter. Bit #0 is the least significant bit. |
| NA# CAL_TYPE cc               | The type of calibration used:  
  1 = response calibration.  
  2 = response and isolation calibration.  
  3 = one-port calibration on port 1.  
  4 = one-port calibration on port 2.  
  5 = two-port calibration (includes one-path full & TRL) |
| NA# POWER_SLOPE ss            | Change in power versus frequency.  
  ss = dBm/GHz |
| NA# SLOPE_MODE mm             | On/off flag for power slope.  
  mm = 0 = off  
  mm = 1 = on |
| NA# TRIM_SWEEP tt             | Linearity adjustment value for swept sources. |
| NA# SWEEP_MODE ss             | Type of sweep done to make measurement.  
  0 = swept  
  1 = stepped  
  2 = single-point  
  3 = fast CW  
  4 = list |
| NA# LOWPASS_FLAG ff           | Low-pass time domain flag.  
  ff = 0 = low-pass time domain enabled.  
  ff = 1 = low-pass time domain disabled. |
| NA# FREQ_INFO ii              | The frequency information flag.  
  ii = 0 = frequency information displayed on instrument screen.  
  ii = 1 = frequency information not displayed on instrument screen. |
## Table 20-2. Network Analyzer Keyword Commands (continued)

<table>
<thead>
<tr>
<th>Statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA# DUPLICATES dd</td>
<td>Delete duplicates flag. Determines if points listed more than once should be measured more than once.</td>
</tr>
<tr>
<td></td>
<td>dd = 0 = points listed more than once are measured as many times as they are listed.</td>
</tr>
<tr>
<td></td>
<td>dd = 1 = points are measured only once.</td>
</tr>
<tr>
<td>NA# SPAN xx yy pp</td>
<td>The sweep parameters:</td>
</tr>
<tr>
<td></td>
<td>xx = start value</td>
</tr>
<tr>
<td></td>
<td>yy = stop value</td>
</tr>
<tr>
<td></td>
<td>pp = number of points</td>
</tr>
<tr>
<td>NA# IF_BW gg</td>
<td>The IF bandwidth setting of the receiver.</td>
</tr>
<tr>
<td></td>
<td>gg = IF bandwidth in Hertz.</td>
</tr>
</tbody>
</table>
Error Array Numbering

Current network analyzer implementations use between one and twelve error coefficient arrays to perform error correction. The **CAL_TYPE** keyword description in “Network Analyzer (#NA) Definitions” on page 20-97 lists the currently defined calibration types. Table A-3 defines the meanings of each coefficient array with respect to the error model used.

<table>
<thead>
<tr>
<th>Error Array Name</th>
<th>Frequency Response</th>
<th>Response &amp; Isolation</th>
<th>All 1-Port</th>
<th>All 2-Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Er or Et</td>
<td>Ed or Ex</td>
<td>Ed</td>
<td>Edf</td>
</tr>
<tr>
<td>E2</td>
<td>—</td>
<td>Er or Et</td>
<td>Es</td>
<td>Esf</td>
</tr>
<tr>
<td>E3</td>
<td>—</td>
<td>—</td>
<td>Er</td>
<td>Erf</td>
</tr>
<tr>
<td>E4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Exf</td>
</tr>
<tr>
<td>E5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Elf</td>
</tr>
<tr>
<td>E6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Etf</td>
</tr>
<tr>
<td>E7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Edr</td>
</tr>
<tr>
<td>E8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Esr</td>
</tr>
<tr>
<td>E9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Err</td>
</tr>
<tr>
<td>E10</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Exr</td>
</tr>
<tr>
<td>E11</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Elr</td>
</tr>
<tr>
<td>E12</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Etr</td>
</tr>
</tbody>
</table>
Disk Filename Requirements

Some instruments or programs identify a particular type of file by characters that are added before or after the file name. In general, an instrument or program should not require any such characters when reading a file.

There exist CITIfile implementations which do have file naming restrictions. This section explains how to work around these restrictions.

Agilent 8510 Series CITIfile

The 8510 checks the first 3 letters of the filename to determine what is stored in the file. The file prefixes for an 8510 CITIfile are listed in Table A-4.

<table>
<thead>
<tr>
<th>File Prefix</th>
<th>File Contents</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>Raw Data</td>
<td>Raw (uncorrected data array(s)).</td>
</tr>
<tr>
<td>DD</td>
<td>Data Data</td>
<td>Error corrected data array(s)</td>
</tr>
<tr>
<td>FD</td>
<td>Formatted Data</td>
<td>Corrected &amp; formatted data array.</td>
</tr>
<tr>
<td>DM</td>
<td>Display Memory</td>
<td>File holds one memory.</td>
</tr>
<tr>
<td>MA</td>
<td>Display Memory All</td>
<td>Holds all memories in 8510.</td>
</tr>
<tr>
<td>CS</td>
<td>Cal Set</td>
<td>One set of calibration data.</td>
</tr>
<tr>
<td>CA</td>
<td>Cal Set All</td>
<td>All sets of calibration data.</td>
</tr>
<tr>
<td>DT</td>
<td>Delay Table</td>
<td>One delay table.</td>
</tr>
</tbody>
</table>

DD_MYDATA is an example of a file name for a file that contains one array of corrected data.

The current 8510 CITIfile implementation is unable to read files unless they have the prefixes above. It is expected that a future 8510 revision will remove this restriction.
Working with Data Files

**Agilent 8700 Series CITIfile**

Storing a data file from an 8700-series analyzer in CITIfile format requires that you choose the **SAVE USING ASCII** option.

The 8700 series of instruments check the last two characters of the filename to determine what is stored in the file. The file endings for an 8700 CITIfile are listed in Table A-5.

<table>
<thead>
<tr>
<th>Last 2 Chars of File Name</th>
<th>File Contents</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx</td>
<td>Raw Data</td>
<td>x = 1 = channel 1. x = 5 for channel 2.</td>
</tr>
<tr>
<td>Dx</td>
<td>Data Data</td>
<td>x = channel number.</td>
</tr>
<tr>
<td>Fx</td>
<td>Formatted Data</td>
<td>x = channel number.</td>
</tr>
<tr>
<td>Mx</td>
<td>Display Memory</td>
<td>x = channel number.</td>
</tr>
<tr>
<td>xy</td>
<td>Cal Set</td>
<td>x = channel number. y = number of error coefficient arrays in the file. y is displayed in hexadecimal.</td>
</tr>
</tbody>
</table>

**FILE1D1** is an example of a file name for a file that contains corrected data for channel #1. **FILE1R5** is a filename for raw data arrays from channel #2. **MYFILE2C** is an example of a name for a file that contains a cal set used by channel #2, with 12 arrays of data (hexadecimal C).

To load data from disk into an 8700-series instrument, there must be a matching instrument state file to go with the data that is being loaded. Consult the 8700-series documentation for more information.
Writing Model Files

Nonlinear devices obtain their model parameters either from a model item or a file. For those devices that use a file, such as DIODEAP, this section discusses how to create a model file in the appropriate format.

Model files are text files that contain model parameter names and values. A sample model file for a DIODEAP device is shown below. Comments can be placed in the file by starting the line with REM. The model parameters are placed between BEGIN BDTA and END BDTA. One or more parameter names are placed on a line beginning with %; corresponding values are placed in the same order on the next line.

REM any line that starts with REM is ignored
BEGIN BDTA
% KM0D KVER
405 100
% Is Rs N Tt Cjo Vj
1e-14 10 1 5e-9 0 1
% M Eg Xti Kf Af Fc
0.5 1.11 3 0 1 0.5
% Bv Ibv Isr Nr Ikf Nbv
0 0 0 2 0 1
END BDTA

Parameter names for each device are listed in the documentation for that device in this chapter. Parameter names can be in any order and are not case sensitive. Any parameters that are not present in the file take on their documented default values. (The BJT AP, DIODEAP, J FETAP and MOSFETAP devices use the same parameter names and equations as the BJ TM, DIODEM, J FETM and MOSLVL3 model items.)

In each model file there are two special parameters, KM0D and KVER. KM0D identifies which device the model file can be used with. If this value is not correct (for example, if a DIODEAP device tries to read a BJ TAP model file with KM0D=403), the simulator will issue an error message Incompatible MDIF file (model). The KM0D values for each device are listed below. A KVER value must be present in the file but its value is not used by the simulator for user-written model files.

<table>
<thead>
<tr>
<th>Device</th>
<th>KM0D</th>
<th>KVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>BJT AP</td>
<td>403</td>
<td>EEFET2</td>
</tr>
<tr>
<td>DIODEAP</td>
<td>405</td>
<td>EEFET3</td>
</tr>
<tr>
<td>EEBJT</td>
<td>400</td>
<td>EEHEMT1</td>
</tr>
</tbody>
</table>
## Working with Data Files

<table>
<thead>
<tr>
<th>Models</th>
<th>402</th>
<th>Descriptions</th>
<th>404</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEBJT2</td>
<td></td>
<td>EEMOS1</td>
<td>600</td>
</tr>
<tr>
<td>EEBJT2A</td>
<td>402</td>
<td>EETOM1</td>
<td>105</td>
</tr>
<tr>
<td>EEBJT3</td>
<td>403</td>
<td>JFETAP</td>
<td>404</td>
</tr>
<tr>
<td>EEFET1</td>
<td>100</td>
<td>MOSFETAP</td>
<td>406</td>
</tr>
</tbody>
</table>
Index

A
absolute tolerance, 4-16
ac frequency conversion, 5-6
and S-parameter simulations, 6-13
AC Simulation component, 5-1
ac simulations, 5-1
about, 5-6
and dc simulations, 5-6
and harmonic balance simulations, 5-6
and noise, 5-5
for power amplifier analysis, 5-3
in RF system analysis, 18-2
performing, 5-2
ACDATA blocks, 20-29, 20-49
p2d examples, 20-53
aliasing, 7-23
ascsig file formats
See also time-domain waveform data files
examples, 20-81
SPW version 3.0 data file formats, 20-81

B
binning
defined, 4-6
bintim file formats, 20-70
budget
analysis, 18-8
measurement functions, 18-9, 18-10
measurements, 18-8
budget simulations
options, 18-9
BudLinearization component, 18-10

C
Circuit Envelope simulations
and FDDs, 16-12
envelope formulas, 16-11
in noise analysis, 16-15
of oscillators, 16-14
overview, 16-8
performing, 16-2
troubleshooting, 16-14
using V_1Tone source components, 16-10
versus harmonic balance, 16-8
circuit simulation options
using the Options component, 3-8
circuit simulations
about the simulators, 1-1
and measurements, 2-9
applications for, 1-3
constants, 2-5
current probes, 2-2
data points, 3-4
displaying results, 1-14, 3-10
displaying simulation options, 3-6
editing simulation parameters, 3-5
expressions, 1-9
functions, 2-6
instruments, 2-19
node sets, 2-4
options component, 1-7
preparing schematics, 2-1
reusing solutions, 1-7, 1-16
saving data, 1-10
selecting simulation type, 3-5
starting, 1-13, 3-9
stopping, 1-13
sweeping parameters, 1-8
sweeps, 3-7
using multiple components, 1-13
using templates, 2-12
variables, 1-9, 2-6
viewing DC node solutions, 1-15
viewing device operating points, 1-15
circuit sources
connecting, 3-2
CITIfiles
and HFS formats, 20-95
and network analyzers, 20-101
data arrays, 20-86
data conversion, 20-95
data format, 20-85
DOS format, 20-96
filename requirements, 20-101
guidelines, 20-94
header, 20-86
keyword reference, 20-91
keywords, 20-86
naming conventions, 20-101
network analyzers, 20-97
package, 20-85
specifications, 20-94
Comdisco SPW file formats, 20-81
constants
for simulation, 2-5
convergence
DC simulations, 4-10
conversion gain
mixers, 11-10
convolution analysis, 17-7
convolution simulations
See transient/convolution simulations
current probes
and circuit simulations, 3-4
for circuit simulation, 2-2
current tolerances, 4-16

D
data
collecting for circuit simulations, 3-4
data arrays
in CITIfiles, 20-86
data conversion
of CITIfiles, 20-95
data files
adding noise parameters, 20-12
discrete, 20-74
data formats
CITIfiles, 20-85
datasets
and circuit simulations, 1-10
DC node solutions
viewing, 1-15
DC Simulation component, 4-1
DC simulations
about, 4-8
and other simulations, 4-9
convergence, 4-16
convergence failures, 4-10
eliminating open-circuit errors, 4-12
impossible circuits, 4-15
node sets, 2-4
performing, 4-2
problems with I-V curves, 4-11
single-point simulation, 4-3
tolerance parameters, 4-16
troubleshooting, 4-9
using Options component to set tolerance
parameters, 4-16
dc solutions
viewing, 3-10
DC Block component
and S-parameter simulations, 6-9
DC Feed component
and S-parameter simulations, 6-9
device operating points
viewing, 1-15, 3-11
direct solver
harmonic balance simulations, 7-16
discrete data files, 20-74
DOS format
for CITIfiles, 20-96
dscr format, 20-74

E
envelope simulations. See Circuit
Envelope simulator
examples
CITIfile packages, 20-87
directories, 1-4
expressions
for circuit simulations, 1-9

F
FCAC (frequency-converting AC), 5-6, 6-8
FDDs
using with Circuit Envelope simulations, 16-12
file formats
s2d, 20-26
file names
CITIfiles, 20-101
frequency conversion, 5-6
and S-parameters, 6-13
effects on ac linear noise, 5-6
ports, 6-8
frequency translation
and S-parameter simulations, 6-8
functions
for simulation, 2-6
GaAsFETs and TAU parameter in transient/convolution simulations, 17-13

gain compression
definition, 15-8
gain compression simulations, 15-1
calculations of, 15-8
performing, 15-2
GCOMP1 data blocks, 20-34
GCOMP2 data blocks, 20-35
GCOMP3 data blocks, 20-36
GCOMP4 data blocks, 20-37
GCOMP5 data blocks, 20-38
GCOMP7 data blocks, 20-40
Gear’s method, 17-10
generalized mdif, 20-76
Gibb’s phenomenon, 17-16
global nodes
and circuit simulations, 2-3
G-parameters
file format, 20-7, 20-14
group delay, 6-6, 6-12
using S-parameter simulations to find, 18-23
guidelines
CITIfiles, 20-94

I
I (in-phase modulation component), 16-9
IMD (intermodulation distortion), 16-3
using circuit envelope simulations, 16-1
imt files, 18-27
imt format
intermodulation product table
file format, 20-68
IMTDATA blocks, 20-58
instruments
for circuit simulations, 2-19
integration methods
in transient/convolution simulations, 17-10
intermodulation distortion (IMD), 11-14, 16-3
intermodulation products, 18-26

K
KCL (Kirchoff’s Current Law), 4-10, 7-9
keywords
CITIfile, 20-86
reference, 20-91
Krylov solver
harmonic balance simulations, 7-16

L
large-signal S-parameter simulations
in RF system analysis, 18-3
performing, 13-2
layouts
simulating, 1-17
licenses
for simulators, 1-2
linear noise about, 5-8 calculating, 5-5 LSSP simulation component, 13-1

M
maximum order, 7-10
mdif
generalized format, 20-76 measurements for circuit simulations, 2-9 mixer noise simulations using NoiseCons, 12-5 mixer simulations and harmonic balance simulations, 11-1 and MixerIMT components, 11-3 and small-signal noise, 12-12 and Small-signal option, 11-1, 11-7 determining mixer noise, 12-2 finding conversion gain, 11-10 finding output tones, 11-3 fundamentals, 11-17 nonlinear spot noise, 12-12 nonlinear swept noise, 12-13 performing, 11-2 small-signal simulation, 11-7 small-signal simulations, 11-17 MixerIMT component, 11-3 mixers analyzing frequency translation, 6-8 finding intermodulation distortion (IMD), 16-3 model files, 20-103 Monte Carlo simulations and cumulative distribution function cdf(), 19-15 and Data Display, 19-14 enabling, 19-15 transient simulations, 19-15 MOSFETs and harmonic balance simulations, 7-22 multiple circuit simulations, 1-13

N
named nodes and circuit simulations, 2-3, 3-4 removing, 3-3 NDATA blocks, 20-32, 20-56 negative-resistance oscillators, 9-7, 9-11 network analyzers and CITIfiles, 20-97, 20-101 error correction, 20-100 filename requirements, 20-101 node sets for circuit simulations, 2-4 nodes and circuit simulations, 2-3 global, 2-3 highlighting, 2-4 named, 2-3 noise about, 6-14 flicker FM noise, 10-20 flicker PM noise, 10-20 in systems, 18-30 linear noise simulation and S-parameter simulations, 6-7 linear, calculating, 5-5 mixer noise, 12-2 Monte-Carlo in time domain, 17-12 noise parameter definitions, 5-9 nonlinear, 8-1 overview, 8-5 nonlinear noise, 12-12 nonlinear spot noise, 12-12 port thermal, 5-5 small-signal noise, 12-12 spot noise, 12-12 swept, 12-13 system noise, 18-30 using noise sources, 10-21 white PM noise, 10-20 noise analysis, 6-14 noise figure, 6-14 mixer, computing, 12-14 of mixers, 12-6 noise parameters, 5-6 adding to data files, 20-12 noise power single-sideband noise power, 10-19 noise simulations of mixers, 12-1 noise temperature using the Options component, 1-7 NoiseCon component for mixer simulations, 12-5
for phase noise, 10-4
overview, 8-6
using, 8-4
noisy two-port parameters, 6-14, 8-6
nonlinear devices
and harmonic balance simulation, 7-26
nonlinear noise simulations, 8-1
about, 8-5
performing, 8-2
with NoiseCons, 8-4
Nyquist criterion, 7-23

O
optimization
updating values, 1-9
options
circuit simulation, displaying, 3-6
Options component
setting tolerance parameters, 4-16
using, 1-7
order
and harmonic balance simulations, 7-10
oscillator loop gain, 9-21, 9-22
oscillator phase noise, 10-20
oscillator simulations
and Circuit Envelope simulations, 16-14
correcting for search failures, 9-26
finding oscillation frequency, 9-4
inaccurate results in, 9-26
of phase noise, 10-1
oscillator loop gain, 9-11, 9-16
overview, 9-21
performing, 9-2
power calculation problems, 9-21
probes, 9-2
steady-state conditions, 9-6
troubleshooting, 9-25
VCO tuning, 9-22
with harmonic balance, 9-1
oscillators
negative resistance, 9-7, 9-11, 9-24
OscPort component
for phase noise, 10-2
usage, 9-25
using, 9-2
OscPort2 component
using, 9-2, 9-14
OscTest component
using, 9-2, 9-11
output
controlling, 1-10
OutVar
output variable parameter, 3-7
using to send data to dataset, 11-12
oversampling, 7-23
P
P2D file
using, 14-6
P2D file formats, 20-46
P2D simulations
about, 14-1
performing, 14-2
Term component, 14-6
packages
CITIfile, 20-85
parameter sweeps
and hierarchical sweep plans, 19-1
and Monte Carlo simulations, 19-15
and placement of sweep components, 19-3
avoiding too much data, 19-15
candidate parameters for sweeping, 19-1
curve-tracer display, 19-4
elements, 19-4
of a single parameter, 19-2
order of execution, 19-3
troubleshooting, 19-14
using two SweepPlan components, 19-8
parameters
sweeping, 1-8
pdf
format, 20-79
phase distortion
and group delay, 6-12
and group delay measurement, 6-6
phase noise
about, 10-13
and noise figure, 12-8
and sideband level, 10-18
in oscillators, 10-20
of oscillators, 10-1
pnfm (FM noise), 10-3
pnmx (phase noise), 10-3
port thermal noise, 5-5
ports
pin mismatches, 4-14
tim file formats, 20-70
examples, 20-73
guidelines, 20-70
time domain simulations
versus harmonic balance, 7-10
TIMEDATA blocks, 20-71
time-domain waveform data file. See
.ascsig file formats
tolerance parameters, 4-16
and dc simulation, 4-16
topology checks
failures, 4-13
transient simulations
transient analysis, 17-7
transient/convolution simulations
about, 17-1
avoiding errors, 17-16
integration method, 17-10
noncausal frequency responses, 17-16
overview, 17-7
performing, 17-2
process, 17-8
reusing solutions, 17-11
setting convolution parameters, 17-14
transient versus convolution analysis,
17-7
troubleshooting, 17-12
using S-parameter data, 17-17
troubleshooting
DC simulations, 4-9
harmonic balance simulations, 7-18
oscillator simulations, 9-25

U
updating
optimization values, 1-9

V
V_1Tone source components
in Circuit Envelope simulations, 16-10
variables
for circuit simulations, 1-9
for simulation, 2-6
voltage tolerances, 4-16

W
wires
connecting, 3-2

X
XDB simulation component, 15-1
Xqc device parameter, 7-22

Y
Yield component, 19-15
YieldOptim component, 19-15
Y-parameters
file format, 20-7, 20-21

Z
Z-parameters
file format, 20-7, 20-21