



**Agilent Technologies**

**Advanced Design System 2011.01**

**February 2011**

**Harmonic Balance Simulation**

© **Agilent Technologies, Inc. 2000-2011**

5301 Stevens Creek Blvd., Santa Clara, CA 95052 USA

No part of this documentation may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

**Acknowledgments**

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. \* Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXIm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 <http://www.xs4all.nl/~kholwerd/bool.html> . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at <http://www.mozilla.org/MPL/> . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", <http://www.cs.umn.edu/~metis> , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel@ Math Kernel Library, <http://www.intel.com/software/products/mkl>

SuperLU\_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7-zip - 7-Zip Copyright: Copyright (C) 1999-2009 Igor Pavlov. Licenses for files are: 7z.dll: GNU LGPL + unRAR restriction, All other files: GNU LGPL. 7-zip License: This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program. All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: <http://www.7-zip.org/>

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: <http://www.cise.ufl.edu/research/sparse/amd>

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at

your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: <http://www.cise.ufl.edu/research/sparse/umfpack> UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at <http://www.cise.ufl.edu/research/sparse> . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at <http://www.cise.ufl.edu/research/sparse> . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. <http://www.mathworks.com> . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at <http://www.netlib.org> ). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: <http://www.qtsoftware.com/downloads> Patches Applied to Qt can be found in the installation at: `$(HPEESOF_DIR)/prod/licenses/thirdparty/qt/patches`. You may also contact Brian Buchanan at Agilent Inc. at [brian\\_buchanan@agilent.com](mailto:brian_buchanan@agilent.com) for more information.

The HiSIM\_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

**Errata** The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

**Warranty** The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

**Technology Licenses** The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at <http://systemc.org/> . This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

**Restricted Rights Legend** U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

|  |     |
|--|-----|
| Harmonic Balance Basics . . . . .                                      | 7   |
| Overview . . . . .   | 7   |
| Using Harmonic Balance Simulation . . . . .                            | 8   |
| Examples of Harmonic Balance Simulation . . . . .                      | 9   |
| Reference Equations . . . . .  | 14  |
| Limitations . . . . .  | 14  |
| HB Simulation Parameters . . . . .                                     | 14  |
| Theory of Operation . . . . .  | 34  |
| Troubleshooting a Simulation . . . . .                                 | 42  |
| Harmonic Balance for Nonlinear Noise Simulation . . . . .              | 50  |
| Performing a Nonlinear Noise Simulation . . . . .                      | 50  |
| Performing a Noise Simulation with NoiseCons . . . . .                 | 52  |
| Nonlinear Noise Simulation Description . . . . .                       | 52  |
| NoiseCon Component Description . . . . .                               | 53  |
| NoiseCon Component . . . . .   | 53  |
| Harmonic Balance for Oscillator Simulation . . . . .                   | 61  |
| Performing an Oscillator Simulation . . . . .                          | 61  |
| Performing an Oscillator Noise Simulation . . . . .                    | 64  |
| Examples of Oscillator Simulations . . . . .                           | 64  |
| Oscillator Simulation Description . . . . .                            | 83  |
| Phase Noise Simulation Description . . . . .                           | 86  |
| Troubleshooting a Simulation . . . . .                                 | 91  |
| Simulation Techniques for Recalcitrant Oscillators . . . . .           | 93  |
| Harmonic Balance for Mixers . . . . .                                  | 107 |
| Performing a Basic Mixer Simulation . . . . .                          | 107 |
| Examples of Mixer Simulations . . . . .                                | 108 |
| Small-Signal Mode Description . . . . .                                | 125 |
| Small-Signal Noise Simulation . . . . .                                | 126 |
| Transient Assisted Harmonic Balance . . . . .                          | 129 |
| Setting Additional Transient Parameters . . . . .                      | 129 |
| Using a One-Tone Transient for a Multi-Tone Harmonic Balance . . . . . | 130 |
| Using Sweeps and Optimization Simulations . . . . .                    | 131 |
| Outputting the Transient Data to the Dataset . . . . .                 | 131 |
| Harmonic Balance Assisted Harmonic Balance . . . . .                   | 132 |
| Modes of HBAHB Operation . . . . .                                     | 132 |
| HBAHB, Parameter Sweeps, and Noise . . . . .                           | 133 |
| HBAHB and TAHB . . . . .   | 133 |
| HBAHB and Non-Convergence . . . . .                                    | 133 |

# Harmonic Balance Basics

This is a description of Harmonic Balance (HB) simulation, including when to use it, how to set it up, and the data it generates. Examples are provided to show how to use this simulation. Detailed information describes the parameters, theory of operation, and troubleshooting information.

## Overview

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. Within the context of high-frequency circuit and system simulation, harmonic balance offers several benefits over conventional time-domain transient analysis. Harmonic balance simulation obtains frequency-domain voltages and currents, directly calculating the steady-state spectral content of voltages or currents in the circuit. The frequency integration required for transient analysis is prohibitive in many practical cases. Many linear models are best represented in the frequency domain at high frequencies. Use the HB simulation to:

- Determine the spectral content of voltages or currents.
- 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:

- [Using Harmonic Balance Simulation](#) explains when to use Harmonic Balance simulation, describes the minimum setup requirements, and gives a brief explanation of the Harmonic Balance simulation process.
- [Examples of Harmonic Balance Simulation](#) describes in detail how to set up a basic single-point and a swept harmonic balance simulation, using a power amplifier.
- [Reference Equations](#)
- [Limitations](#) describes the harmonic balance simulator's limitations.
- [HB Simulation Parameters](#) provides details about the parameters available in the HB Simulation controller in ADS.
- [Theory of Operation](#) is a brief description of the harmonic balance simulator.
- [Troubleshooting a Simulation](#) offers suggestions on how to improve a simulation.
- *Harmonic Balance for Nonlinear Noise Simulation* (cktsimhb) describes how to use the simulator for calculating noise.
- *Harmonic Balance for Oscillator Simulation* (cktsimhb) describes how to use the simulator with oscillator designs.
- *Harmonic Balance for Mixers* (cktsimhb) describes how to use the simulator with mixer designs.
- *Transient Assisted Harmonic Balance* (cktsimhb) describes how to use the automated TAHB to generate the transient initial guess for the Harmonic Balance simulation.

- *Harmonic Balance Assisted Harmonic Balance* (cktsimhb) describes how to use HBAHB when performing a multi-tone harmonic balance simulation so the simulator automatically selects which tones to use in generating the final HB solution.
- For the most detailed description about setting up, running, and converging a harmonic balance simulation, see *Guide to Harmonic Balance Simulation in ADS* (adshbapp).

## Using Harmonic Balance Simulation

This section describes when to use Harmonic Balance simulation, how to set it up, and the basic simulation process used to collect data.

### License Requirements

The Harmonic Balance simulation uses the Harmonic Balance Simulator license (sim\_harmonic) which is included with all Circuit Design suites except RF Designer. You must have this license to run Harmonic Balance simulations. You can work with examples described here and installed with the software without the license, but you will not be able to simulate them.

### When to Use Harmonic Balance Simulation

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

### How to Use Harmonic Balance Simulation

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 mixing order. This limits the number of mixing products to be considered in the simulation. For more information on this parameter, see [Harmonics and Maximum Mixing Order](#).
- Choose Auto Select option for Matrix Solver under the Solver tab in the Harmonic Balance controller. For tips on using this option, see [Selecting a Solver](#).



- You can use previous simulation solutions to speed the simulation process. For more information, see [Reusing Simulation Solutions](#).
- You can perform budget calculations as part of the simulation. For information on budget analysis, see *Using Circuit Simulators for RF System Analysis* (cktsim) in *Using Circuit Simulators* (cktsim).
- You can perform small-signal analysis. Enable the *Small-signal* option and fill in the fields under the Small-Sig tab. For details, see *Harmonic Balance for Mixers* (cktsimhb).
- You can perform nonlinear noise analysis. Select the *Noise* tab, enable the *Nonlinear noise* option, and fill in the fields in the Noise(1) and Noise(2) dialog boxes. For details, see *Harmonic Balance for Nonlinear Noise Simulation* (cktsimhb).
- If your design includes NoiseCon components, select the *Noise* tab, enable the *NoiseCons* option and fill in the fields. For more information, see *Harmonic Balance for Nonlinear Noise Simulation* (cktsimhb).
- If your design includes an OscPort component, enable *Oscillator* and fill in the fields under the *Osc* tab. *Harmonic Balance for Oscillator Simulation* (cktsimhb) focuses specifically on simulating oscillator designs.

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

## What Happens During Harmonic Balance Simulation

To perform a harmonic balance simulation, you only need to specify one or more fundamental frequencies and the order for each fundamental frequency. Agilent EEsof EDA recommends that all other parameters remain set to their default values. The simulator will set up the simulation in a proper way so that near optimal performance can be achieved without any additional parameter tweaking. For example, with Auto Select as the default choice for selecting matrix solvers, the simulator will determine whether the Direct Solver or the Krylov Solver is more effective for a particular circuit. For multi-tone HB simulations, the simulator will automatically determine whether to use Harmonic Balance Assisted Harmonic Balance (HBAHB) and how to set it up to achieve the optimal simulation speed.

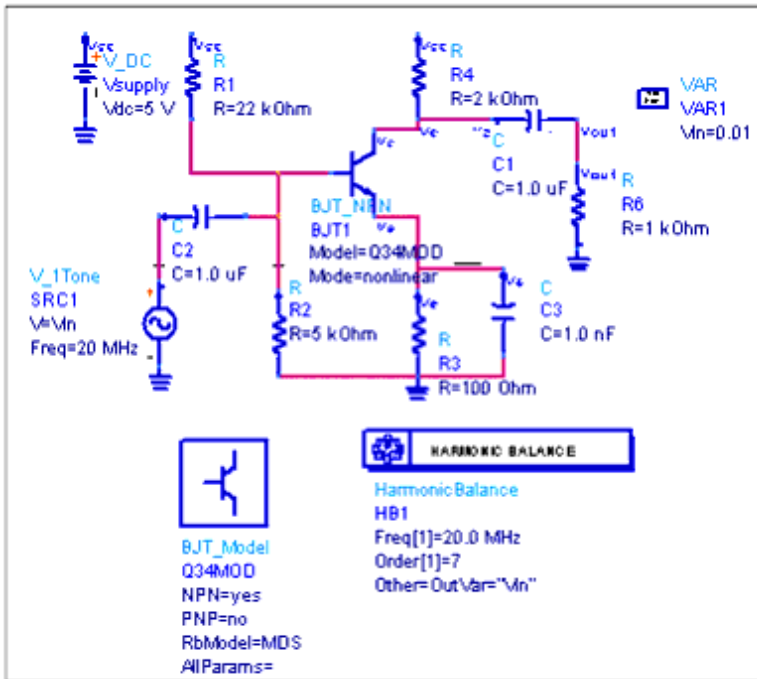
## Examples of Harmonic Balance Simulation

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

- [Single Tone Harmonic Balance Simulation](#) applies a single tone to the power amplifier. This tone and 7 harmonics are analyzed.
- [Swept Harmonic Balance Simulation](#) 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

The following figure illustrates the setup for simulating a power amplifier circuit.



#### Example setup for a basic harmonic balance simulation

#### Note

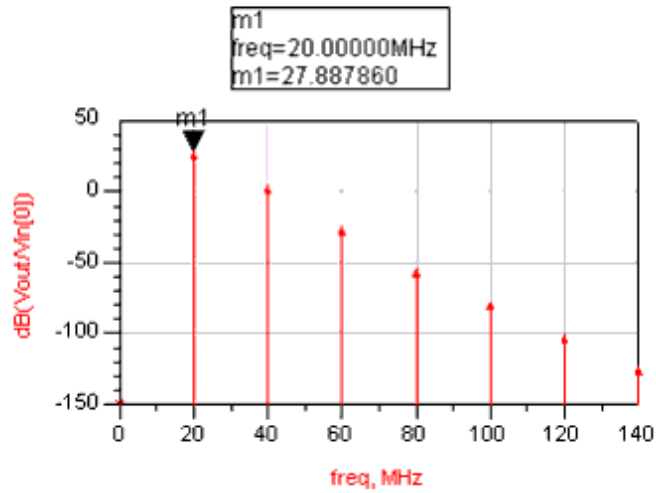
This design, *HB1*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *HB1.dds*.

- From the **Sources-Freq Domain** palette, select and place a **V\_1Tone** source on the schematic. Edit the component to set these values:
  - $V = 0.01 \text{ V}$
  - $\text{Freq} = 20 \text{ MHz}$ . This is the first and only fundamental.
- 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*.
- 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 mixing order* is not available.

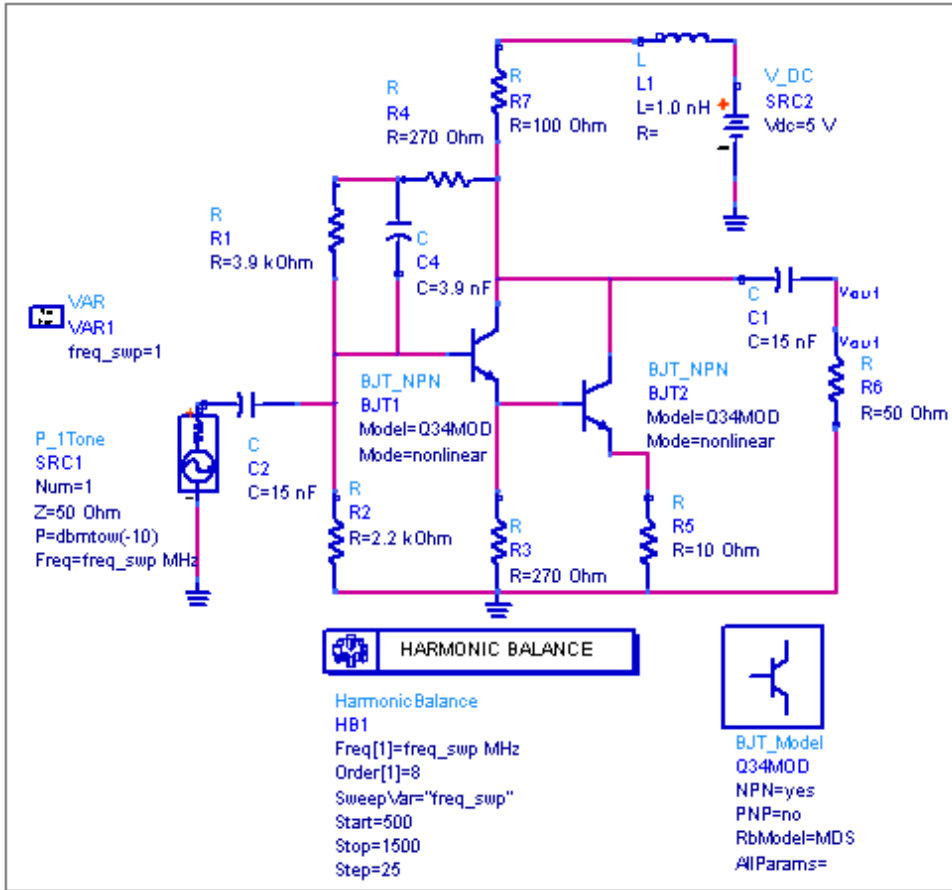
- Simulate**. When the simulation is finished, a Data Display window opens. The following plot illustrates the results of the simulation, showing the fundamental and seven harmonics, declining in voltage (*Vout*) with increasing frequency.

Fundamental and Harmonic Output  
Voltages Relative to Input Voltage

## 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.

The following figure illustrates the setup for a swept harmonic-balance simulation of a power amplifier circuit.



### Setup for a swept harmonic balance simulation

#### Note

This design, *HB2*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *HB2.dds*.

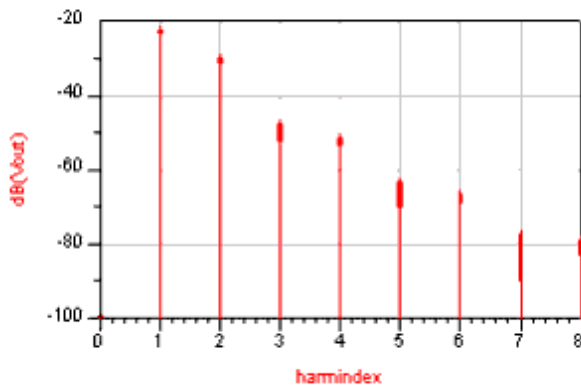
To perform a swept harmonic balance simulation:

- 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.
- Label the nodes at which you want data to be reported, in this example, label the output node as **Vout**.
- 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**
  - 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 **freq\_swp=1**. This assigns *freq\_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.



The harmonic index (*harminindex*) is a sequential index of frequencies, starting with zero. It is generated either during a frequency sweep, or during oscillator analysis (since the oscillation frequency as an unknown variable is swept during the analysis). The *harminindex* is always the innermost variable, which makes it the independent variable. Each *harminindex* value corresponds to an actual simulated frequency (harmonic or intermod product) for each frequency value of the frequency sweep. The following illustration shows the relationship between the LO, the RF frequency, and the harmonics generating the *harminindex* in a table of results. Assuming a downconverting mixer design, the

- LO freq = 16 GHz
- Rffreq (swept) = 12 GHz to 14 GHz in 2 GHz steps
- The Harmonic Balance controller is set to:
  - Freq [1] = LO (Order = 1)
  - Freq [2] = RF (Order = 1)
  - MaxOrder = 2

Here are the reported values for the simulation frequency *freq* relative to the *harmindex* value that would appear in the data display:

| harmindex | freq          |               |
|-----------|---------------|---------------|
|           | RFfreq=12 GHz | RFfreq=14 GHz |
| 0         | 0 GHz         | 0 GHz         |
| 1         | 4 GHz         | 2 GHz         |
| 2         | 12 GHz        | 14 GHz        |
| 3         | 16 GHz        | 16 GHz        |
| 4         | 28 GHz        | 30 GHz        |

## Reference Equations


For equations on which the harmonic balance simulation is based, please see *Harmonic Balance Background* (adshbapp) in the *Guide to Harmonic Balance Simulation in ADS* (adshbapp).

## Limitations

Harmonic balance is limited in that the signal must be quasi-periodic and representable as a superposition of a number  $M$  of discrete tones. As  $M$  becomes large, the amount of required internal memory becomes excessive since the internal matrix size grows as  $M^2$ . Using Krylov linear solvers instead of direct methods reduces the memory growth from quadratic to linear (proportional to  $M$ ). Therefore, the Krylov solvers enable harmonic balance to be used on very large circuits, and circuits with a large number of tones.

## HB Simulation Parameters

ADS provides access to harmonic balance simulation parameters enabling you to define aspects of the simulation listed in the following table:

|   |
|---|
|  <b>Important</b><br>For details about backward compatibility of default values for selected parameters, see <a href="#">Backward Compatibility Exceptions</a> . |
|---|

| Tab Name      | Description   | For details, see...  |
|---------------|---|--|
| Freq          | Frequencies of fundamentals.  | <a href="#">Setting Fundamental Frequencies</a>                    |
| Sweep         | Sweep type and associated characteristics.  | <a href="#">Setting Up a Sweep</a>                                 |
| Initial Guess | Sets parameters related to initial guess, including automated transient assisted harmonic balance (TAHB), harmonic balance assisted harmonic balance (HBAHB), initial guess from a data file, and initial guess for parameter sweep. It also allows the user to save the final solution in a data file. | <a href="#">Setting Up the Initial Guess</a>                       |
| Oscillator    | Enabling and setting up parameters for oscillator analysis.   | <a href="#">Enabling Oscillator Analysis</a>                       |
| Noise         | Parameters related to noise simulation, including sweeps, input and output ports, and the nonlinear noise controllers to be simulated.  | <a href="#">Selecting Nonlinear Noise Analysis</a>                 |
| Small-Sig     | Parameters related to small-signal/large-signal simulation.   | <a href="#">Setting Up Small-Signal Simulations</a>                |
| Params        | Parameters related to status level for summary information and device operating-point levels, as well as parameters related to FFT oversampling and convergence.  | <a href="#">Defining Simulation Parameters</a>                     |
| Solver        | Parameters enabling you to choose between a Direct or Krylov solver or an automatic selection. The automatic selection is the default and recommended choice, since it allows the simulator to choose the most effective solver for each particular circuit. Additional parameters manage memory usage. | <a href="#">Selecting a Harmonic Balance Solver Technique</a>      |
| Output        | Selectively save simulation data to a dataset.  | <i>Selectively Saving and Controlling Simulation Data</i> (cktsim) |
| Display       | Control the visibility of simulation parameters on the Schematic.   | <i>Displaying Simulation Parameters on the Schematic</i> (cktsim)  |

For additional information on setting up a harmonic balance simulation, refer to these topics:

- *Harmonic Balance for Nonlinear Noise Simulation* (cktsimhb) describes how to use the simulator for calculating noise.
- *Harmonic Balance for Oscillator Simulation* (cktsimhb) describes how to use the simulator with oscillator designs.
- *Harmonic Balance for Mixers* (cktsimhb) describes how to use the simulator with mixer designs.
- For a thorough description about setting up, running, and converging a harmonic balance simulation, see *Guide to Harmonic Balance Simulation in ADS* (adshbapp).

## Setting Fundamental Frequencies

On the *Freq* tab, you can specify the frequency portion of the simulation. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

## HB Simulation Frequency Parameters

| Setup Dialog Name       | Parameter Name | Description  |
|-------------------------|----------------|--|
| Fundamental Frequencies |                |  |
| Edit                    |                | Edit the Frequency and Order fields, then use the buttons to Add the frequency to the list displayed under Select.   |
| Frequency               | Freq[n]        | The frequency of the fundamental(s). Change by typing over the entry in the field. Select the units (None, Hz, kHz, MHz, GHz) from the drop-down list.   |
| Order                   | Order[n]       | <p>The maximum order (harmonic number) of the fundamental(s) that will be considered. Change by typing over the entry in the field.</p> <p>The number of harmonics need to be sufficiently large to represent nonlinear signals (sharp transitions, square waves). An increase in the Order slows down the simulation considerably or results in excessive memory usage. Use the Krylov solver if the problem is too big for the Direct solver.</p> <p>The lower the Order, the greater the Harmonic Balance truncation error as a result of the Fourier truncation in the solution representation. As a rule of thumb, anything below 5-7 harmonics is unacceptable. Using the Manual Convergence Mode (ConvMode=1) and StatusLevel=4 or 5 will give an estimate of this error.</p> <p>The computational complexity of the Krylov solver is determined by the size of FFT (i.e. by the number of samples). With Krylov Harmonic Balance, set Order to 7, 15, 31, etc.</p> <p>Keep in mind that according to the Nyquist theorem at least <math>2 \times \text{Order} + 1</math> samples are needed to represent the highest harmonic. The Oversample parameter increases the number of samples beyond the minimum by this factor, and, due to the nature of FFT, the number of samples is rounded up to the nearest power of 2.</p> |
| Select                  |                | <p>Contains the list of fundamental frequencies. Double-click in the Edit field to add fundamental frequencies to this window.</p> <p><i>Add</i> enables you to add an item.</p> <p><i>Cut</i> enables you to delete an item.</p> <p><i>Paste</i> enables you to take an item that has been cut and place it in a different order.</p> <p>Run transient analysis long enough to approach steady state.</p> <ul style="list-style-type: none"> <li>- At least 4-5 periods of excitation frequency</li> <li>- Use fixed time-step (make sure it is small enough)</li> <li>- Apply window (HB_Window=yes) to smooth the transient waveforms</li> </ul>  |
| Maximum mixing order    | MaxOrder       | <p>The maximum order of the intermodulation terms in the simulation. The combined order is the sum of the individual frequency orders that are added or subtracted to make up the frequency list. For example, assume there are two fundamentals and Order (see below) is 3.</p> <p>If Maximum mixing order is 0 or 1, no mixing products are simulated. The frequency list consists of the fundamental and the first, second, and third harmonics of each source.</p> <p>If Maximum mixing order is 2, the sum and difference frequencies are added to the list.</p> <p>If Maximum mixing order is 3, the second harmonic of one source can mix with the fundamental of the others, and so on.</p>  |
| Levels                  |                | Enables you to set the level of detail in the simulation status report.  |
| Status level            | StatusLevel    | <p>Prints information about the simulation in the Status/Summary part of the Message Window.</p> <ul style="list-style-type: none"> <li>- 0 reports little or no information, depending on the simulation engine.</li> </ul>   |



- 1 and 2 yield more detail.

- Use 3 and 4 sparingly since they increase process size and simulation times considerably.

The type of information printed may include the sum of the current errors at each circuit node, whether convergence is achieved, resource usage, and where the dataset is saved. The amount and type of information depends on the status level value and the type of simulation.

## Setting Up a Sweep

On the *Sweep* tab, setting up the sweep portion of the simulation consists of two basic parts:

- Selecting the sweep type and setting the associated characteristics
- Optionally, specifying a sweep plan

To shorten simulation time in any parameter sweep, select a start point as close as possible to the convergence point and vary the parameter gradually. This yields better estimates for the next simulation, and achieves convergence more rapidly than if the parameter were changed abruptly. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Sweep Parameters

| Setup Dialog Name  | Parameter Name                       | Description   |
|--|--------------------------------------|---|
| Parameter to sweep   | SweepVar                             | The name of the parameter to be swept. Choose a parameter that can be set to a value for which the circuit will easily converge: source amplitude / power (amplifiers), bias voltage / current, or any component parameter that controls the amount of non-linearity in the circuit.<br>Find the parameter value for which the circuit converges. This is the start point of the sweep (e.g. smaller input power).<br>The actual parameter value for which the circuit does not converge is the end point of the sweep. |
| Parameter sweep-The sweep type and parameters.   |                                      |   |
| Sweep Type   |                                      | A linear sweep works best in most cases Make sure Restart on the HB Initial Guess tab is not checked so that the sweep is used as a continuation (solution from previous sweep step used as an initial guess for the next step).<br>The number of sweep points (or step-size) controls the continuation step. The more sweep points, the greater the chances of success, but longer computation time as well.   |
| Single point   | Pt                                   | Enables simulation at a single frequency point. Specify the desired value in the Parameter field.   |
| Linear   |                                      | Enables sweeping a range of values based on a linear increment. Click Start/Stop to set start and stop values for the sweep, or Center/Span to set the center value and a span of the sweep.  |
| Log  |                                      | Enables sweeping a range of values based on a logarithmic increment. Click Start/Stop to set start and stop values for the sweep, or Center/Span to set the center value and a span of the sweep.   |
| Start/Stop<br>Start, Stop,<br>Step-size,<br>Pts./decade,<br>Num. of pts.   | Start<br>Stop<br>Step<br>Dec<br>Lin  | Select the Start/Stop option to sweep based on start, stop, step-size or pts./decade, and number of points. Linear sweep uses Step-size; Log sweep uses Pts./decade.<br>- Start-the start point of a sweep<br>- Stop-the stop point of a sweep<br>- Step-size-the increments at which the sweep is conducted<br>- Pts./decade-number of points per decade<br>- Num. of pts.-the number of points over which sweep is conducted  |
| Center/Span<br>Center, Span,<br>Step-size,<br>Pts./decade,<br>Num. of pts.   | Center<br>Span<br>Step<br>Dec<br>Lin | Select the Center/Span option to sweep based on center and span, step-size or pts./decade, and number of points. Linear sweep uses Step-size; Log sweep uses Pts./decade.<br>- Center-the center point of a sweep<br>- Span-the span of a sweep<br>- Step-size-the increments at which the sweep is conducted<br>- Pts./decade-number of points per decade<br>- Num. of pts.-the number of points over which sweep is conducted   |
| Note: Changes to any of the Start, Stop, etc. fields causes the remaining fields to be recalculated automatically. |                                      |   |
| Use sweep plan   | SweepPlan                            | Enables use of an existing sweep plan component. Select this option and enter the name of the plan or select it from the drop-down list.  |

## Setting Up the Initial Guess

On the *Initial Guess* tab, setting up the initial guess for a harmonic balance simulation consists of:

- Setting Transient Assisted Harmonic Balance (TAHB).
- Setting Harmonic Balance Assisted Harmonic Balance (HBAHB).
- Setting Initial Guess and Final Solution parameters.

To set up a TAHB analysis:

- On the **Initial Guess** tab in the Harmonic Balance controller, select **Auto**, **On**, or **Off** for Transient Assisted Harmonic Balance.

It is recommended to use the TAHB *Auto* mode, which is the default setting. The simulator will turn on TAHB automatically if the circuit involves a divider. The TAHB *On* and *Off* choices are for you to manually turn on or off TAHB, which should be done only when you would like to override the simulator's automatic setting.

To set up a HBAHB analysis:

- Under the **Initial Guess** tab, in the HBAHB section, select either **Auto**, **On**, or **Off**.

The *Auto* mode is the default and is recommended, which allows the simulator to determine whether to use HBAHB and to optimize the HBAHB setup if it is used. Selecting the *On* mode forces HBAHB to be turned on and the default sequencing (1-tone, 2-tone, ...) will be used. Selecting the *Off* mode forces HBAHB to be turned off.

By using HBAHB, the simulator will generate its own initial guess for multi-tone Harmonic Balance from another harmonic balance analysis with fewer fundamental frequencies than the original multi-tone problem. You do not need to supply an initial guess. If you do provide an initial guess by enabling *Use Initial Guess* and entering a name for *File* (parameters *UseInFile* and *InFile* ), then that will take precedence over HBAHB so long as the file exists.

The following table shows the parameters available to set the *Initial Guess*. Names listed in the *Parameter Name* column are used in netlists and on schematics.

#### HB Simulation Initial Guess Parameters

| Setup Dialog Name   | Parameter Name | Description  |
|---|----------------|--|
| Transient Assisted Harmonic Balance (TAHB)  |                |  |
| Transient Assisted Harmonic Balance   | TAHB_Enable    | Set the TAHB mode to Auto (default), On, or Off. Auto is set automatically if the circuit contains a divider. Choose On or Off to override the default settings. The Advanced Transient Settings are available when On is set.   |
| Transient Setup - available with Advanced Transient Settings when <i>Transient Assisted Harmonic Balance</i> is <i>On</i> . |                |  |
| Transient StopTime  | StopTime       | This is the transient stop time. The default is 100 cycles of the commensurate frequency. The commensurate frequency for a single tone simulation will be <i>Freq</i> [1]. If steady state is detected earlier than the <i>StopTime</i> , then transient will end earlier than the |

|   |                    |  |
|---|--------------------|--|
|   |                    | <i>StopTime</i> .  |
| Transient<br>MaxTimeStep  | MaxTimeStep        | This is the transient maximum time step. The default is $1/(8 \times \text{Maximum frequency})$ .  |
| Additional Transient Settings - available with Advanced Transient Settings when <i>Transient Assisted Harmonic Balance</i> is On. |                    |  |
| Min Time for detecting steady state   | SteadyStateMinTime | This is the earliest point in time that the transient simulator starts checking for steady state conditions. If your circuit exhibits a large amount of over/undershoot, then this needs to be larger than the default so that the detector will begin to check for steady state after some of the initial transients have settled.  |
| Transient<br>IV_RelTol  | IV_RelTol          | This is the transient relative voltage and current tolerance. The default is $1e-3$ . When simulation options are included in the simulation (using the Options controller), use this value to set specific relative tolerances to be used for transient only. The value will be used for both current and voltage relative tolerance for transient.   |
| Transient<br>Other  | AddlTranParamsTAHB | Enables ability to set other transient simulation parameters that are not found in this dialog box. For example, use this parameter to set the following transient convolution parameter <i>ImpMaxFreq=10 GHz</i> .  |
| Use only<br>Freq[1] for<br>transient  | OneToneTranTAHB    | Tells the simulator to perform a single tone transient simulation for a multitone harmonic balance simulation. The default setting is enabled.   |
| Save transient<br>data to dataset   | OutputTranDataTAHB | When enabled, the transient simulation data used in generating the initial guess is output to the dataset, in addition to the final harmonic balance data. For large circuits, this can cause the datasets to become quite large.  |
| Harmonic<br>Balance Assisted<br>Harmonic<br>Balance   | HBAHB_Enable       | Set the HBAHB mode to Auto, On, or Off.  |
| Initial Guess   |                    |  |
| Use Initial<br>Guess  | UseInFile          | <p>Check this box to enter a file name for a solution to be used as initial guesses. This file is typically generated from a previous simulation by enabling <i>Write Final Solution</i>. If no initial guess file name is supplied, a default name (using DC solution) is generated internally, using the design name and appending the suffix <i>.hbs</i>. A suffix is neither required nor added to any user-supplied file name. For example, if you have saved the Harmonic Balance solution from a previous simulation, you can later do a nonlinear noise simulation and use this saved solution as the initial guess, removing the time required to recompute the nonlinear Harmonic Balance solution. Or you could quickly get to the initial Harmonic Balance solution, then sweep a parameter to see the changes. In this latter case, you will probably either want to disable the <i>Write Final Solution</i> option or use a different file name for the final solution to avoid overwriting the initial guess solution. (See <i>Write Final Solution</i> in the Final Solution section below).</p> <p>The Annotate value specified in the DC Solutions tab in the Options block is also used to control the amount of annotation generated when there are topology changes detected during the reading of the initial guess file. Refer to <i>DC Simulation</i>. Since HB simulations also utilize the DC solution, to get optimum speed-up, both the DC solution and the HB solution should be saved and re-used as initial</p> |

|   |                   |  |
|---|-------------------|--|
|   |                   | <p>guesses.</p> <p>The initial guess file does not need to contain all the HB frequencies. For example, one could do a one-tone simulation with just a very nonlinear LO, save that solution away 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.</p> <p>If the simulator cannot converge with the supplied initial guess, it then attempts to a global node-setting by connecting every node through a small resistor to an equivalent source. It then attempts to sweep this resistor value to a very large value and eventually tries to remove it.</p> |
| File  | InFile            | Specify a file name to save results.   |
| Regenerate Initial Guess for ParamSweep (Restart) | Restart=yes or no | Instructs the simulator to not use the last solution as the initial guess for the next solution.   |
| Final Solution                                    |                   |  |
| Write Final Solution                              | UseOutFile        | <p>Check this box to save your final HB solution to the output file. If a file name is not supplied, a file name is internally generated using the design name, followed by an <i>.hbs</i> suffix. If a file name is supplied, the suffix is neither appended nor required. If this box is checked, then the last HB solution is put out to the specified file. If this is the same file as that used for the Initial Guess, this file is updated with the latest solution.</p> <p>Transient simulations can also be programmed to generate a harmonic balance solution that can then be used as an initial guess for an HB simulation. Refer to <i>Harmonic Balance Simulation</i>.</p>   |
| File  | OutFile           | Specify a file name to save results.   |

## Enabling Oscillator Analysis

On the *Oscillator* tab, setting up an oscillator analysis consists of:

- Enabling Oscillator Analysis.
- Setting the simulation method to use OscPort or nodes (OscProbe).
- If specifying nodes, enter the node parameter values.

The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Oscillator Analysis Setup

| Setup Dialog Name   | Parameter Name | Description  |
|---|----------------|--|
| Enable Oscillator Analysis  | OscMode        | Choose this if you want an oscillator analysis.  |
| Method  |                | Select <i>Use Oscport</i> if an OscPort or OscPort2 component is being used. There is no need to specify the name of the OscPort component; the simulator will find it automatically. Select <i>Specify Nodes</i> to use the OscProbe method which does not use an OscPort or OscPort2 component.  |
| Specify Oscillator Nodes - The following parameters are available only when selected Method is <i>Specify Nodes</i> . |                |  |
| Node Plus   | OscNodePlus    | This is the required name of a named node in the oscillator. Recommended nodes are those at the input or output of the active device, or in the resonator. Hierarchical node names are permitted.  |
| Node Minus  | OscNodeMinus   | This second node name should only be specified for a differential (balanced) oscillator. Leave it blank for single-ended oscillators. Node Plus and Node Minus should be chosen symmetrically. Hierarchical node names are permitted.  |
| Fundamental Index   | OscFundIndex   | Specifies which of the fundamental frequencies is to be treated as the unknown oscillator frequency which the simulator will solve for. The default value of 1 means that Freq[1] is the unknown.  |
| Harmonic Number   | OscHarmNum     | Specifies which harmonic of the fundamental frequency is to be used for the oscillator. Normally this parameter stays at its default setting of 1. If an oscillator followed by a frequency divider is to be analyzed, this parameter should be set to the frequency divider ratio.  |
| Octaves to Search   | OscOctSrch     | Specifies the number of octaves used in the initial frequency search during oscillator analysis. This many octaves are searched, centered around the frequency specified you specify for the Fundamental Tone. To skip the initial frequency search, provide a good initial guess of the frequency for Fundamental Tone, and set this parameter to zero. |
| Steps per Octave  | OscOctStep     | Specifies the number of steps per octave used in the initial frequency search. A high-Q oscillator may require a much larger value, such as 1000, in order for the search to find the phase shift at resonance.  |

## Selecting Nonlinear Noise Analysis

The NoiseCon is used to select which *NoiseCon* nonlinear noise controllers should be simulated with the current harmonic balance analysis. These noise simulations will be performed in addition to any noise simulation that may be set up with *Noise(1)* and *Noise(2)* dialog boxes. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Noise Parameters

| Setup Dialog Name | Parameter Name | Description   |
|-------------------|----------------|---|
| NoiseCons         | NoiseConMode   | The NoiseCons check-box in the tab must be clicked to enable noise simulation with NoiseCons. This button can be used to disable noise simulation of all NoiseCons without deleting them from the Select NoiseCons list.  |
| Select NoiseCons  |                |   |
| Edit              |                | Specifies the name of the NoiseCon item to add to the simulation list.  |
| Select            | Noisecon[n]    | Holds the names of the NoiseCon items to be simulated.<br>- Add -- adds a NoiseCon name.<br>- Cut -- deletes a NoiseCon name.<br>- Paste -- takes a NoiseCon name that has been cut and places it in a different order in the Select window.  |
| Nonlinear noise   | NLNoiseMode    | Click to enable nonlinear noise analysis with harmonic balance.   |
| Noise (1)...      |                | Choose Noise (1) to set up and run a small-signal/Harmonic Balance noise analysis following the final time point of the simulation.   |
| Noise (2)...      |                | Defining the Noise2 parameters consists of the following basic parts:<br>- Enable noise calculation.<br>- Specifying the nodes to use for noise parameter calculation.<br>- Specifying the noise contributors and the threshold for noise contribution.<br>- Optionally, specifying the bandwidth over which the noise simulation is performed. |

## Setting Up Parameters for Noise(1)

Choose Noise (1) to set up and run a small-signal/Harmonic Balance noise analysis following the final time point of the simulation. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Noise(1) Parameters

| Setup Dialog Name  | Parameter Name   | Description   |
|--|--|---|
| Noise frequency  |  |   |
| Sweep Type   |  |   |
| Single point   | FreqForNoise   | Enables simulation at a single frequency point. Specify the desired value in the Parameter field.   |
| Linear   |  | Enables sweeping a range of values based on a linear increment. Click the Start/Stop option to select start and stop values for the sweep.  |
| Log  |  | Enables sweeping a range of values based on a logarithmic increment. Click the Center/Span option to select a center value and a span of the sweep.   |
| Start/Stop<br>Start, Stop,<br>Step-size,<br>Num. of pts.   | NLNoiseStart<br>NLNoiseStop<br>NLNoiseStep<br>NLNoiseLin | Select the Start/Stop option to sweep based on start, stop, step-size and number of points.<br>- Start -- the start point of a sweep<br>- Stop -- the stop point of a sweep<br>- Step-size -- the increments at which the sweep is conducted<br>- Num. of pts. -- the number of points over which sweep is conducted  |
| Center/Span<br>Center, Span,<br>Pts./decade,<br>Num. of pts.   | NLNoiseCenter<br>NLNoiseSpan<br>NLNoiseDec<br>NLNoiseLin | Select the Center/Span option to sweep based on center and span.<br>- Center -- the center point of a sweep<br>- Span -- the span of a sweep<br>- Pts./decade -- number of points per decade<br>- Num. of pts. -- the number of points over which sweep is conducted  |
| Note: Changes to any of the Start, Stop, etc. fields causes the remaining fields to be recalculated automatically. |  |   |
| Use sweep plan   | NoiseFreqPlan  | Enables use of an existing sweep plan component (SweepPlan). Select this option and enter the name of the plan or select it from the drop-down list.  |
| Input Frequency  | InputFreq  | Because the simulator uses a single-sideband definition of noise figure, the correct input sideband frequency must be specified here. This parameter identifies which input frequency will mix to the noise frequency of interest. In the case of mixers, Input frequency is typically determined by an equation that involves the local oscillator (LO) frequency and the noise frequency. Either the sum of or difference between these two values is used, depending on whether upconversion or downconversion is taking place.<br>The above parameters do not need to be specified if only the output noise voltage is desired (that is, if no noise figure is computed). |
| Noise input port   | NoiseInputPort   | Number of the source port at which noise is injected. This is commonly the RF port. Although any valid port number can be used, the input port number is frequently defined as Num=1.   |
| Noise output port  | NoiseOutputPort  | Number of the Term component at which noise is retrieved. This is commonly the IF port. Although any valid port number can be used, the output port number is frequently defined as Num=2.  |

## Setting Up Parameters for Noise(2)

Defining the Noise2 parameters consists of the following basic parts:

- Enable noise calculation.
- Specifying the nodes to use for noise parameter calculation.
- Specifying the noise contributors and the threshold for noise contribution.
- Optionally, specifying the bandwidth over which the noise simulation is performed



The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Noise(2) Parameters

| Setup Dialog Name   | Parameter Name                       | Description  |
|---|--------------------------------------|--|
| Nodes for noise parameter calculation   | NoiseNode[n]                         | The fewer the number of nodes requested, the quicker the simulation and the less memory required.  |
| Edit  |                                      | Selects the named node(s) for the simulator to consider  |
| Select  |                                      | <p>Holds the names of the nodes the simulator will consider.</p> <ul style="list-style-type: none"> <li>- Add -- adds a named node.</li> <li>- Cut -- deletes a named node.</li> <li>- Paste -- takes a named node that has been cut and places it in a different order in the Select window.</li> </ul>   |
| <p>Noise contributors - Use this area to sort the noise contributors list and to select a threshold below which noise contributors will not be reported. A list shows how each component contributes to noise at a specific node. The noise contributor data are always in units of V/sqrt(Hz) for noise voltages, and A/sqrt(Hz) for noise currents; they do not scale with noise bandwidth.</p> |                                      |  |
| Mode  | SortNoise                            | Provides options for sorting noise contributors by value or name.  |
| Off   | Off                                  | Causes no individual noise contributors to be selected. The result is simply a value for total noise at the output.  |
| Sort by value   | Sort by value                        | Sorts individual noise contributors, from largest to smallest, that exceed a user-defined threshold (see below). The subcomponents of the nonlinear devices that generate noise (such as Rb, Rc, Re, Ib, and Ic in a BJT) are listed separately, as well as the total noise from the device.   |
| Sort by name  | Sort by name                         | Causes individual noise contributors to be identified and sorts them alphabetically. The subcomponents of the nonlinear devices that generate noise (such as Rb, Rc, Re, Ib, and Ic in a BJT) are listed separately, as well as the total noise from the device.   |
| Sort by value with no device details  | Sort by value with no device details | Sorts individual noise contributors, from largest to smallest, that exceed a user-defined threshold (see below). Unlike <i>Sort by value</i> , only the total noise from nonlinear devices is listed without any subcomponent details.   |
| Sort by name with no device details   | Sort by name with no device details  | Causes individual noise contributors to be identified and sorts them alphabetically. Unlike <i>Sort by name</i> , only the total noise from nonlinear devices is listed without subcomponent details.  |
| Dynamic range to display  | NoiseThresh                          | A threshold below the total noise, in dB, that determines what noise contributors are reported. All noise contributors less than this threshold will be reported. For example, assuming that the total noise voltage is 10 nV, a setting of 40 dB (a good typical value) ensures that all noise contributors up to 40 dB below 10 nV (that is, up to 0.1 nV) are reported. The default of 0 dB causes all noise contributors to be reported. This parameter is used only with <i>Sort by value</i> and <i>Sort by value with no device details</i> . |

|   |                   |  |
|---|-------------------|--|
| Include port noise in node noise voltages | IncludePortNoise  | Causes port noise to be included in noise currents and voltages. Ports must be placed and defined.   |
| Calculate noisy two-port parameters       | NoisyTwoPort      | Causes an S-parameter simulation to be performed. Ports must be placed and defined. The Noise input port parameter should be set equal to the port number specified by the Num parameter on the input source, and the Noise output parameter to the number of the output Term (termination) component to Num=2. The following two-port parameters (dataset variables) are then returned and can be plotted: <ul style="list-style-type: none"> <li>- NFmin -- minimum noise figure of a two-port circuit. It is equal to the noise figure when the optimum source admittance is connected to the circuit. (Its default unit is dB).</li> <li>- Sopt -- the optimum source match for a two-port circuit. It is the reflection coefficient (looking into the source) that gives the minimum noise figure.</li> <li>- Rn -- the effective noise resistance in ohms (unnormalized) of a two-port circuit. Effective noise resistance can be used to plot noise-figure circles or related quantities. This parameter determines how rapidly the minimum noise figure deteriorates when the source impedance is not at its optimum value.</li> <li>- Icor -- the noise current correlation matrix, in units of Amperes squared. It describes the short circuit noise currents squared at each port, and the correlation between noise currents at different ports.</li> </ul> These expressions for noise simulation can be manipulated in equations. At low powers, NFmin agrees with NFssb and both match the noise figure found from a small-signal analysis. As the input power increases and nonlinear devices compress, this is no longer the case. NFmin and NFssb both deteriorate from their small-signal values. NFmin is an approximation to NFssb which neglects higher order conversion gains, and the difference between NFmin and NFssb expresses the importance of these higher order conversion gains. For most applications, compression is moderate and NFmin and NFssb are close. For applications driven into severe compression, NFmin and NFssb can differ significantly. Also, note that Sopt is only defined at one frequency and that severe compression may demand a matching network that takes into account other frequencies. Deep into compression, care should be taken when using the noisy two-port parameters for design. |
| Use all small-signal frequencies          | UseAllSS_Freqs    | Use all small-signal frequencies causes the simulator to solve for all small-signal mixer sidebands. This default option requires more memory but delivers more accurate results. In addition, it may require large kernel swap-size parameters. Only if there is insufficient memory should this option be set to no. Setting this option to no, causes only half of the small-signal mixer sidebands to be used and also uses one-fourth of the memory, but at the cost of generating potentially inaccurate results. Exercise caution when setting this option to no.<br>Note: If you find you are running out of RAM, either set this parameter to no after reading the paragraph above, or switch to the Krylov option.   |
| Bandwidth                                 | BandwidthForNoise | Bandwidth for spectral noise simulation. 1 Hz is the recommended bandwidth for measurements of spectral noise power. The noise contributor data do not scale with noise bandwidth.   |

## Setting Up Small-Signal Simulations

The *Small-Sig* tab enables you to use a large-signal/small-signal method to achieve faster

simulations when some signal sources are much smaller than others, and are assumed not to exercise circuit nonlinearities. For example, in a mixer the LO tone can be considered the large-signal source and the RF the small-signal source. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Small-Signal Parameters

| Setup Dialog Name  | Parameter Name                                      | Description   |
|--|---|---|
| Small-signal   | SS_MixerMode  | Choose this if you want a small-signal analysis.  |
| Small-signal frequency   |   |   |
| Sweep Type   |   |   |
| Single point   | SS_Freq   | Enables simulation at a single frequency point. Specify the desired value in the Frequency field.   |
| Linear   |   | Enables sweeping a range of values based on a linear increment. Click Start/Stop to set start and stop values for the sweep, or Center/Span to set the center value and a span of the sweep.  |
| Log  |   | Enables sweeping a range of values based on a logarithmic increment. Click Start/Stop to set start and stop values for the sweep, or Center/Span to set the center value and a span of the sweep.   |
| Start/Stop<br>Start, Stop,<br>Step-size,<br>Pts./decade,<br>Num. of pts.   | SS_Start<br>SS_Stop<br>SS_Step<br>SS_Dec<br>SS_Lin  | Select the Start/Stop option to sweep based on start, stop, step-size or pts./decade, and number of points. Linear sweep uses Step-size; Log sweep uses Pts./decade. <ul style="list-style-type: none"> <li>- Start -- the start point of a sweep</li> <li>- Stop -- the stop point of a sweep</li> <li>- Step -- size-the increments at which the sweep is conducted</li> <li>- Pts./decade -- number of points per decade</li> <li>- Num. of pts. -- the number of points over which sweep is conducted</li> </ul>  |
| Center/Span<br>Center, Span,<br>Step-size,<br>Pts./decade,<br>Num. of pts.   | SS_Center<br>SS_Span<br>SS_Step<br>SS_Dec<br>SS_Lin | Select the Center/Span option to sweep based on center and span, step-size or pts./decade, and number of points. Linear sweep uses Step-size; Log sweep uses Pts./decade. <ul style="list-style-type: none"> <li>- Center -- the center point of a sweep</li> <li>- Span -- the span of a sweep</li> <li>- Step -- size-the increments at which the sweep is conducted</li> <li>- Pts./decade -- number of points per decade</li> <li>- Num. of pts. -- the number of points over which sweep is conducted</li> </ul> |
| Note: Changes to any of the Start, Stop, etc. fields causes the remaining fields to be recalculated automatically. |   |   |
| Use sweep plan   | SS_Plan   | Enables use of an existing sweep plan component (SweepPlan). Select this option and enter the name of the plan or select it from the drop-down list.  |
| Use all small-signal frequencies   | UseAllSS_Freqs                                      | Solves for all small-signal mixer frequencies in both sidebands. This default option requires more memory and simulation time, but is required for the most accurate simulations.   |
| Merge small- and large-signal frequencies  | MergeSS_Freqs                                       | By default, the simulator reports only the small-signal upper and lower sideband frequencies in a mixer or oscillator simulation. Selecting this option causes the fundamental frequencies to be restored to the dataset, and merges them sequentially.   |

## Defining Simulation Parameters

The Params tab enables you to define these basic simulation parameters:

- Enabling the Budget simulation.
- Specifying the amount of device operating-point information to save.
- Specifying the FFT oversampling ratio.

The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Parameters

| Setup Dialog Name            | Parameter Name | Description   |
|------------------------------|----------------|---|
| Device operating point level | DevOpPtLevel   | Enables you to save all the device operating-point information to the dataset. In ADS, if this simulation performs more than one HB analysis (from multiple HB controllers), the device operating point data for all HB analyses will be saved, not just the last one. Default setting is None.   |
| None                         | None           | No information is saved.  |
| Brief                        | Brief          | Saves device currents, power, and some linearized device parameters.  |
| Detailed                     | Detailed       | Saves the operating point values which include the device's currents, power, voltages, and linearized device parameters.  |
| FFT                          |                |   |
| Fundamental Oversample       | FundOversample | Sets the FFT oversampling ratio. Higher levels increase the accuracy of the solution by reducing the FFT aliasing error and improving convergence. Memory and speed are affected less when the direct harmonic balance method is used than when the Krylov option is used. Increasing the Oversample can help convergence by ensuring that rapid transitions and sharp features in waveforms are more precisely sampled. This does not increase the problem size, but does increase the number of device evaluations. The computational complexity of the direct Harmonic Balance solver (determined by the Order and size of circuit) is largely not affected. As a rule of thumb, try to set the Oversample to 2, 4, etc. For multi-tone Harmonic Balance, the number of samples is equal to the product of the sample sizes of the fundamentals. |
| More...                      | Oversample[n]  | Displays a small dialog box. To increase simulation accuracy, enter in the field an integer representing a ratio by which the simulator will oversample each fundamental.   |
| Budget                       |                |   |
| Perform Budget simulation    | OutputBudgetIV | Enables Budget simulation, which reports current and voltage data at the pins of devices following a simulation. Current into the <i>n</i> th terminal of a device is identified as <i>...device_name.tn.i</i> . Voltage at the <i>n</i> th terminal of a device is identified as <i>...device_name.tn.v</i> .  |

## Selecting a Harmonic Balance Solver Technique

The *Solver* tab enables you to select a Direct or Krylov solver, or to allow the simulator to assign one automatically. The automatic selection is the default and recommended choice, since it allows the simulator to choose the most effective solver for each particular circuit. Newton's method needs to solve a sequence of linear problems. If Newton's method doesn't converge or the convergence rate is too slow the Direct method will use arc-length continuation, while the Krylov method will use source stepping. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Solver Parameters

| Setup Dialog Name                | Parameter Name     | Description  |
|----------------------------------|--------------------|--|
| Convergence                      |                    |  |
| Convergence Mode                 | ConvMode           |  |
| Auto (Preferred)                 | =Auto (Preferred)  | This is the default mode setting. It is both fast and robust, combining capabilities of the Basic and Advanced modes. This mode will automatically activate advanced features to achieve convergence. The Auto mode also allows for convergence at looser tolerances if the simulation does not meet the default tolerances. A warning message is given in the status window when this occurs, and it includes the tolerance level up to which convergence was achieved. |
| Advanced (Robust)                | =Advanced (Robust) | Enables an advanced Newton solver. This mode is extremely robust, and ensures maximal KCL residual reduction at each iteration. It is recommended that the maximum number of iterations ( <i>MaxIters</i> ) be increased to the 50-100 range when this mode is selected.   |
| Basic (Fast)                     | =Basic (Fast)      | Enables the basic Newton solver. It is fast and performs well for most circuits. For highly nonlinear circuits the basic mode may have difficulties converging. It is then recommended to switch to the Advanced convergence mode.   |
| Max. iterations                  | MaxIters           | The maximum number of Newton iterations to be performed. The simulation will iterate until it converges, an error occurs, or this limit is reached. The default and recommended option is Robust. You can also specify the number manually by choosing the Custom option and entering an integer. The larger the number is, the more robust the simulation will be.  |
| Advanced Continuation Parameters |                    | Opens dialog to set the arc-length continuation parameters.  |
| Matrix Solver                    |                    |  |
| Solver Type                      | UseKrylov          |  |
| Auto Select                      | =auto              | This is the default mode setting. It is recommended because  |

|                            |                   |   |
|----------------------------|-------------------|---|
|                            |                   | optimal performance can be achieved for most circuits. It allows the simulator to choose which solver would be most effective for the active design.  |
| Direct                     | =no               | Best suited for smaller problems and faster. The computation time grows with the cube of the problem size and memory grows with the square of the problem size.<br>The parts (blocks) of the Jacobian are truncated to a specified threshold (bandwidth) by default (GuardThresh= 10 to the power 4). This bandwidth truncation speeds up the Jacobian factorization, but can lead to convergence problems as the Newton direction is not accurate. Try setting GuardThresh=0 (full bandwidth).   |
| Krylov                     | =yes              | Intended for larger problems, includes advanced preconditioning technology with an iterative linear solver. This method greatly reduces memory requirements in large harmonic balance problems, such as those encountered in RFICs or RF System simulations. The computation time grows slightly faster than linear with the number of samples (FFT size), and memory grows linearly with the number of harmonics.<br>This is an iterative linear solver that does not require explicit storage of Jacobian. The linear problem can be approximately solved in fewer iterations to a desired (loose) tolerance and the Newton direction is computed approximately. This can affect the Newton convergence properties, but not the accuracy of the final solution.<br>Krylov solver iterations are limited by the max number of iterations ( HB Solver tab, or KrylovMaxIters, default 150). Increase this limit if it is often reached.<br>The Krylov solver achieves full convergence if the linear system residual is smaller than the tight tolerance (KrylovTightTol, default 0.001). After KrylovLooseIters iterations (default 50), the solver uses KrylovLooseTol (default 0.1) to achieve partial convergence. The solver fails if residual reduction factor in two adjacent iterations is larger than KrylovConvRatio (default 0.9). |
| Matrix Re-use              | SamanskiiConstant | This parameter is for the Direct Solver only. It controls how frequently the Jacobian is constructed and factored rather than being reused. The default and recommended option is Fast. The user can specify the number by choosing Custom and entering 0, 1, or 2. The smaller the number is, the more robust the simulation will be.  |
| Krylov Restart Length      | GMRES_Restart     | This parameter determines the number of iterations after which the Krylov Solver is restarted. The larger this parameter, the more memory and CPU time will be required but the more robust the simulation will be as well.   |
| Advanced Krylov Parameters |                   | Opens dialog to set the Krylov solver's parameters.   |
| Memory Management          |                   |   |
| Matrix Bandwidth           | GuardThresh       | The Jacobian matrix from the direct solver within the Newton solver is a block matrix. A block matrix is a matrix whose elements are matrices and vectors. The blocks of the Jacobian are truncated to a specified threshold by default. The default threshold (bandwidth) is set by Guard Threshold, and its default option is Fast. The bandwidth truncation speeds up the Jacobian factorization and saves memory, but can lead to convergence problems due to an inaccurate Newton direction. In order to get   |

|                                    |                      |   |
|------------------------------------|----------------------|---|
|                                    |                      | the full bandwidth of the Jacobian blocks and improve the convergence, choose the Robust option.  |
| FFT Options                        | PackFFT              | Controls the frequency map packing for multitone Harmonic Balance. By default, when it is not explicitly set to <i>yes</i> or <i>no</i> by the user, the simulator enables it (sets it to <i>yes</i> ).   |
| Minimize memory & runtime          | PackFFT=yes          | Enables frequency map packing, which may improve the simulation speed and reduce memory consumption by using a smaller number of time samples (smaller FFTs), but at the potential loss of dynamic range and accuracy due to the aliased harmonics of the first fundamental now possibly landing on various mixing tones. |
| Minimize aliasing                  | PackFFT=no           | Disables frequency map packing to achieve most accurate results.  |
| Waveform Memory Reduction          |                      |   |
| Use Dynamic Waveform Recalculation | RecalculateWaveForms | Enables reuse of dynamic waveform memory instead of upfront storage on all waveforms. Small circuits might simulate a little slower, but not significantly.   |
| Use Compact Frequency Map          | UseCompactFreqMap    | Enables a spectral compression, typically requiring less memory for individual waveforms.   |

## Advanced Continuation Parameters

These parameters are for arc-length continuation. The arc-length continuation is an extremely robust algorithm. If it fails, try all other convergence remedies first before adjusting these arc-length parameters. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Solver Advanced Continuation Parameters

| Setup Dialog Name  | Parameter Name  | Description  |
|--------------------|-----------------|--|
| Arc Max Step       | ArcMaxStep      | Limits the maximum size of the arc-length step during arc-length continuation. In the arc-length continuation, the arc-length is increased in steps. The step size is calculated automatically for each problem. However if the ArcMaxStep is specified and is nonzero, it will define an upper-limit for the size of the arc-length step. The default is 0 which means there is no upper limit for the ArcMaxStep. Display and set this parameter directly on the schematic.  |
| Arc Level Max Step | ArcLevelMaxStep | Limits the maximum arc-length step size for source-level continuation. The default is 0 which means there is no limit for the ArcLevelMaxStep. Display and set this parameter directly on the schematic.   |
| Arc Min Value      | ArcMinValue     | Set relative to ArcMaxValue. ArcMinValue determines the lower limit that is allowed for the continuation parameter $p$ during the simulation. In the arc-length continuation, $p$ can trace a complicated manifold and its value can vary non-monotonically. ArcMinValue specifies a lower bound for $p$ such that if during the arc-length continuation, $p$ becomes smaller than ArcMinValue, the simulation is considered to have failed to converge. The default is $p_{min}-delta$ , where $delta$ is $p_{max}-p_{min}$ , $p_{min}$ is the lower end of the parameter sweep, and $p_{max}$ is the upper end of the parameter sweep. Display and set this parameter directly on the schematic. |
| Arc Max Value      | ArcMaxValue     | Set relative to ArcMinValue. ArcMaxValue determines the allowed upper limit of the continuation parameter $p$ during the simulation. In the arc-length continuation, $p$ can trace a complicated manifold and its value can vary non-monotonically. ArcMaxValue specifies an upper bound for $p$ such that if during the arc-length continuation, $p$ becomes greater than ArcMaxValue, the simulation is considered to have failed to converge. The default is $p_{max}+delta$ , where $delta$ is $p_{max}-p_{min}$ , $p_{min}$ is the lower end of the parameter sweep, and $p_{max}$ is the upper end of the parameter sweep. Display and set this parameter directly on the schematic.         |
| Max Step Ratio     | MaxStepRatio    | Controls the maximum number of continuation steps (default is 100). Display and set this parameter directly on the schematic.  |
| Max Shrinkage      | MaxShrinkage    | Controls the minimum size of the arc-length step (default is 1e-5). Display and set this parameter directly on the schematic.  |

## Advanced Krylov Parameters

These parameters allow detailed configuration for the Krylov solver. Default values are recommended for optimal performance. The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

### HB Simulation Advanced Krylov Parameters



| Setup Dialog Name      | Parameter Name      | Description   |
|------------------------|---------------------|---|
| Max Iterations         | KrylovMaxIters      | Maximum number of GMRES iterations allowed. It is used to interrupt an otherwise infinite, loop in the case of poor or no convergence. The default is intentionally set to a large value of 150 to accommodate even slowly convergent iterations. You can still increase this number in cases where poor convergence may be improved and you are willing to allow more time for it.   |
| Krylov Noise Tolerance | KrylovSS_Tol        | Sets the tolerance for the Krylov solver when that solver is used either for small-signal harmonic balance analysis or for nonlinear noise analysis. It needs to be tight, and the default value is 1e-6. Larger values may lead to less accurate results, while further tightening may require longer simulation times.  |
| Packing Threshold      | KrylovPackingThresh | Used with Matrix Packing. Packing Threshold sets the bandwidth threshold for the packing. The default value is 1e-8. Set this to a larger value to increase the memory reduction. Display and set this parameter directly on the schematic.   |
| Tight Tolerance        | KrylovTightTol      | The solver achieves full convergence if the Krylov solver residual is less than this tight tolerance setting (default=0.001). Display and set this parameter directly on the schematic.   |
| Loose Tolerance        | KrylovLooseTol      | After the number of iterations specified by the parameter Loose Iterations, the solver then uses Loose Tolerance (default=0.1) to achieve partial convergence. Display and set this parameter directly on the schematic.  |
| Loose Iterations       | KrylovLooseIters    | Sets the number of iterations allowed (default=50) to achieve convergence before using the Loose Tolerance value. When the number of Loose Iterations is reached, the solver then uses the Loose Tolerance value to achieve partial convergence. Display and set this parameter directly on the schematic.  |
| Matrix packing         | KrylovUsePacking    | Directs the solver to use the technique known as spectral packing, which reduces the memory needed for the Jacobian, typically by 60-80%. The penalty is a longer computation time if no swapping is required. By default, this feature is turned off. You should turn on for extremely large problems in which the available RAM would not be able to accommodate the Jacobian.  |
| Preconditioner         | KrylovPrec          | The Krylov solver requires a preconditioner for robust and efficient convergence. Preconditioners (matrices approximating the Jacobian) are used to speed up the Krylov solver's convergence. ADS uses GMRES, a robust and theoretically optimal Krylov solver that is memory intensive without a restart.  |
| DCP                    | =DCP                | (DC Preconditioner) is the default preconditioner, which is effective in most cases, but fails for some very strong nonlinear circuits. It uses a DC approximation on the entire circuit. Due to its block-diagonal nature, it can be factored once and applied inexpensively at each linear solve step. This preconditioner approximates the Jacobian by ignoring all but the DC Fourier coefficients (consists of the diagonal blocks of the Jacobian). Hidden DCP parameter is accessible only by using <code>Other=</code> :<br><ul style="list-style-type: none"> <li>- <code>PrecMatrixSolver</code> -- sets the matrix solver for the DCP preconditioner. Two solvers are available currently: the general solver that is preferable for smaller or typical-size circuits and the large-circuit solver that is preferable for larger circuits. Allowed values are:</li> <li>- <code>PrecMatrixSolver=0</code> (default, the simulator automatically chooses the solver)</li> <li>- <code>PrecMatrixSolver=1</code> (use the general solver)</li> <li>- <code>PrecMatrixSolver=2</code> (use the large-circuit solver)</li> </ul> |

|     |      |  |
|-----|------|--|
| BSP | =BSP | (Block Select Preconditioner) is recommended for instances when a Krylov HB simulation fails to converge using the DCP option. The BSP preconditioner is more robust than the DCP for highly nonlinear circuits. On those circuits that converge with DCP, the overhead that the BSP preconditioner introduces is small. On circuits that fail with the DCP, using the BSP option will often achieve convergence at the cost of additional memory usage. Hidden BSP parameter is accessible only by using <code>Other=</code> :<br>- <code>bspRHS_Thresh</code> -- activate BSP if Newton residual smaller than this threshold (default 0.05)  |
| SCP | =SCP | (Schur-Complement Preconditioner) is also intended for use with circuits that fail to converge with the DCP preconditioner. This is a robust choice for highly nonlinear circuits. It uses the DC approximation for most of the circuit similar to DCP. The most nonlinear parts of the circuit are excluded, and are instead factored with a specialized Krylov solver. The complex technology of the SCP preconditioner results in a memory usage overhead. This overhead is due to a construction of a knowledge base that enables the SCP to be much more efficient in the later phase of the harmonic balance solution process. Hidden SCP parameters are accessible only by using <code>Other=</code> :<br>- <code>ScpRhsThresh</code> activate SCP if Newton residual smaller than this threshold (default 0.05)<br>- <code>ScpRestart</code> inner SCP GMRES restart value (default 100)<br>- <code>ScpTol</code> inner SCP GMRES tolerance (default 0.001)<br>- <code>ScpStartIter</code> use SCP from this Newton iteration onward (default 0) |

## Backward Compatibility Exceptions

Beginning with ADS 2006A, the default value for the following parameter does not maintain backwards compatibility with ADS 2004A as shown in the following table. When a project created in 2004A is opened in 2005A or 2006A, this parameter will use the default value specified in 2006A regardless of the value specified in 2004A either by default or by the user.

| Parameter Name | Setup Dialog Name                       | Default Value |                |
|----------------|---|---------------|----------------|
|                |   | 2004A         | 2005A<br>2006A |
| Restart        | Regenerate Initial Guess for ParamSweep | yes           | no             |

## Theory of Operation

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 enables 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.

The Harmonic Balance solution is approximated by truncated Fourier series and this method is inherently incapable of representing transient behavior. The time-derivative can be computed exactly with boundary conditions,  $v(0)=v(t)$ , automatically satisfied for all iterates.

The truncated Fourier approximation + N circuit equations results in a residual function that is minimized.

$N \times M$  nonlinear algebraic equations are solved for the Fourier coefficients using Newton's method and the inner linear problem is solved by:

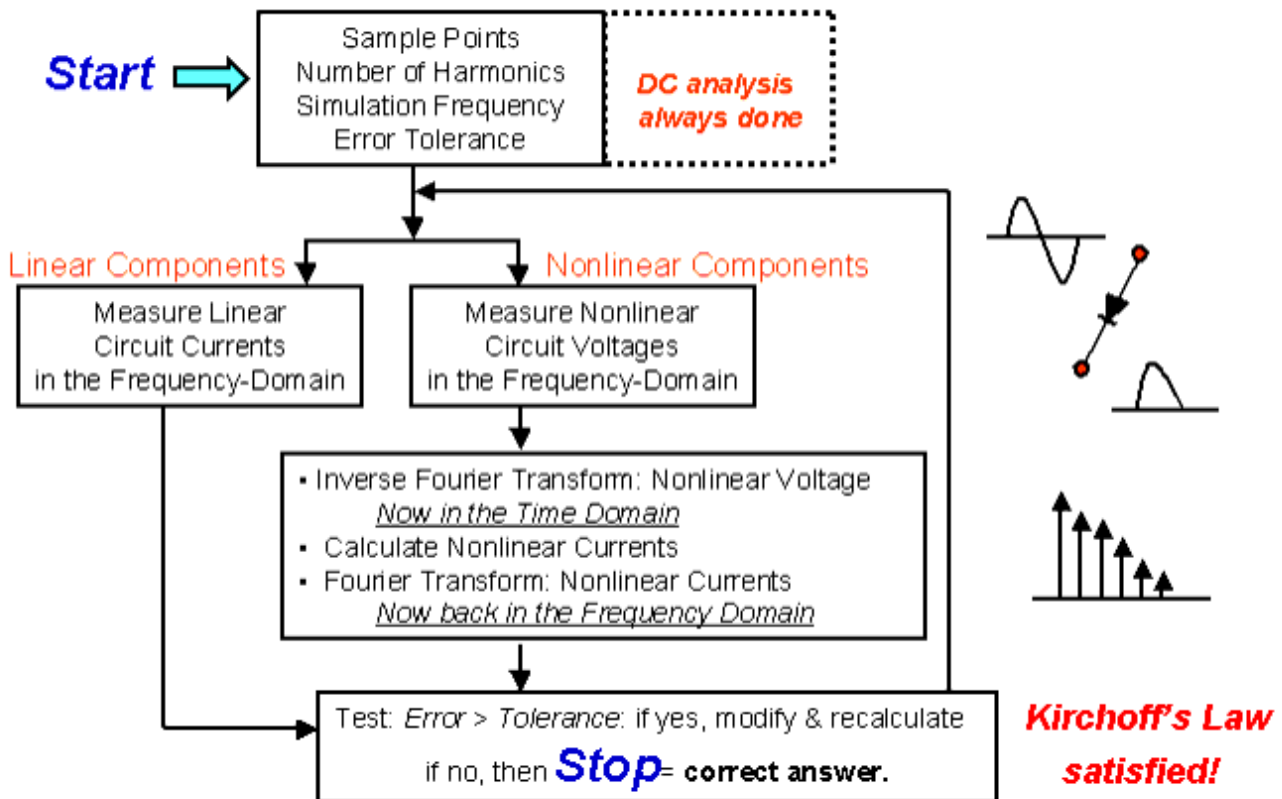
- Direct method (Gaussian elimination) for small problems.
- Krylov-subspace method (e.g. GMRES) for larger problems.

Nonlinear devices (transistors, diodes, etc.) in Harmonic Balance are evaluated (sampled) in the time-domain and converted to frequency-domain via the FFT.

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. The following flow chart presents a visual representation of the process:

## Harmonic Balance Simulation Flow Chart



## 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.
- Harmonic balance is faster at solving typical high-frequency problems that transient analysis can't solve accurately or can only do so at prohibitive costs. 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 Mixing Order

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

| Source | Frequency | Order |
|--------|-----------|-------|
| Fund 1 | 5 MHz     | 3     |
| Fund 2 | 7 MHz     | 3     |

If *Maximum mixing 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:

| Frequency | Combination               |
|-----------|---------------------------|
| 0 Hz      | DC term                   |
| 5 MHz     | Fund 1                    |
| 7 MHz     | Fund 2                    |
| 10 MHz    | Second harmonic of Fund 1 |
| 14 MHz    | Second harmonic of Fund 2 |
| 15 MHz    | Third harmonic of Fund 1  |
| 21 MHz    | Third harmonic of Fund 2  |

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

| Frequency | Combination     |
|-----------|-----------------|
| 2 MHz     | Fund 2 - Fund 1 |
| 12 MHz    | Fund 2 + Fund 1 |

If *Maximum mixing 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:

| Frequency | Combination                        |
|-----------|------------------------------------|
| 3 MHz     | Second harmonic of Fund 1 - Fund 2 |
| 9 MHz     | Second harmonic of Fund 2 - Fund 1 |
| 17 MHz    | Second harmonic of Fund 1 + Fund 2 |
| 19 MHz    | Second harmonic of Fund 2 + Fund 1 |

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.

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

| Source | Frequency | Order |
|--------|-----------|-------|
| Fund 1 | 10.00 GHz | 5     |
| Fund 2 | 10.95 GHz | 2     |
| Fund 3 | 11.05 GHz | 2     |

The frequencies that are used in the simulation and the combinations that produce them are listed in the following table.

**Frequency List (Maximum mixing order = 5)**

| Frequency | Combination  | Order          |
|-----------|--|----------------|
| 0.000 Hz  | DC term  | always present |
| 100.0 MHz | Fund 3 - Fund 2  | 2              |
| 200.0 MHz | Second harmonic of Fund 3 - Second harmonic of Fund 2          | 4              |
| 850.0 MHz | Second harmonic of Fund 2 - Fund 1 - Fund 3                    | 4              |
| 950.0 MHz | Fund 2 - Fund 1  | 2              |
| 1.050 GHz | Fund 3 - Fund 1  | 2              |
| 1.150 GHz | Second harmonic of Fund 3 - Fund 1 - Fund 2                    | 4              |
| 1.900 GHz | Second harmonic of Fund 2 - Second harmonic of Fund 1          | 4              |
| 2.000 GHz | Fund 2 + Fund 3 - Second harmonic of Fund 1                    | 4              |
| 2.100 GHz | Second harmonic of Fund 3 - Second harmonic of Fund 1          | 4              |
| 7.900 GHz | Third harmonic of Fund 1 - Second harmonic of Fund 3           | 5              |
| 8.000 GHz | Third harmonic of Fund 1 - Fund 2 - Fund 3                     | 5              |
| 8.100 GHz | Third harmonic of Fund 1 - Second harmonic of Fund 2           | 5              |
| 8.850 GHz | Second harmonic of Fund 1 + Fund 2 - Second harmonic of Fund 3 | 5              |
| 8.950 GHz | Second harmonic of Fund 3 - Second harmonic of Fund 1          | 4              |
| 9.050 GHz | Second harmonic of Fund 2 - Second harmonic of Fund 1          | 4              |
| 9.150 GHz | Second harmonic of Fund 1 + Fund 3 - Second harmonic of Fund 2 | 5              |
| 9.800 GHz | Fund 1 + Second harmonic of Fund 3 - Second harmonic of Fund 2 | 5              |
| 9.900 GHz | Fund 1 + Fund 2 - Fund 3                                       | 3              |
| 10.00 GHz | Fundamental 1  |                |
| 10.10 GHz | Fund 1 + Fund 3 - Fund 2                                       | 3              |
| 10.20 GHz | Fund 1 + Second harmonic of Fund 3 - Second harmonic of Fund 2 | 5              |
| 10.85 GHz | Second harmonic of Fund 2 - Fund 3                             | 3              |
| 10.95 GHz | Fundamental 2  |                |
| 11.05 GHz | Fundamental 3  |                |
| 11.15 GHz | Second harmonic of Fund 3 - Fund 2                             | 3              |
| 11.90 GHz | Second harmonic of Fund 2 - Fund 1                             | 3              |
| 12.00 GHz | Fund 2 + Fund 3 - Fund 1                                       | 3              |
| 12.10 GHz | Second harmonic of Fund 3 - Fund 1                             | 3              |
| 12.95 GHz | Second harmonic of Fund 2 + Fund 3 - Fund 1                    | 4              |
| 13.05 GHz | Fund 2 + Second harmonic of Fund 3 - Second harmonic of Fund 1 | 5              |
| 18.95 GHz | Third harmonic of Fund 1 - Fund 3                              | 4              |
| 19.05 GHz | Third harmonic of Fund 1 - Fund 2                              | 4              |
| 19.90 GHz | Second harmonic of Fund 1 + Fund 2 - Fund 3                    | 4              |

Advanced Design System 2011.01 - Harmonic Balance Simulation

|           |  |   |
|-----------|--|---|
| 20.00 GHz | Second harmonic of Fund 1                                      |   |
| 20.10 GHz | Second harmonic of Fund 1 + Fund 3 - Fund 2                    | 4 |
| 20.85 GHz | Fund 1 + Second harmonic of Fund 2 - Fund 3                    | 4 |
| 20.95 GHz | Fund 2 + Fund 1 - 2nd order                                    |   |
| 21.05 GHz | Fund 3 + Fund 1 - 2nd order                                    |   |
| 21.15 GHz | Fund 1 + Second harmonic of Fund 3 - Fund 2                    | 4 |
| 21.90 GHz | Second harmonic of Fund 2                                      |   |
| 22.00 GHz | Fund 2 + Fund 3 - 2nd order                                    |   |
| 22.10 GHz | Second harmonic of Fund 3                                      |   |
| 22.95 GHz | Fund 3 + Second harmonic of Fund 2 - Fund 1                    | 4 |
| 23.05 GHz | Fund 2 + Second harmonic of Fund 3 - Fund 1                    | 4 |
| 28.95 GHz | Fourth harmonic of Fund 1 - Fund 3                             | 5 |
| 29.05 GHz | Fourth harmonic of Fund 1 - Fund 2                             | 5 |
| 29.90 GHz | Third harmonic of Fund 1 + Fund 2 - Fund 3                     | 5 |
| 30.00 GHz | Third harmonic of Fund 1                                       |   |
| 30.10 GHz | Third harmonic of Fund 1 + Fund 3 - Fund 2                     | 5 |
| 30.85 GHz | Second harmonic of Fund 1 + Second harmonic of Fund 2 - Fund 3 | 5 |
| 30.95 GHz | Second harmonic of Fund 1 + Fund 2                             | 3 |
| 31.05 GHz | Second harmonic of Fund 1 + Fund 3                             | 3 |
| 31.15 GHz | Second harmonic of Fund 1 + Second harmonic of Fund 3 - Fund 2 | 5 |
| 31.90 GHz | Fund 1 + Second harmonic of Fund 2                             | 3 |
| 32.00 GHz | Fund 1 + Fund 2 + Fund 3                                       | 3 |
| 32.10 GHz | Fund 1 + Second harmonic of Fund 3                             | 3 |
| 32.95 GHz | Fund 3 + Second harmonic of Fund 2                             | 3 |
| 33.05 GHz | Fund 2 + Second harmonic of Fund 3                             | 3 |
| 34.00 GHz | Second harmonic of Fund 2 + Second harmonic of Fund 3 - Fund 1 | 5 |
| 40.00 GHz | Fourth harmonic of Fund 1                                      |   |
| 40.95 GHz | Third harmonic of Fund 1 + Fund 2                              | 4 |
| 41.05 GHz | Third harmonic of Fund 1 + Fund 3                              | 4 |
| 41.90 GHz | Second harmonic of Fund 1 + Second harmonic of Fund 2          | 4 |
| 42.00 GHz | Second harmonic of Fund 1 + Fund 2 + Fund 3                    | 4 |
| 42.10 GHz | Second harmonic of Fund 1 + Second harmonic of Fund 3          | 4 |
| 42.95 GHz | Fund 1 + Second harmonic of Fund 2 + Fund 3                    | 4 |
| 43.05 GHz | Fund 1 + Fund 2 + Second harmonic of Fund 3                    | 4 |
| 44.00 GHz | Second harmonic of Fund 2 + Second harmonic of Fund 3          | 4 |
| 50.00 GHz | Fifth harmonic of Fund 1                                       |   |
| 50.95 GHz | Fourth harmonic of Fund 1 + Fund 2                             | 5 |
| 51.05 GHz | Fourth harmonic of Fund 1 + Fund 3                             | 5 |
| 51.90 GHz | Third harmonic of Fund 1 + Second harmonic of Fund 2           | 5 |
| 52.00 GHz | Third harmonic of Fund 1 + Fund 2 + Fund 3                     | 5 |
| 52.10 GHz | Third harmonic of Fund 1 + Second harmonic of Fund 3           | 5 |
| 52.95 GHz | Second harmonic of Fund 1 + Second harmonic of Fund 2 + Fund 3 | 5 |
|           |  |   |

|           |  |   |
|-----------|--|---|
| 53.05 GHz | Second harmonic of Fund 1 + Fund 2 + Second harmonic of Fund 3 | 5 |
| 54.00 GHz | Fund 1 + Second harmonic of Fund 2 + Second harmonic of Fund 3 | 5 |

## 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.

Use the following guidelines when selecting a solver:

- **Auto Select**  
This option allows the simulator to choose the linear solver, which is recommended because the optimal choice can be made automatically for most circuits by this option. 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. Furthermore, when a solver is chosen by the simulator, parameters for that solver are also automatically optimized.
- **Direct Solver**  
The *Direct Solver* option is only recommended when the problem is small, and the simulator chooses the Krylov Solver yet convergence difficulty is encountered. 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. The simulator will usually choose the Direct Solver for small problems automatically with the *Auto Select* option selected.
- **Krylov Solver**  
The *Krylov Solver* option is only recommended when the simulator chooses the Direct Solver and runs out of memory or convergence difficulty is encountered, especially



when solving *large problems*. A large problem can roughly be described as one where memory usage exceeds 400 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. The simulator will usually choose the Krylov Solver for large problems automatically with the *Auto Select* option selected. Krylov is less robust than the Direct Solver method because it uses iterative algorithms to solve the matrix equations. However, it is much faster and requires much less memory than the Direct Solver for large problems.

In most cases, *Auto Select* will choose the most effective solver automatically. Choose *Direct Solver* or *Krylov Solver* only when the *Auto Select* option cannot deliver satisfactory performance for a certain circuit.

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 setup 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 `<workspace_name>.hbs`. The file will be saved in the *networks* folder of the workspace.
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 information on how to do this, refer to the topic *Reusing Simulation Solutions* (cktsim).

**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  $\text{Freq}[1] = 1\text{GHz}$  and  $\text{Freq}[2] = 1\text{kHz}$  would not be a good match for a simulation with  $\text{Freq}[1] = 1\text{kHz}$  and  $\text{Freq}[2] = 1\text{GHz}$ .

## 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](#) describes how to select an optimum number of harmonics in a simulation to ensure accuracy and minimize simulation time.
- [Reducing Simulation Time](#) describes other methods to help reduce simulation time.
- [Solving Convergence Problems](#) offers a variety of methods to try if your simulation is failing because it cannot converge.
- [Oversampling to Prevent Aliasing](#) describes how the Oversampling parameter can be set to prevent aliasing during a simulation.
- [Linearizing Nonlinear Devices](#) describes how making a circuit more linear can improve the chances of a successful simulation.

Also, refer to the troubleshooting sections in *DC Simulation* (cktsimdc) and *AC Simulation* (cktsimac).

## 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](#)).

- **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.
- Use the Status server window as the main tool in solving convergence problems (set StatusLevel=4). For each Newton iteration the L-1 norm of the residuals throughout the circuit is printed: a "\*" indicates a full Newton step (vs. a Samanskii step).
- Convergence criteria are controlled by *Voltage relative tolerance*, and *Current relative tolerance*. 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. To set these values, use the Convergence tab on the *Options* component.
- Newton convergence issues with Krylov methods (because linear problem solutions can only approximate) can be improved by using better preconditioners.
- For non-convergence due to tight tolerances, monitor the residuals in the Status Server window:
  - Increase I\_AbsTol if the circuit is converging to within a few pA but not quite to I\_AbsTol=1pA.
  - Increase I\_RelTol if the problem is with nodes associated with large currents.
  - Increase I\_AbsTol if the small current nodes are the issue.
  - Relax voltage tolerances for failure in the Newton update criterion.
- 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](#).
- 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.

**Note**

There are many references in the literature to SPICE MOSFET models. One is Antognetti and Mossobrio, *Semiconductor Device Modeling with SPICE*, Second Edition, McGraw-Hill, 1993.

- 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. For more information, see *Setting Convergence Options* (cktsim).
- 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](#) for instructions on how to minimize the number of harmonics.
- If the simulator cannot find a solution, it will attempt to do source stepping. This sets all large-signal AC sources to zero (leaving DC sources alone) and attempts to achieve convergence at a lower level. Starting from this solution point, the simulator will try to converge at the original level by repeatedly solving the circuit at incrementally increasing source levels, 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.
- The internal circuit simulator engine in ADS (ADSSim) runs from a netlist. ADS writes a netlist file (netlist.log) before invoking ADSSim. The order of the components and model definitions in the netlist determine the initial Jacobian matrix ordering. This matrix ordering can affect the efficiency of the Jacobian factorization and cause either a simulation slow down or non-convergence.
- For convergence problems due to errors in the component model equations (incorrect derivatives, etc.) make sure ancient Berkeley MOSFET Level 1, 2, 3 are not the culprit and that the latest model version is used (especially BSIM3 models). Model problems can cause the Newton residual to hit a threshold (greater than the convergence criteria tolerances) and stall the convergence process or even exhibit random jumps (sudden increase in value). Set the device's Xqc parameter to a nonzero value to allow the simulator to use a charge-based model for the gate capacitance. This often enables convergence, but at the cost of extracting an extra SPICE model parameter.

## Sweeps as Convergence Tools

Continuation methods provide a sequence of initial guesses that are sufficiently close to the solution to assure Newton's method convergence in Harmonic Balance. Sweeps can be used to formulate a specialized continuation method geared towards the particular circuit problem.

Try sweeping a parameter to correct a convergence problem. 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*, on the *Initial Guess* tab. Usually, a better initial guess at each step helps the simulator to converge.

The two main ways to perform sweeps are:

- HB sweep within the HB controller. This is preferred for most sweeps, except frequency.
- Parameter sweep using a separate sweep controller.

## Convergence and the Samanskii Steps

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.



### Note

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

## Convergence and Arc-Length Continuation

Arc-length continuation is an extremely robust algorithm. If it fails, try all other

convergence remedies first before adjusting arc-length parameters:

- MaxStepRatio controls the maximum number of continuation steps (default 100).
- MaxShrinkage controls the minimum size of the arc-length step (default 1e-5).
- ArcMaxStep limits the maximum size of the arc-length step (default is 0, i.e. no limiting).
- ArcMinValue & ArcMaxValue define the allowed range for the variation of the continuation parameter.

## Oversampling to Prevent Aliasing

Oversampling increases the accuracy of the solution by reducing the FFT aliasing error, and improves convergence for circuits with many nonlinear devices.

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 *Fundamental Oversample*, on the Params tab). The relationship between *Order*, *Fundamental Oversample*, and the number of samples for a single tone simulation is given by:

Number of Samples =  $2 * (\text{Order}[1] + 1) * \text{Oversample}$ , rounded up to the nearest power of two.

For the same value of *Order*, different values of *Fundamental Oversample* can yield the same number of samples. Also, oversampling can occur even with a *Fundamental Oversample* value of 1, because the system rounds the FFT size up to the next power of two. This can be seen from the following two tables:

| Order | Fundamental Oversample | Number of Samples |
|-------|------------------------|-------------------|
| 7     | 1                      | 16                |
| 7     | 2                      | 32                |
| 7     | 3                      | 32                |
| 7     | 4                      | 64                |
| 7     | 5                      | 128               |
| 7     | 6                      | 128               |

| Order | Fundamental Oversample | Number of Samples |
|-------|------------------------|-------------------|
| 8     | 1                      | 32                |
| 8     | 2                      | 64                |
| 8     | 3                      | 64                |
| 8     | 4                      | 128               |
| 8     | 5                      | 128               |
| 8     | 6                      | 128               |

In a multitone HB simulation, it is possible to set the oversample for each tone. To do this, click *More* next to the *Fundamental Oversample* parameter. The Oversample Options dialog box appears enabling you to enter the Oversample values for each fundamental in the multitone simulation. For a multitone simulation, the number of samples is determined in two parts. First, the same method used for the single tone case applies to the first tone; that is,  $2 * (\text{Order}[1] + 1) * \text{Oversample}$ , rounded up to the nearest power of two. Second, for all successive tones ( $\text{Order}[i]$ ,  $i > 1$ ), the number of samples is  $(2 * \text{Order}[i] + 1) * \text{Oversample}$ , rounded up to the nearest power of two. The final resultant number of samples is the product of the number of samples from each part.

| Order[1] | Order[2] | Fundamental Oversample | Number of Samples |
|----------|----------|------------------------|-------------------|
| 3        | 4        | 1                      | 128               |
| 3        | 4        | 2                      | 512               |
| 3        | 4        | 3                      | 1024              |
| 3        | 4        | 4                      | 2048              |
| 7        | 3        | 1                      | 128               |
| 7        | 3        | 2                      | 512               |
| 7        | 3        | 3                      | 2048              |
| 7        | 3        | 4                      | 2048              |

In the case of having different sample values, the same rules apply for determining the total number of samples. The value for *Oversample[1]* applies only to *Order[1]*, and *Oversample[2]* applies only to *Order[2]*. If a value is given for *Fundamental Oversample* and for *Oversample[1]*, the value of *Oversample[1]* will be used.

| Order[1] | Order[2] | Oversample[1] | Oversample[2] | Number of Samples |
|----------|----------|---------------|---------------|-------------------|
| 5        | 3        | 1             | 4             | 512               |
| 5        | 3        | 2             | 3             | 1024              |
| 7        | 4        | 3             | 2             | 2048              |
| 7        | 4        | 4             | 1             | 1024              |

While oversampling does not increase the number of harmonics, it does increase the size of the FFT used in HB. This means that the HB simulation run time using the direct solver (which is determined by the Order and the circuit size) is not largely affected when the *Fundamental Oversample* is increased. However, an HB simulation run time using the Krylov solver will be slower since this solver's computational complexity depends on the size of the FFT.

Note that *MaxOrder* does not enter any of the calculations for the number of samples.



## 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.

# 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 tab 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 *Harmonic Balance Basics* (cktsimhb), before continuing with this topic.

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

- [Performing a Nonlinear Noise Simulation](#) describes the minimum setup for calculating noise.
- [Performing a Noise Simulation with NoiseCons](#) describes how to set up NoiseCons and include them in the simulation.
- [Nonlinear Noise Simulation Description](#) is an overview on how nonlinear noise is calculated in a simulation.
- [NoiseCon Component Description](#) provides more details about the NoiseCon component.
- [NoiseCon Component](#) provides details on the tabs and fields for the Noise component.
- *Harmonic Balance for Oscillator Simulation* (cktsimhb), includes information on setting up noise simulations for oscillators.
- *Harmonic Balance for Mixers* (cktsimhb), includes information on setting up 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. Select the *Noise* tab and enable the *Nonlinear noise* checkbox.
- Click the *Noise(1)* button:
  - 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.
- Click the *Noise (2)* button:
  - 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* tab 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 Simulation Description](#).

## 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 Component Description](#).

## Nonlinear Noise Simulation Description

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\_wrk/NoiseFloor*.

If the noise figure is needed, add pins 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 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 *Sopt*, *Rn*, or *NFmin* 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 Component Description

The NoiseCon nonlinear noise controller allows more flexibility in the simulation of noise than can be specified using the Noise(1) and Noise(2) dialog boxes under the *Noise* tab. 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) dialog boxes 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).

In the dataset, noise results from the Noise(1) dialog box 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.

## NoiseCon Component

Following are details on the tabs and data fields for the NoiseCon component, which is available from the Simulation - HB library.

Two common tabs for controller components include the following:

- **Output** – Selectively save simulation data to a dataset. For details, refer to the topic *Selectively Saving and Controlling Simulation Data* (cktsim).
- **Display** – Control the visibility of simulation parameters on the Schematic. For details, refer to the topic *Displaying Simulation Parameters on the Schematic* (cktsim).

## Setting Up NoiseCon Frequency

The frequency at which noise should be computed is specified on the *Freq* tab. If a phase noise simulation is to be performed, as specified on the *PhaseNoise* tab, the interpretation of these noise frequencies is different. Instead of computing noise directly at these frequencies, they are interpreted as offset frequencies from a large signal carrier which is specified on the *PhaseNoise* tab. In the following table, names used in netlists and ADS schematics appear under *Parameter Name*.

### NoiseCon Frequency Options

| Setup Dialog Name  | Parameter Name   | Description  |
|--|--|--|
| Noise frequency  |  |  |
| Sweep Type   |  |  |
| Single point   | FreqForNoise   | Enables simulation at a single frequency point. Specify the desired value in the Parameter field.  |
| Linear   |  | Enables sweeping a range of values based on a linear increment. Click the Start/Stop option to select start and stop values for the sweep.   |
| Log  |  | Enables sweeping a range of values based on a logarithmic increment. Click the Center/Span option to select a center value and a span of the sweep.  |
| Start/Stop<br>Start, Stop,<br>Step-size,<br>Num. of pts.   | NLNoiseStart<br>NLNoiseStop<br>NLNoiseStep<br>NLNoiseLin | Select the Start/Stop option to sweep based on start, stop, step-size and number of points.<br>- Start -- the start point of a sweep<br>- Stop -- the stop point of a sweep<br>- Step -- size-the increments at which the sweep is conducted<br>- Num. of pts. -- the number of points over which sweep is conducted |
| Center/Span<br>Center, Span,<br>Pts./decade,<br>Num. of pts.   | NLNoiseCenter<br>NLNoiseSpan<br>NLNoiseDec<br>NLNoiseLin | Select the Center/Span option to sweep based on center and span.<br>- Center -- the center point of a sweep<br>- Span -- the span of a sweep<br>- Pts./decade -- number of points per decade<br>- Num. of pts. -- the number of points over which sweep is conducted   |
| Note: Changes to any of the Start, Stop, etc. fields causes the remaining fields to be recalculated automatically. |  |  |
| Use sweep plan   | NoiseFreqPlan  | Enables use of an existing sweep plan component (SweepPlan). Select this option and enter the name of the plan or select it from the drop-down list.   |

## Setting Up NoiseCon Nodes

The nodes at which noise voltages and branches through which noise currents are to be calculated are specified on this tab. Unlike the main harmonic balance simulation, noise is only computed at those nodes that the user specifies, not at all named nodes. If only the single sideband noise figure is to be computed, then no information need be entered here. In the following table, names used in netlists and ADS schematics appear under *Parameter Name*.

### NoiseCon Node Options

| Setup Dialog Name                         | Parameter Name   | Description  |
|---|------------------|--|
| Nodes for Noise Parameter Calculation     |                  | Use this area to select nodes at which you want linear noise data to be reported. Noise voltages and currents are reported in rms units. Note: The fewer the number of nodes requested, the quicker the simulation and the less memory required.   |
| Pos Node                                  |                  | Selects the named node(s) for the simulator to consider.   |
| Neg Node                                  |                  | Allows differential noise measurements to be set up. This dialog allows you to specify a differential noise measurement between two nodes. Unlike a harmonic balance simulation, the noise voltage between two nodes can't be computed by first computing the noise voltage separately at each node and subtracting the results, due to possible correlation between the noise at each node. Using a differential noise measurement here the correlation effects will be considered. To request a differential noise measurement, enter the first node as Pos Node and the second node as Neg Node. The result in the dataset will have a name of the form HB1.NC1.posnode_minus_negnode. If a differential noise measurement is not required, leave the Neg Node field blank. Differential measurements are supported only for node voltages, not branch current. |
| Select                                    |                  | <p>Holds the names of the nodes the simulator will consider.</p> <ul style="list-style-type: none"> <li>- Add -- Adds a named node.</li> <li>- Cut -- Deletes a named node.</li> <li>- Paste -- Takes a named node that has been cut and places it in a different order in the Select window.</li> </ul>   |
| Include port noise in node noise voltages | IncludePortNoise | Causes port noise to be included in noise currents and voltages. Ports must be placed and defined. If <i>Include port noise</i> is selected, the thermal noise generated by the source resistance in frequency domain power sources and Term elements are included in the simulation results for noise voltages and currents. This does not affect the computation of noise figure.  |

## Setting Up the NoiseCon Misc. Tab

The NoiseCon noise tab includes an assortment of options. In the following table, names used in netlists and ADS schematics appear under *Parameter Name*.

### NoiseCon Misc. Options



| Setup Dialog Name  | Parameter Name  | Description  |
|--------------------|-----------------|--|
| Input frequency    | InputFreq       | <p>ADS calculates the single-sideband definition of noise figure, therefore the desired input sideband frequency must be specified using this parameter. The "InputFreq" should identify which input frequency will mix to the noise frequency of interest.</p> <p>In the case of mixers, Input frequency is typically determined by an equation that involves the local oscillator (LO) frequency and the noise frequency (noisefreq). For example, consider a down-conversion mixer where <math>IFfreq = RFreq - LOfreq</math>. In this case, the image frequency is <math>IMAGEfreq = LOfreq - IFfreq</math> and any noise at this image frequency will translate to the IF port. To include the noise that comes from the RF signal and not from its image, you can set <math>InputFreq = LOfreq + noisefreq</math>. Note that "noisefreq" is a reserved keyword that ADS recognizes and replaces with the actual noise frequency as being defined in the "Noise Frequency" section of the controller.</p> <p>Note that the "InputFreq" parameters do not need to be specified if only the output noise voltage is desired (that is, if no noise figure is computed). Also, the "Input frequency" parameter needs to be specified only for frequency conversion devices. For other cases, you can leave it unchanged so that it remains at its default value which is "noisefreq".</p> |
| Noise input port   | NoiseInputPort  | Number of the source port at which noise is injected. This is commonly the RF port. Although any valid port number can be used, the input port number is frequently defined as Num=1. If noise figure is not required, this field should be blank.   |
| Noise output port  | NoiseOutputPort | Number of the Term component at which noise is retrieved. This is commonly the IF port. Although any valid port number can be used, the input port number is frequently defined as Num=2. If noise figure is not required, this field should be blank.   |
| Noise contributors |                 | Use this area to sort the noise contributors list and to select a threshold below which noise contributors will not be reported. A list shows how each component contributes to noise at a specific node. The noise contributor data are always in units of V/sqrt(Hz) for noise voltages, and A/sqrt(Hz) for noise currents; they do not scale with noise bandwidth.  |
| Mode               | SortNoise       | <p>Off -- causes no individual noise contributors (nodes) to be selected. The result is simply a value for total noise at the node.</p> <p>Sort by value -- sorts individual noise contributors, from largest to smallest, that exceed a user-defined threshold (see below). The subcomponents of the nonlinear devices that generate noise (such as Rb, Rc, Re, Ib, and Ic in a BJT) are listed separately, as well as the total noise from the device.</p> <p>Sort by name -- causes individual noise contributors to be identified and sorts them alphabetically. The subcomponents of the nonlinear devices that generate noise (such as Rb, Rc, Re, Ib, and Ic in a BJT) are listed separately, as well as the total noise from the device.</p> <p>Sort by value with no device details -- sorts individual noise contributors, from largest to smallest, that exceed a user-defined threshold (see below). Unlike Sort by value, only the total noise from nonlinear devices is listed without any subcomponent details.</p> <p>Sort by name with no device details -- causes individual noise contributors to be identified and sorts them alphabetically. Unlike Sort by name, only the total noise from nonlinear devices is listed without subcomponent details.</p>   |
| Dynamic range to   | NoiseThresh     | A threshold below the total noise, in dB, that determines what noise contributors are reported. All noise contributors more than this threshold  |

|                                     |                   |  |
|-------------------------------------|-------------------|--|
| display                             |                   | will be reported. For example, assuming that the total noise voltage is 10 nV, a setting of 40 dB (a good typical value) ensures that all noise contributors up to 40 dB below 10 nV (that is, above 0.1 nV) are reported. The default of 0 dB causes all noise contributors to be reported. This parameter is used only with <i>Sort by value</i> and <i>Sort by value with no device details</i> .   |
| Calculate noisy two-port parameters | NoiseTwoPort      | Causes an S-parameter simulation to be performed. Ports must be placed and defined. The Noise input port parameter should be set equal to the port number specified by the Num parameter on the input source, and the Noise output parameter to the number of the output Term (termination) component to Num=2. The following two-port parameters (dataset variables) are then returned and can be plotted:<br>NFmin -- minimum noise figure of a two-port circuit. It is equal to the noise figure when the optimum source admittance is connected to the circuit. (Its default unit is dB).<br>Sopt -- the optimum source match for a two-port circuit. It is the reflection coefficient (looking into the source) that gives the minimum noise figure.<br>Rn -- the effective noise resistance in ohms (unnormalized) of a two-port circuit. Effective noise resistance can be used to plot noise-figure circles or related quantities. This parameter determines how rapidly the minimum noise figure deteriorates when the source impedance is not at its optimum value.<br>Icor -- the noise current correlation matrix, in units of Amperes squared. It describes the short circuit noise currents squared at each port, and the correlation between noise currents at different ports. These expressions for noise simulation can be manipulated in equations. |
| Bandwidth                           | BandwidthForNoise | Bandwidth for spectral noise simulation. 1 Hz is the recommended bandwidth for measurements of spectral noise power. The noise contributor data do not scale with noise bandwidth.   |

## Setting Up NoiseCon PhaseNoise

This tab is used to specify that a phase noise simulation should be performed instead of a standard noise simulation. The default is *None (mixer)*, a standard noise simulation where noise voltages are computed at the requested nodes at the noise frequency specified in the *Freq* tab. When a *NoiseCon* is used, phase noise can be simulated for any type of circuit, not just an oscillator. For example, this makes it possible to simulate the added phase noise due to an amplifier. Phase noise can also be computed around any large signal carrier frequency, not just the fundamental frequency of an oscillator. This makes it possible to simulate the phase noise after an oscillator signal passes through a frequency multiplier or mixer. In the following table, names used in netlists and ADS schematics appear under *Parameter Name*.

### NoiseCon PhaseNoise Options

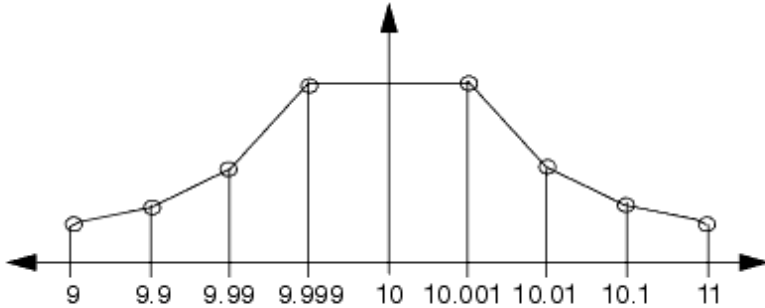
| Setup Dialog Name           | Parameter Name           | Description   |
|-----------------------------|--------------------------|---|
| Phase Noise Type            | PhaseNoise               |   |
| None (mixer)                | None (mixer)             | Noise voltages are computed at the nodes specified at the frequencies specified on the Freq tab. Noise results in the dataset are of the form HB1.NC1.vout.noise. There is no concept of a carrier frequency.   |
| Phase noise spectrum        | Phase noise spectrum     | Single sideband phase noise is computed at the nodes specified, treating the frequency information on the Freq tab as an offset frequency from the large signal carrier. See below on how to specify the large signal carrier frequency. Noise results in the dataset are of the form HB1.NC1.vout.pnmx.  |
| Relative vnoise spectrum    | Relative vnoise 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 present as a relative noise spectrum around zero, with the results extending from minus the largest offset frequency to plus the largest offset frequency. Noise results in the dataset are of the form HB1.NC1.vout.noise.   |
| Absolute vnoise spectrum    | Absolute vnoise spectrum | This is similar to the <i>Relative vnoise spectrum</i> , except the resulting spectrum is presented as an absolute spectrum around the large signal carrier frequency, extending from the carrier frequency minus the largest offset frequency to the carrier frequency plus the largest offset frequency. Noise results in the dataset are of the form HB1.NC1.vout.noise.   |
| Integrate over bandwidth    | Integrate over bandwidth | <p>This type of simulation integrates the noise around a large signal carrier. The user specifies the carrier on the Phase Noise tab, specifying either the Frequency in Hertz or the Carrier Mixing Indices, which specify the frequency as (index1*_freq1 + index2*_freq2 + *). The noise is integrated over a bandwidth specified by the Bandwidth parameter on the Misc tab, from carrier-bandwidth/2 to carrier+bandwidth/2 as shown in the equation below this table.</p> <p>A trapezoidal integration is performed (which takes into account the usual <math>1/f^{\alpha}</math> behavior of noise), using data at the frequencies specified on the Freq tab. The specified frequencies are actually offset frequencies on either side of the carrier. The largest offset frequency should be at least half of the noise bandwidth to avoid extrapolation of the results during integration. Noise results in the dataset are of the form HB1.NC1.vout.noise.</p> <p>The picture below this table shows an example of how the sample points for integration are chosen. Assume the carrier frequency is 10 MHz and the frequency sweep is set up to go from 1 kHz to 1 MHz in a logarithmic sweep with 1 point per decade, with a bandwidth of 2 MHz. This integration can be performed for any type of circuit in which the total noise integrated over some bandwidth is required. It works for mixers as well as oscillators. If the mixer noise figure is computed, the integrated noise voltage will be used instead of the spot noise voltage.</p> |
| Specify phase noise carrier |                          | The frequency of the large signal carrier used in all but the normal noise simulation are specified at the bottom of this tab. Any large signal carrier from the harmonic balance simulation may serve as the carrier frequency for phase noise simulation. There are two ways to specify the carrier frequency: <ul style="list-style-type: none"> <li>- Carrier Frequency</li> <li>- Carrier Mixing Indices</li> </ul> See following items.   |
| Frequency                   | CarrierFreq              | The frequency value may be specified directly by selecting the Frequency button and entering the frequency in the box below it. Either a number or a variable name may be entered. Since the frequency may not be precisely known in an oscillator, the simulator searches for the frequency closest to   |

|                        |                 |  |
|------------------------|-----------------|--|
|                        |                 | the user specified frequency. If the difference between the user specified frequency and the actual large signal frequency exceeds 10%, a warning message will be issued. A carrier frequency of zero cannot be entered directly as zero due to a limitation in the simulator; enter a small value such as 1 Hz instead. |
| Carrier mixing indices | CarrierIndex[n] | Specify these indices. For example, the lower sideband mixing term in a mixer would be entered as 1 and -1. The indices are listed in sequential order by carrier.   |

For the parameter *Integrate over bandwidth*, described above, the following equation shows noise is integrated over a bandwidth specified by the *Bandwidth* parameter on the Misc tab, from carrier-bandwidth/2 to carrier+bandwidth/2:

$$\overline{v_n} = \sqrt{\int_{f_{Carrier}-BW/2}^{f_{Carrier}+BW/2} v_n^2(f) df}$$

The following figure shows an example of how the sample points for integration are chosen:



# Harmonic Balance for Oscillator Simulation

The *HarmonicBalance* 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 *Harmonic Balance Basics* (cktsimhb) before continuing with this topic.

For basic information about using harmonic balance for oscillator and oscillator noise simulation, see the following topics:

- [Performing an Oscillator Simulation](#) has the minimum setup requirements for simulating an oscillator.
- [Performing an Oscillator Noise Simulation](#) focuses on noise simulations for oscillators.

For examples on performing harmonic balance simulations for oscillators and oscillator noise, see the following section:

- [Examples of Oscillator Simulations](#) 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, loop gain, and oscillator noise.

For detailed information on harmonic balance for oscillator and oscillator noise simulation, see the following topics:

- [Oscillator Simulation Description](#) is a brief description of the oscillator simulation process.
- [Phase Noise Simulation Description](#) describes the elements of phase noise and the calculations made during a phase noise analysis.
- [Troubleshooting a Simulation](#) offers suggestions on how to improve a simulation.
- [Simulation Techniques for Recalcitrant Oscillators](#) offers two techniques to solve the occasional problem with an oscillator that converges in a time-domain simulation, yet the harmonic balance oscillator algorithm is unable to find the solution.

## Performing an Oscillator Simulation

The main goal of simulating an oscillator using harmonic balance is to find both the oscillation frequency and the output spectrum of the oscillator. There are two methods that can be used to simulate an oscillator using harmonic balance. The first method

requires the insertion of an OscPort or OscPort2 into the feedback loop of the oscillator. The second, and preferred, method, commonly referred to as OscProbe, involves the identification of nodes in the oscillator's feedback loop.

## Using OscPort

The first oscillator simulation method requires that an OscPort or OscPort2 be inserted into the oscillator.

- *OscPort* is a component that is inserted into the feedback loop of a single-ended oscillator.
- *OscPort2* is a component that is inserted into the feedback loop of a differential (balanced) oscillator.

These components are inserted either in the feedback loop of the oscillator, or between the parts of the circuits that have negative resistance and the resonator. During harmonic balance oscillator analysis, the OscPort or OscPort2 is used by the simulator to monitor the loop gain of the oscillator and adjust the amplitude and frequency of oscillation.

## Specifying Oscillator Nodes

The second oscillator simulation method involves the identification of nodes in the oscillator's feedback loop. This method is often referred to as OscProbe, or Oscillator Probe. Unlike OscPort or OscPort2, OscProbe is not a component, but a preferred method for preparing an oscillator simulation.

For single-ended oscillators, one node in the feedback loop is identified; for differential oscillators, two nodes in the feedback loop are identified. These nodes should be either at the input or output of the active device of the oscillator, or in the oscillator's resonant tank. Nodes in the bias circuitry or an output buffer amplifier should not be used. The simulator uses these nodes to inject a voltage into the oscillator and adjusts the voltage and frequency so that no current flows from the injected source to these nodes.

## Setting Up an Oscillator Analysis

Set up a harmonic balance simulation then enable the Oscillator analysis mode. Select a *Method* to use either an Oscport or to specify oscillator nodes (OscProbe). If you use an Oscport, there is no further information to be specified for this option. If you specify oscillator nodes, then more information must be specified:

- Enter a node name for *Node Plus* for all oscillators.
- Enter a node name for *Node Minus* only for differential oscillators; leave this blank for single-ended oscillators.

Set the fundamental frequency on the Freq tab. Set the Fundamental Frequencies by entering the oscillation *Frequency*. Enter the number of harmonics of interest in the *Order* field.

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

The use of an OscPort in ADS is illustrated in the example [Calculating Large-Signal, Steady-State Oscillation Conditions](#). The use of an OscPort2 is illustrated in the example [Using OscPort2 for Oscillator Analysis](#).

## Other Oscillator Analyses

For a successful analysis, make sure your active device has sufficient feedback gain to permit oscillation and your resonator exhibits resonance. To check this, run an S-parameter simulation on the circuit before attempting the harmonic balance simulation. A brief overview of how to set this up is in the example [Finding the Frequency of Oscillation](#). For more details on how to set up the S-parameter simulations, refer to *S-Parameter Simulation* (cktsimsp).

There are two ways that the small-signal oscillator loop gain can be simulated:

- *OscTest* calculates loop gain for single-ended oscillators. Use it in circuits where you would use the OscPort. This component has a simulation controller already included in it, so you should not set up a harmonic balance simulation. This component is on the Probe Components palette in ADS. This is illustrated in an ADS schematic in the example [Calculating Oscillator Loop Gain](#).
- *OscPort2* can also be used to calculate the loop gain for differential (balanced) oscillators. The OscPort2 has a mode parameter that should be changed from its default setting of *Automatic* (for harmonic balance oscillator analysis) to *Small Signal Loop Gain* mode. This is illustrated in an ADS schematic in the example [Calculating Small-Signal Loop Gain](#).

The placement and setup of the OscTest, OscPort, and OscPort2 components vary. For instructions on how to do this, place one component 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 OscTest, OscPort, or OscPort2 in a circuit.

You can obtain certain oscillator characteristics using simulators other than the harmonic balance simulator:

- A DC simulation can be run separately to find the circuit's DC operating point.
- An S-parameter simulation can be used to find  $S_{21}$  (small-signal oscillator gain), and  $f_0$  (approximate frequency of oscillation).
- Transient analysis or circuit envelope simulation can be used to simulate the

oscillation startup.

## Performing an Oscillator Noise Simulation

This section 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, see [Performing an Oscillator Simulation](#) before continuing with this section.

See the following topics for details on oscillator noise simulation:

- [Simulating Phase Noise Using OscPort](#) describes how to set up a phase noise simulation.
- [Simulating Phase Noise with NoiseCons](#) 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 Simulation Description](#) describes the elements of phase noise, and the calculations made during a phase noise analysis.

## Examples of Oscillator Simulations

This section describes examples showing how to set up and run simulations for oscillators and oscillator noise.

- [Finding the Frequency of Oscillation](#)
- [Calculating Large-Signal, Steady-State Oscillation Conditions](#)
- [Calculating Oscillator Loop Gain](#)
- [Using OscPort2 for Oscillator Analysis](#)
- [Simulating Phase Noise Using OscPort](#)
- [Simulating Phase Noise with NoiseCons](#)



**Note**

The Harmonic Balance simulation uses the Harmonic Balance Simulator license (sim\_harmonic) which is included with all Circuit Design suites except RF Designer. You must have this license to run Harmonic Balance simulations. You can work with examples described here and installed with the software without the license, but you will not be able to simulate them.

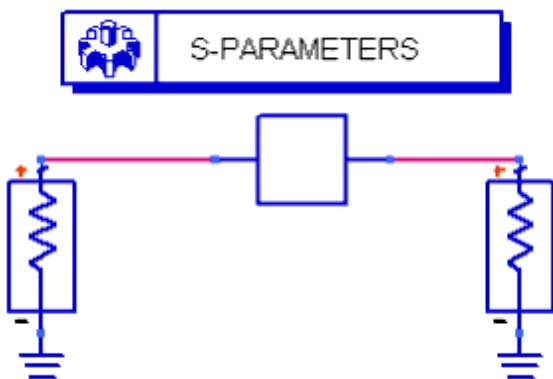
## Finding the Frequency of Oscillation

Fundamental to any oscillator analysis is establishing the frequency of oscillation. The simplest way to do this is to use the S-parameter simulator. The following figure 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 *S-Parameter Simulation* (cktsimsp).

**Note**

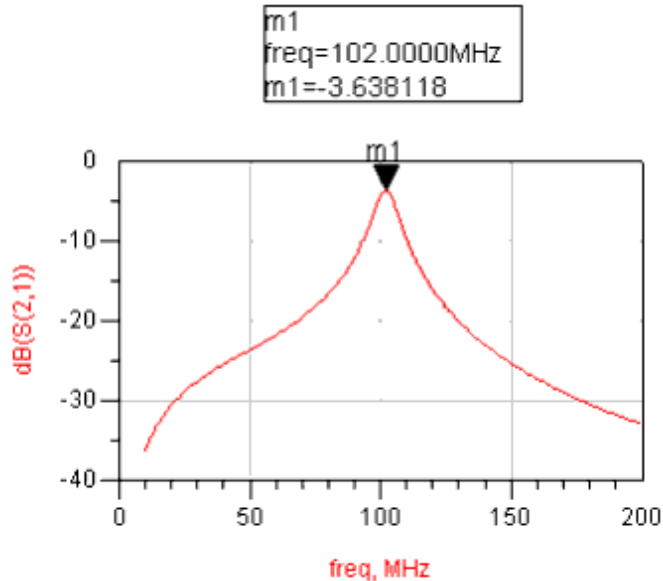
This design, *OscFeedbackTest*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *OscFeedbackTest.dds*.



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:



- 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

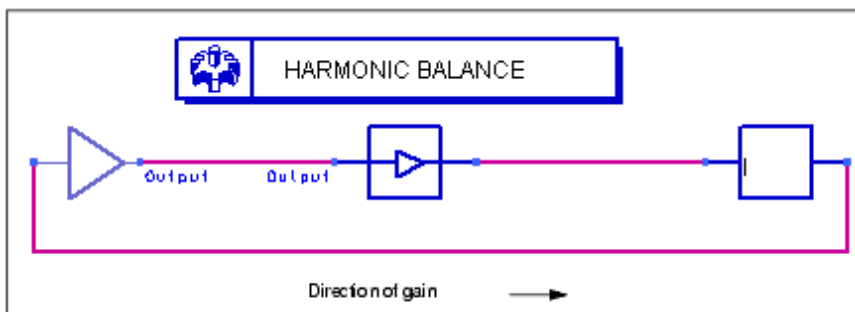
The following figure illustrates a setup that uses the harmonic balance simulator and OscPort component.

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](#).

### Note

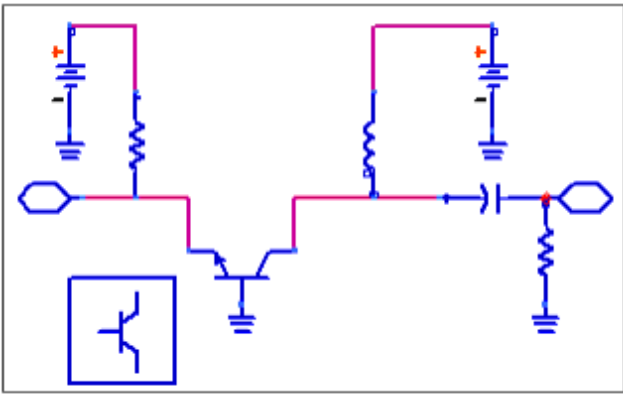
This design, *OSC2*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *OSC2.dds*.

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



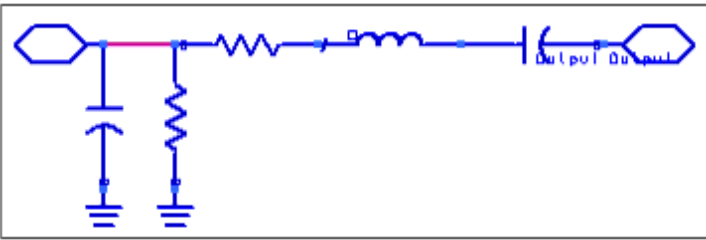
Example setup for a basic oscillator test using the harmonic balance simulator and the OscPort component

The next figure depicts the active gain stage.



Active gain subnetwork

The next figure depicts the resonant feedback stage.



Resonator subnetwork

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

- 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 the figure above, [Example setup for a basic oscillator test using the harmonic balance simulator and the OscPort component](#).
  - 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.

**Note**  
A circuit should contain only one oscillator port.

- 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.

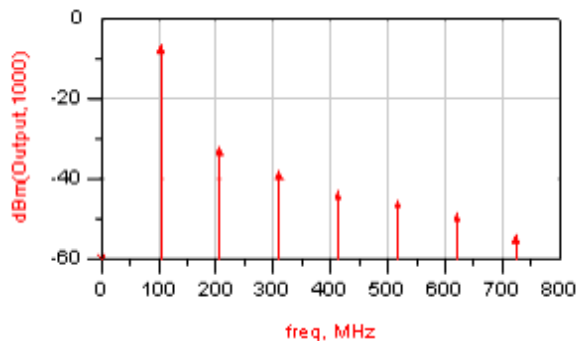
3. Scroll to the **Freq** tab and enter a frequency for the fundamental (Freq[1]), if you know it (in this example, **102 MHz** ). The oscillator will use this value and the values for the OscPort parameters *NumOctaves* and *Steps* to search a range of frequencies until the small-signal oscillation 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:  
**dBm(OSC2..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):



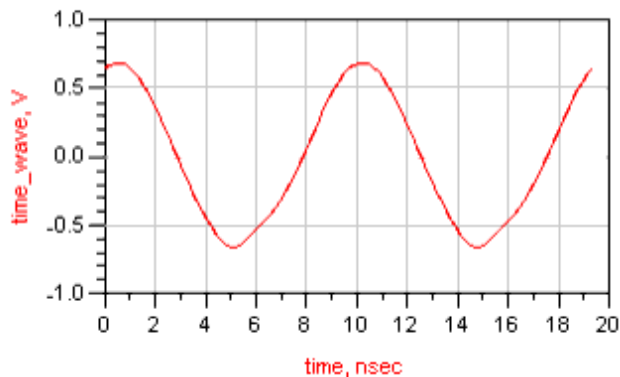
Plot voltage output of the gain stage over time:

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

**time\_wave=ts(Output)**

Plot the equation *time\_wave* to yield the following:

**Eqn** time\_wave=ts(Output)



## Calculating Oscillator Loop Gain

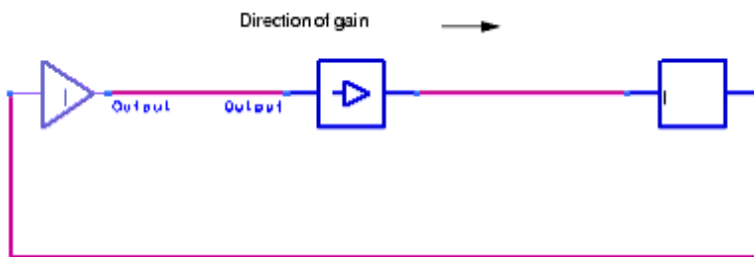
To calculate oscillator loop gain, use the *OscTest* component as described here. Because the *OscTest* component incorporates a simulation component into itself, no HarmonicBalance simulation component is required.

For oscillators with a differential topology, use the *OscPort2* component for calculating loop gain. For an example, refer to [Using OscPort2 for Oscillator Analysis](#).

The following figure illustrates a setup using the *OscTest* component.

### Note

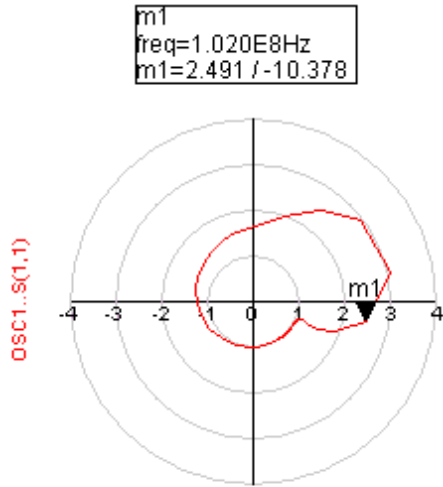
This design, *OSC1*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *OSC1.dds*.



Example setup for using the *OscTest* component

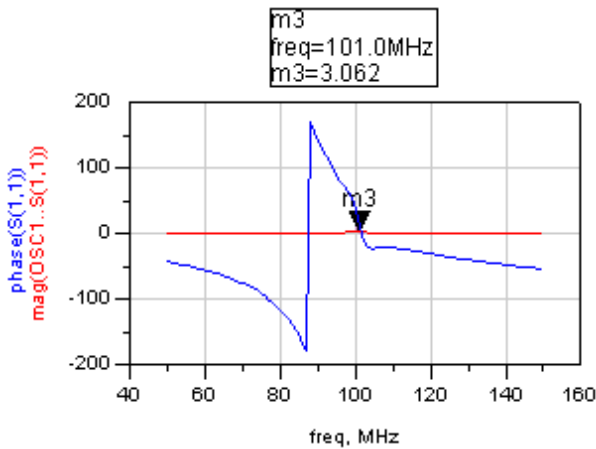
To determine oscillator loop gain:

- 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 the figure above.
  - 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.
- 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**
- 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.



freq (50.00MHz to 150.0MHz)

- 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.

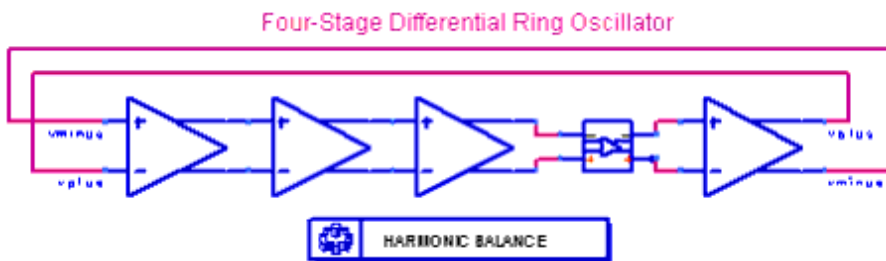
The *OscPort2* component provides the ability to easily handle oscillators with a differential, or balanced, circuit topology, which are commonly found in integrated circuits. *OscPort2*

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

The following figure shows 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*, is in the *Examples* directory under *Tutorial/OscPort2\_wrk*. The results are in *hbosc.dds*.



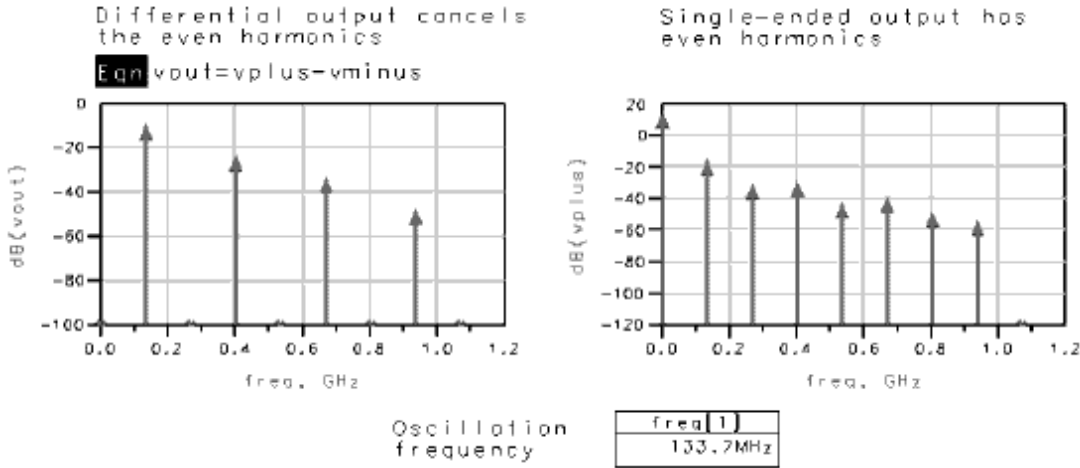
### Differential ring oscillator

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.

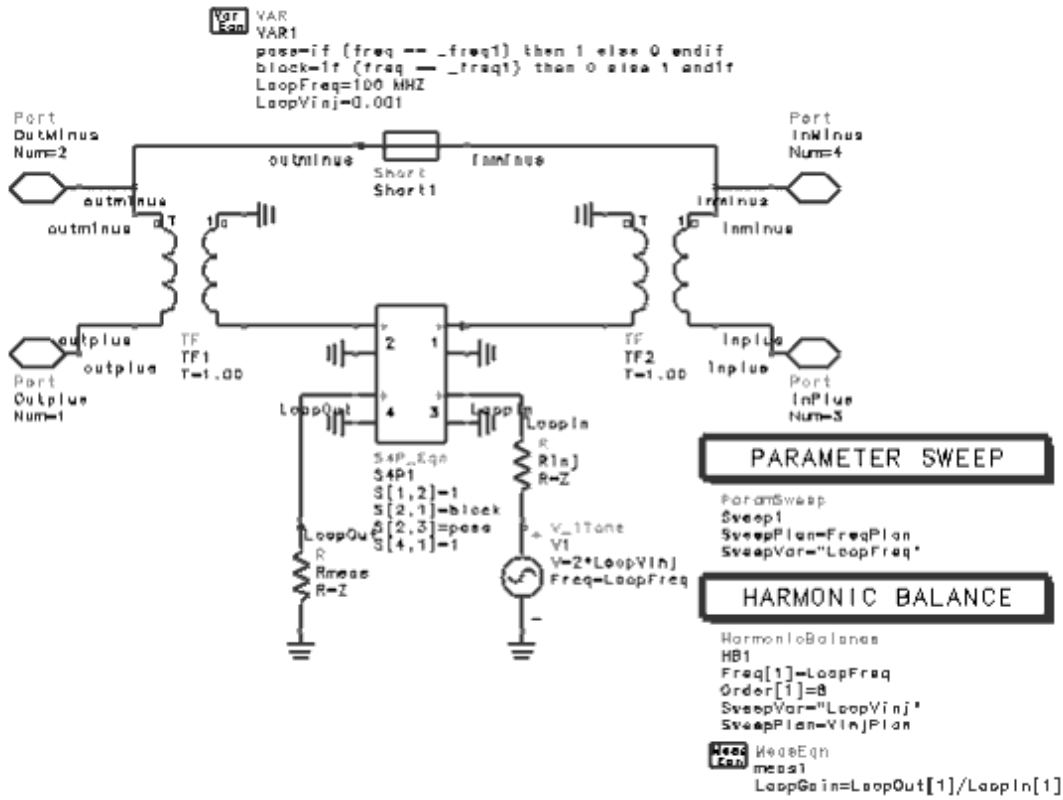


## Calculating Large-Signal Loop Gain

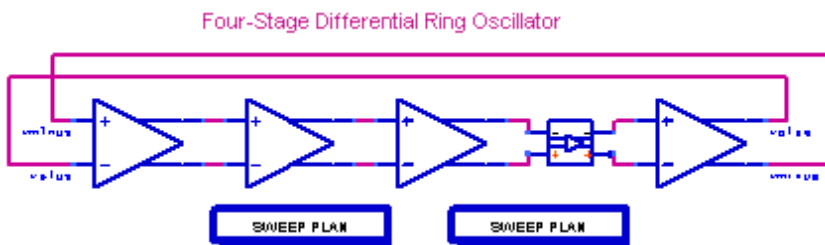
OscPort2 can be used to simulate the large-signal loop gain of an oscillator using harmonic balance. The equivalent circuit of the OscPort2 in Large signal loop gain mode is shown below. The OscPort2 is used to break the oscillator feedback loop at the fundamental frequency only. The loop remains connected 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*, is in the *Examples* directory under *Tutorial/OscPort2\_wrk*. The results are in *loopgain\_large.dds*





Equivalent circuit of OscPort2 in large-signal loop gain mode



Simulation setup to simulate large-signal loop gain of an oscillator

Set the Mode parameter (of OscPort2) to *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*.

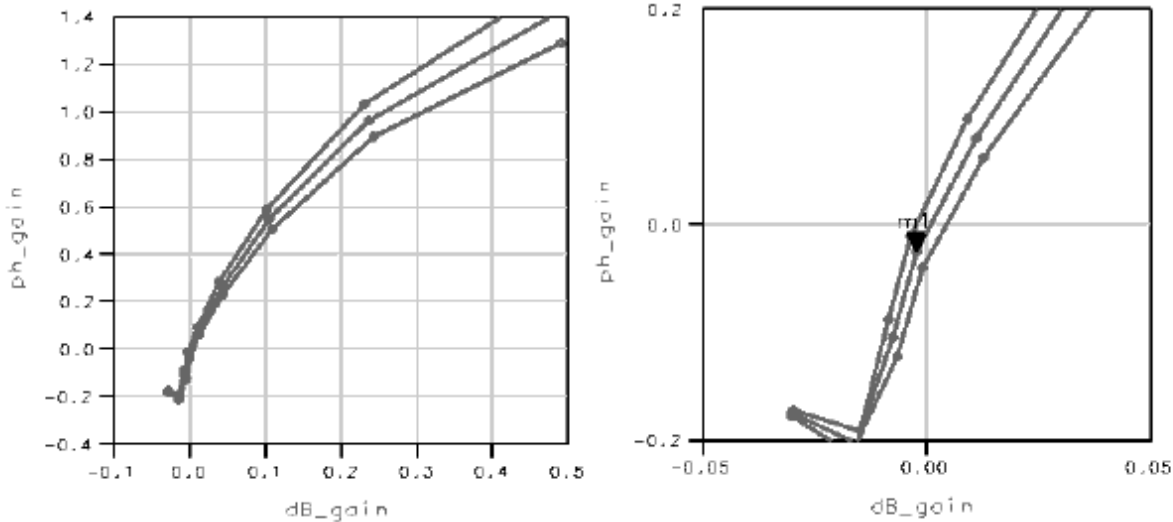
Oscillation will occur at the point  
where gain=0 dB and phase=0 deg.

These are curves of constant frequency  
with swept injected voltage.

Eqn dB\_gain=dB(LoopGain)

Eqn ph\_gain=phase(LoopGain)

```
m1
dB_gain=-0.002
oscport1.LoopFreq=1.340000E8
ph_gain=-0.024
```



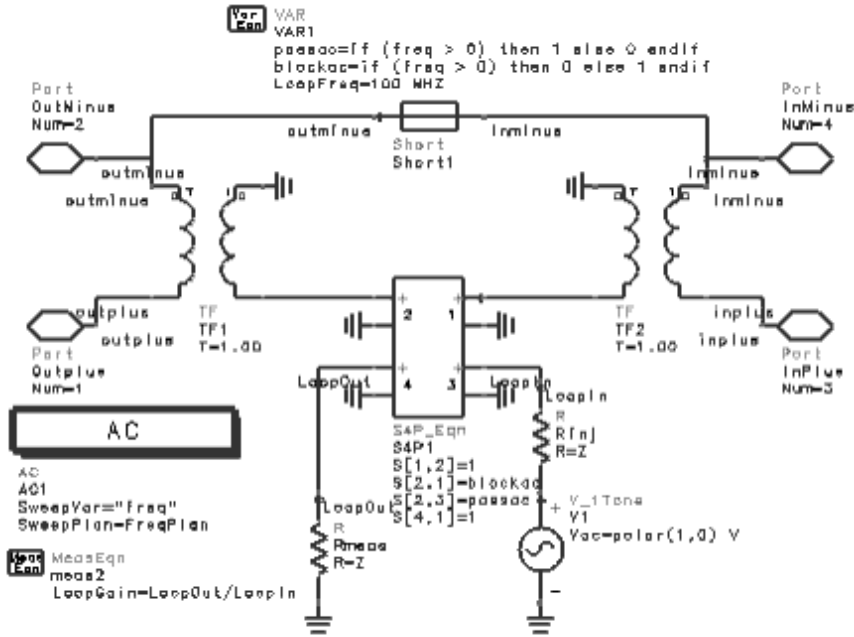
The simulation results are shown above. In the dataset, the result named *LoopGain* is the complex loop gain for the oscillator. One typical way to present this information is to plot the loop gain phase against the loop gain magnitude in dB. The point of oscillation will then be at the 0,0 point on 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

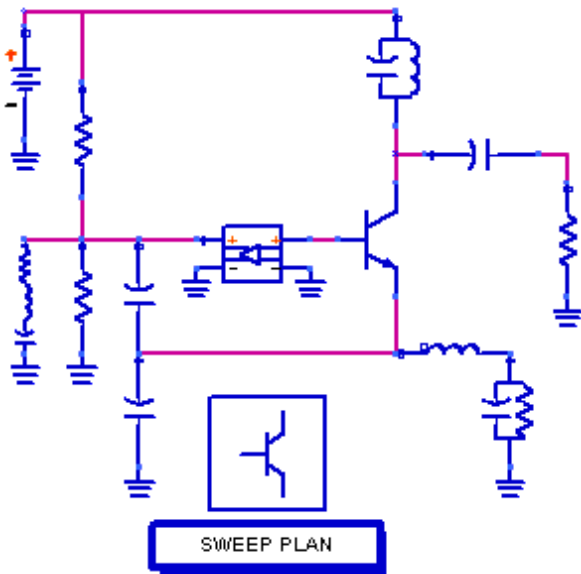
OscPort2 can be used to simulate the small-signal loop gain of an oscillator using small-signal AC analysis (like the OscTest component allows). The equivalent circuit of OscPort2 in Small-signal loop gain mode is shown below. The OscPort2 is used to break the oscillator feedback loop at the small signal 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  $\text{LoopOut}/\text{LoopIn}$  and is placed in the dataset with the name *LoopGain*. The user can sweep the injected signal frequency. The circuit oscillates near the injected frequency where loop gain is approximately  $1+j0$ . The loop gain calculation finds the point where the real part is maximum and the imaginary part is zero. In this example, the real part is slightly greater than one.

### Note

This design, *loopgain\_small*, is in the *Examples* directory under *Tutorial/OscPort2\_wrk*. The results are in *loopgain\_small.dds*



Equivalent circuit of OscPort2 in small-signal loop gain mode

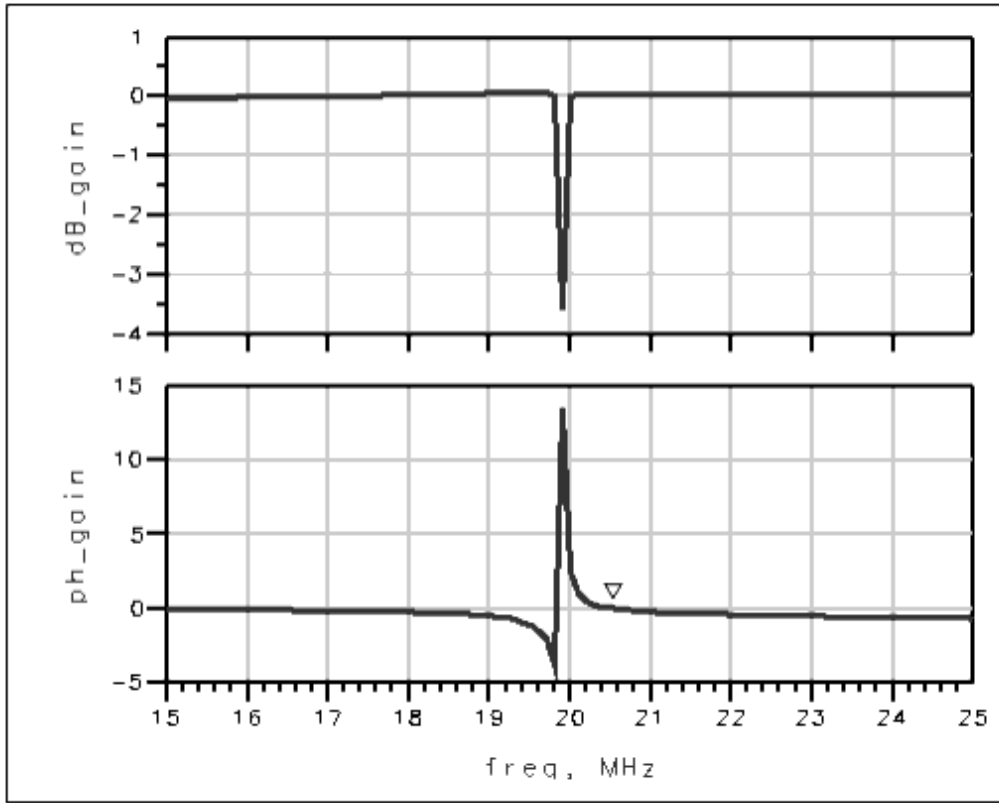


Simulation setup to simulate the small-signal loop gain of an oscillator

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*.

```
Eqn dB_gain=dB(LoopGain)
```

```
Eqn ph_gain=phase(LoopGain)
```



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 20.55 MHz.

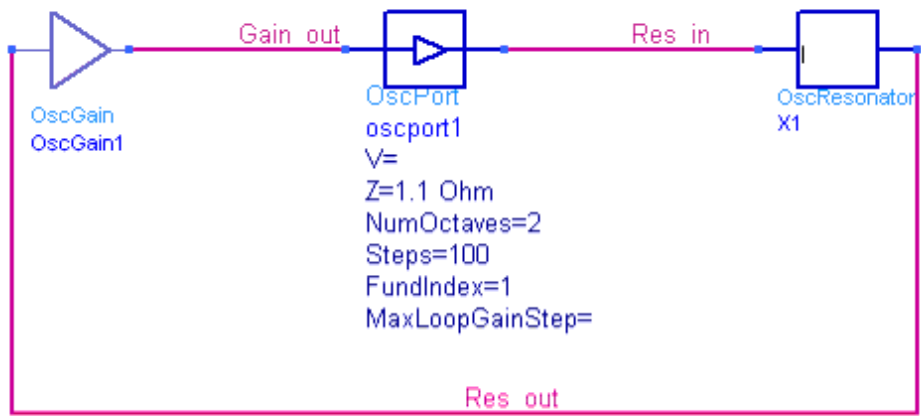
## Simulating Phase Noise Using OscPort

The following figure illustrates an example setup for determining oscillator phase noise.

### Note

This design, *OSC3*, is in the *examples* directory under *Tutorial/SimModels\_wrk*. The results are in *OSC3.dds*.

This example is similar to [Calculating Large-Signal, Steady-State Oscillation Conditions](#) with the addition of the settings described below.



#### HarmonicBalance

```

HB1
Freq[1]=102 MHz
Order[1]=15
Oversample[1]=
NLNoiseMode=yes
NLNoiseUseSweepPlan=
NLNoiseStart=1 kHz
NLNoiseStop=10 MHz
NLNoiseDec=2
PhaseNoise=yes
NoiseNode[1]="Res_in"
NoiseNode[2]="Gain_out"
NoiseNode[3]="Res_out"

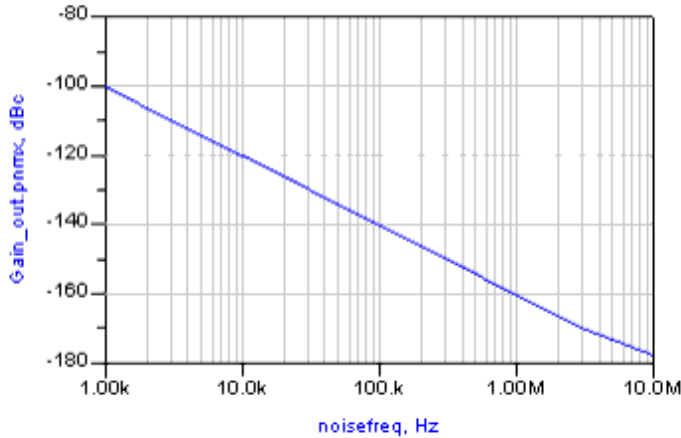
```

#### Setup for determining oscillator phase noise

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 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.pnm<sub>x</sub>*, for phase noise due to noise mixing. The plot appears as follows:

Phase Noise at node "Gain\_out"



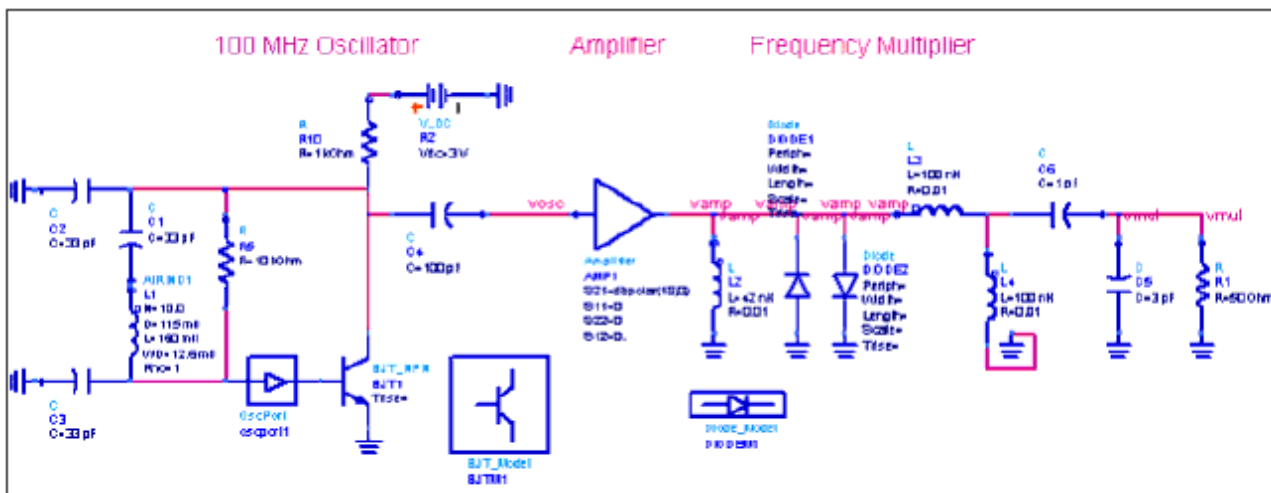
Notice that the noise is at least 100 dB below the fundamental (resonant) frequency.

## 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.

The following figure shows an oscillator with a frequency multiplier that will be used to demonstrate the use of NoiseCons with an oscillator.

**Note**  
Refer to the ADS workspace *examples/Tutorial/Noisecon\_wrk* for an illustration of how to use NoiseCons for oscillator phase noise simulation. See *oscmul* for the design and *oscmul.dds* for the results.



**Setup for demonstrating NoiseCons with an oscillator**

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

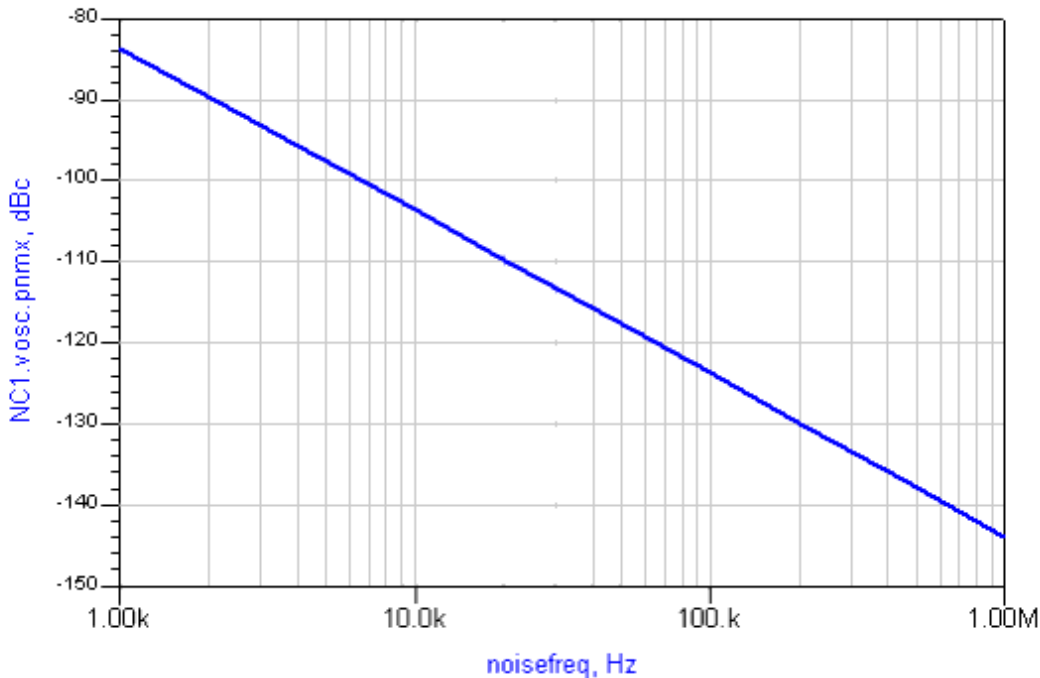
**Single Sideband Phase Noise**

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 result in the dataset has the name *HB1.NC1.vosc.pnm* and is phase noise in dBc. Refer to [How ADS Simulates Phase Noise](#) for more information on phase noise analyses.

## NC1: vosc Phase Noise



## Relative Noise Voltage Spectrum

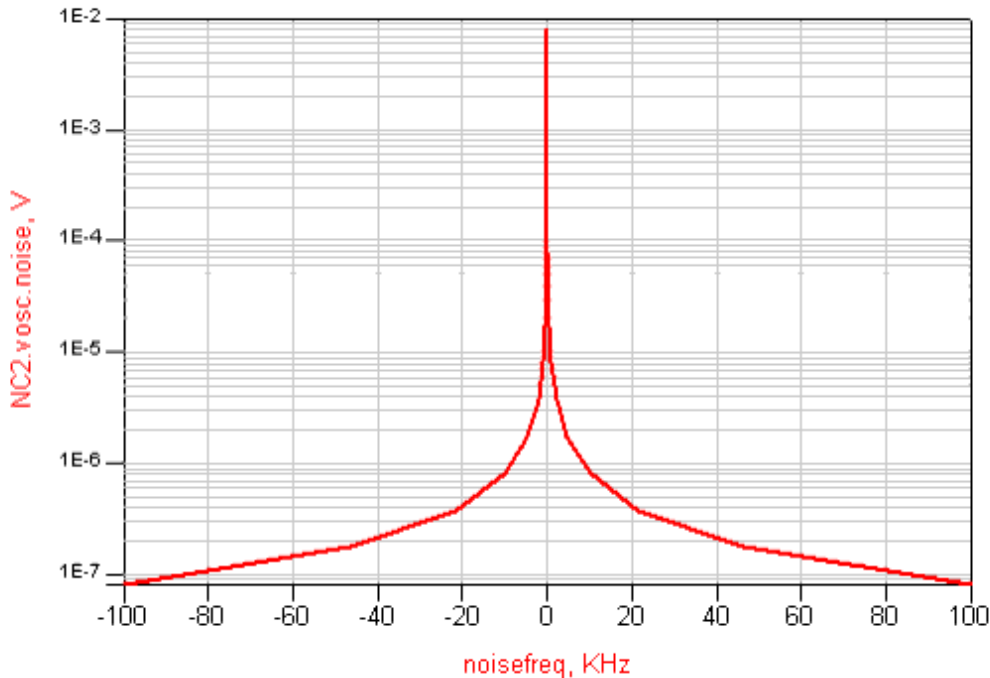
The second NoiseCon illustrates the relative noise voltage spectrum. Rather than compute the single sideband phase noise, the noise voltage is computed both at the carrier frequency minus the offset frequency and at the carrier frequency plus the offset frequency. The results are presented as a relative 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 **NC2**.
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.



## NC2: vosc Noise Spectrum



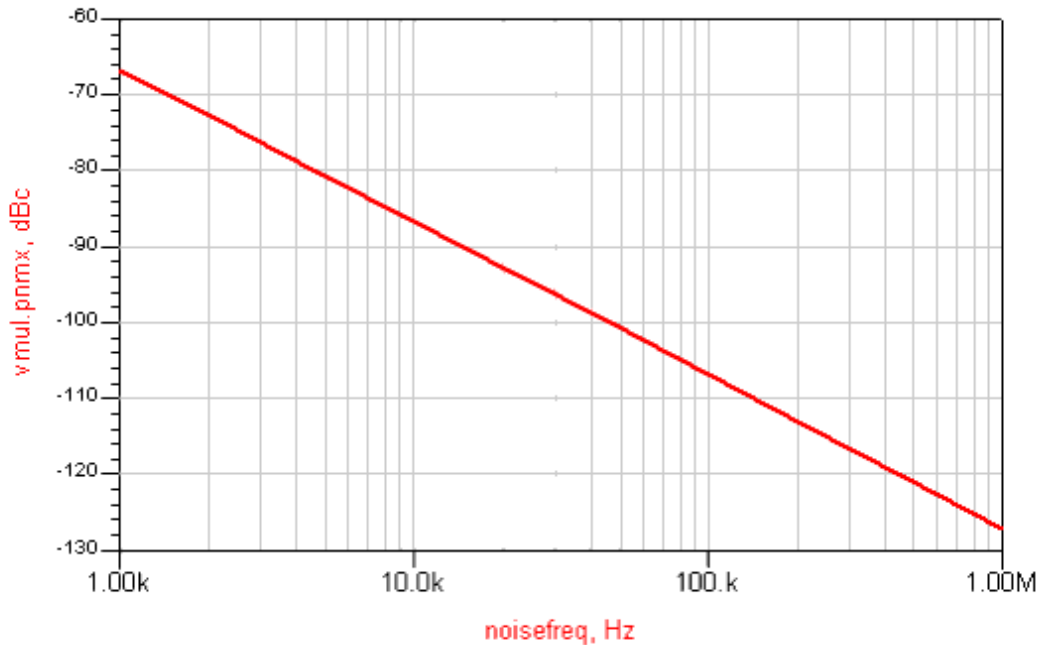
### Phase Noise Voltage Around a Harmonic

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 Harmonic Balance 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 result in the dataset has the name *HB1.NC3.vosc.pnm* and is phase noise in dBc. Refer to [How ADS Simulates Phase Noise](#) 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.

## NC3: vmul Phase Noise



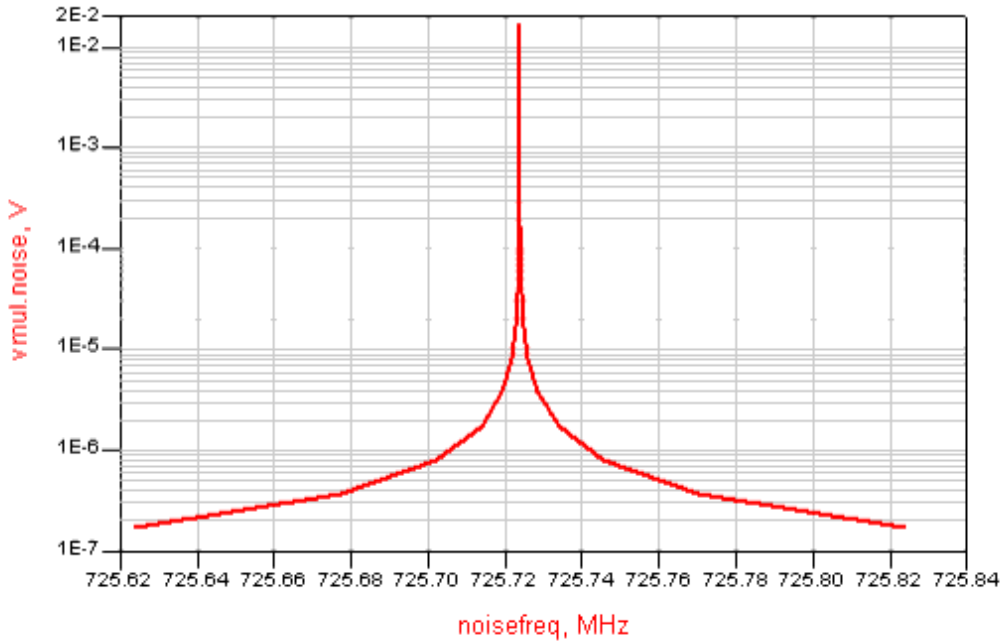
### Absolute Noise Voltage Spectrum

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 725.62 MHz to 725.82 MHz, centered around the seventh harmonic of the oscillator.

## NC4: vmul Noise Spectrum



## Oscillator Simulation Description

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.

When the Specify Oscillator Nodes method is used instead of placing an OscPort component, the concept of loop gain is lost. In this mode the simulator connects a special voltage source to the specified node(s). This special source is connected to the circuit at only the fundamental frequency; nothing is connected at DC or the harmonics. This special source injects a voltage at the fundamental frequency to excite the oscillator. The oscillation will be self sustained when the voltage and frequency are such that the voltage is nonzero and no current flows from the special source.

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.

## Performing VCO Tuning

VCO tuning is accomplished by performing an oscillator simulation in conjunction with a swept variable analysis (refer to *Parameter Sweeps and Sweep Plans* (cktsim)). 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

$$|G_l| > 1, \angle(G_l) = 0$$

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

$$R_e(Z_{out}) < 0, I_m(Z_{out}) = 0$$

For a more rigorous description of the oscillation conditions that must be satisfied, refer to the oscillator component descriptions in *Introduction to Circuit Components* (ccsim).



**Note**

Because the control items are built into OscTest, no analysis controller is necessary.

In addition, an example of this type of analysis can be found in the example file `/examples/Tutorial/LearnOsc_wrk/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.

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.

## Phase Noise Simulation Description

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.

The results from the phase noise analysis have *.pnmx* (phase noise, in dBc/Hz) and *.anmx* (AM noise, in dBc/Hz) appended to the selected node name(s).

**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.

## How ADS Simulates Phase Noise

Phase noise in an oscillator can be analyzed from small-signal mixing of noise. 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.

The phase noise computed 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.

`hb_noise.vout.pnmx` (phase noise (in dBc) from mixing analysis)

`hb_noise.vout.anmx` (AM noise in (dBc) from mixing analysis)

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  $(\omega \pm \kappa\omega_0)$  mix with the oscillator large signals  $(\kappa\omega_0)$  to produce noise

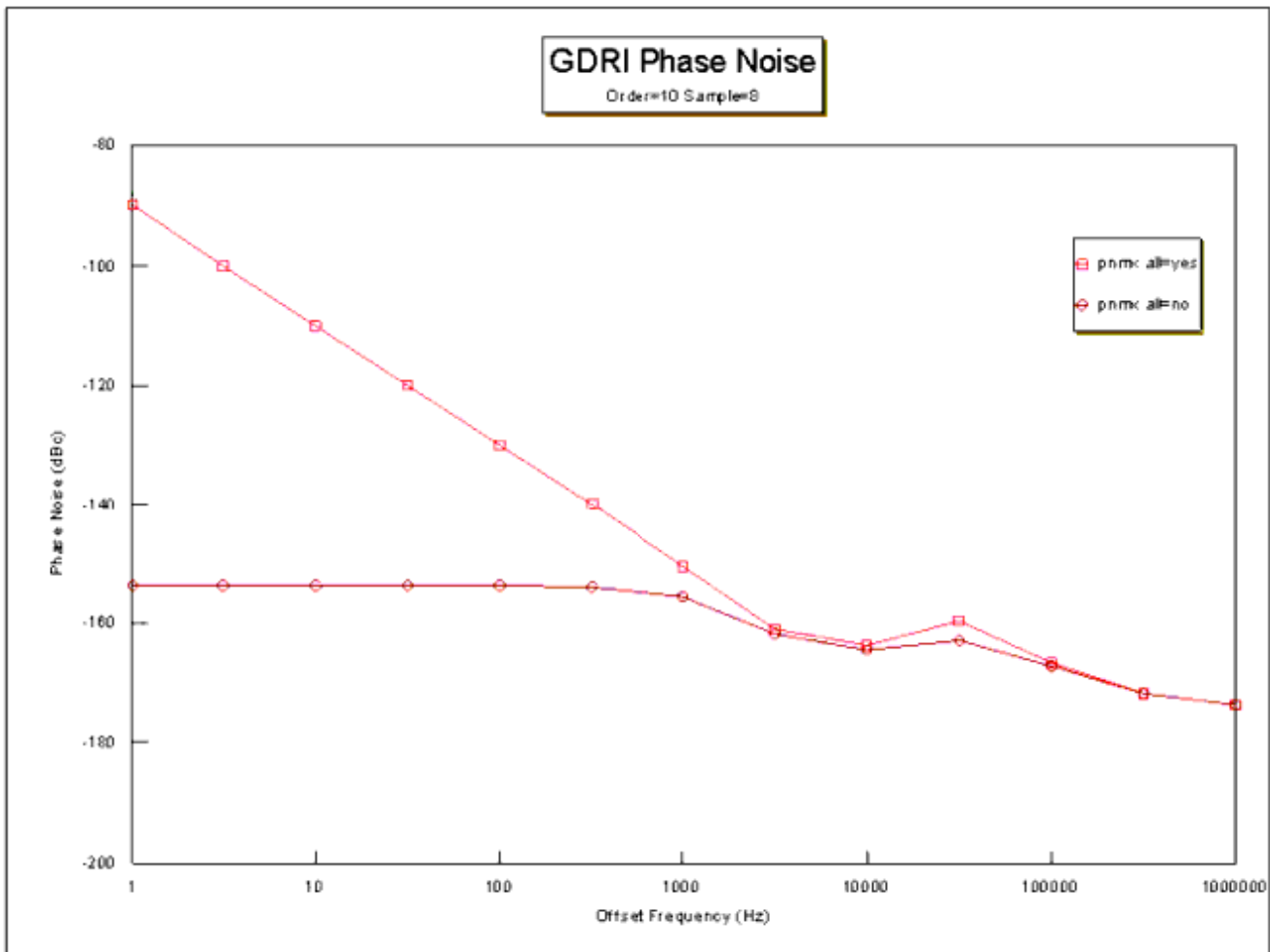
sidebands. The noise at these two sideband frequencies and their correlation is then used to compute the phase noise. The mixing analysis additionally computes the AM noise as well as phase noise.

## Possible Problems with Phase Noise Analysis

When simulating phase noise, it is recommended to set the *Oversample* parameter to 4 or higher.

For the mixing analysis, it is important that the analysis uses all of the small signal frequencies. By default, the simulator will use all of the small signal frequencies. The only way to make it not use all of them, and degrade the accuracy of the noise results, is to perform a small-signal analysis along with the noise analysis, and set *Use all small-signal frequencies* to *no* there. 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 ( $2 \cdot \text{order} + 1$  small-signal tones); otherwise sidebands are used only around the lowest half of the large-signal frequencies ( $\text{order} + 1$  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.

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 (*all=yes*) and when they are not used (*all=no*.)

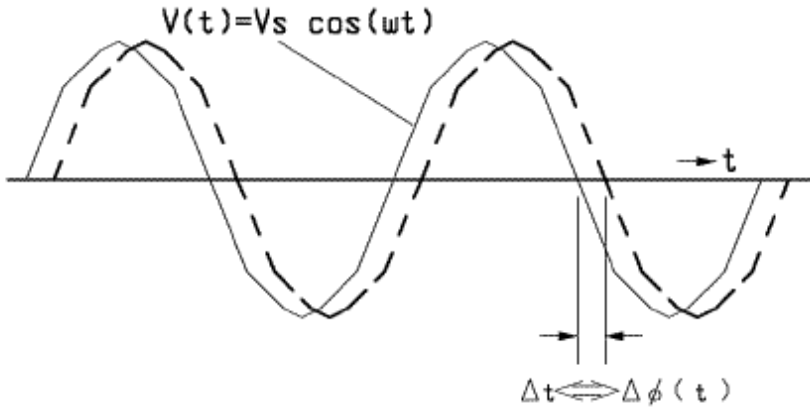


## Basic Phase Noise Theory

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 following figure).



### Phase Noise Represented in the Time Domain

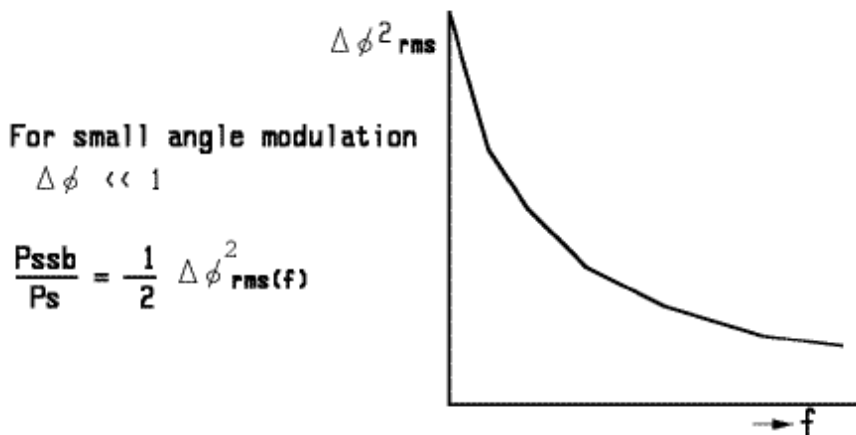


Frequency and phase are related by :

$$f(t) = \frac{1}{2\pi} \frac{d\phi(t)}{dt}$$

### 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:

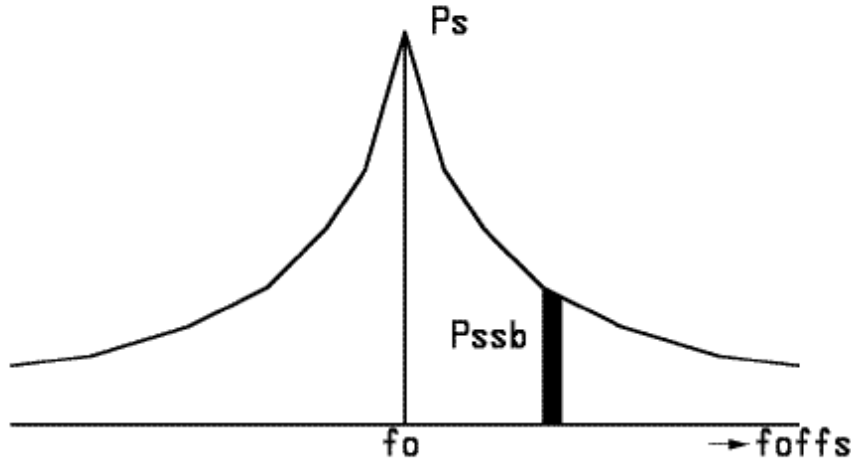


### Single-Sideband Noise Power in a 1-Hz Bandwidth

A common but indirect representation of phase noise is denoted  $L(f)$  (see the following figure). 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.

### Definition of $L(f)$



$$L(f) = \frac{P_{ssb}(\text{per 1Hz})}{P_s} \text{ [dBc/Hz]}$$

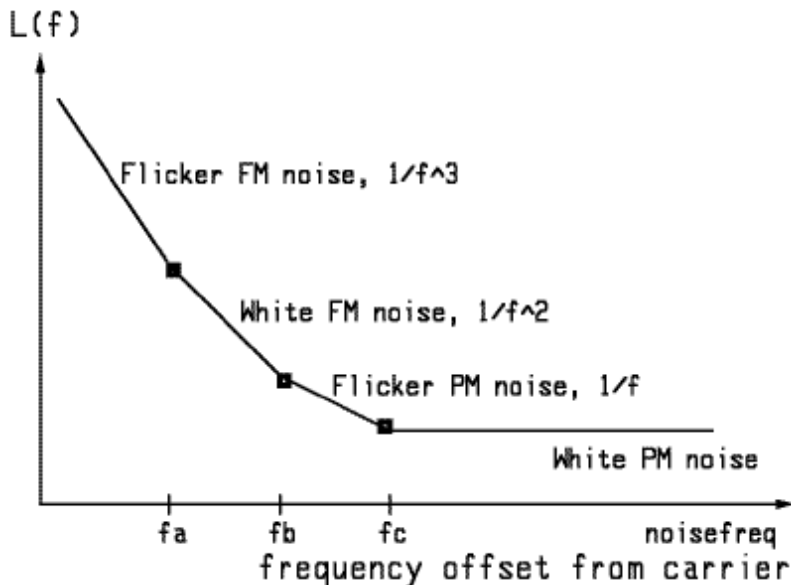
### Oscillator Phase-Noise Analysis

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  $1/f$ .
- *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.

## 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](#)).

- Ensure that the OscPort component is connected to a node in the circuit where a nonzero voltage at the fundamental oscillation frequency is expected. It should not be connected to a node that contains only a DC bias voltage. Good locations to pick are at the input or output of the active device, within the feedback loop, or between the negative resistance and the resonator.
- If oscillator nodes are specified instead of using an OscPort, good nodes to pick are the input or output nodes of the active devices or nodes in the resonator tank. Do not pick nodes in the bias circuitry or nodes in an output buffer amplifier.
- 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.
- Oscillator analysis works best with default convergence tolerances. Avoid loosening tolerances, especially *I\_AbsTol*, unless absolutely necessary.

## 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 circuit is properly biased and that either the OscPort component is located properly or the correct oscillator nodes were identified. If this is so, modify the OscPort parameters *Octaves to Search*, *Steps per Octave*, and *MaxLoopGainStep* – notice high-Q effects.
- If the approximate frequency search fails, switch to the fixed frequency search. Set the *Octaves to Search* parameter to zero, make sure that the *Frequency* specified for the Fundamental Frequencies on the *Freq* tab is close to the expected frequency of oscillation, and resimulate. This analysis mode is useful for oscillators that don't display the usual concept of small-signal loop gain greater than one, such as ring oscillators.
- 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.

## Additional Potential Oscillation Frequencies

Some circuits may have more than one oscillation frequency. During the linear frequency search, which is done at the beginning of the analysis, the simulator will determine if the circuit contains additional potential oscillation frequencies. In the case that more than one frequency is found during the linear frequency search, the simulator uses the smallest oscillation frequency. A warning message will be given in the status window if additional oscillation frequencies are found, and the frequency values of the additional ones. The message will also state the frequency at which the simulation will continue for the remainder of the analysis. If additional potential frequencies are found, it is recommended that you re-examine your circuit design.

When performing an oscillator simulation (of a ring oscillator, for example) using *NumOctaves=0* on the OscPort component, or when *Octaves to Search=0* when specifying oscillator nodes, the simulator uses an alternate method to find the oscillation frequency. With this approach, the simulator does not perform the linear frequency search. Thus, it will not detect any additional potential oscillation frequencies in this mode.

## 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  $|\Delta F| \leq \epsilon_{\text{rel}}|F| + \epsilon_{\text{abs}}$  where  $\epsilon_{\text{rel}}$  is *Frequency relative tolerance* and  $\epsilon_{\text{abs}}$  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.

## Simulation Techniques for Recalcitrant Oscillators

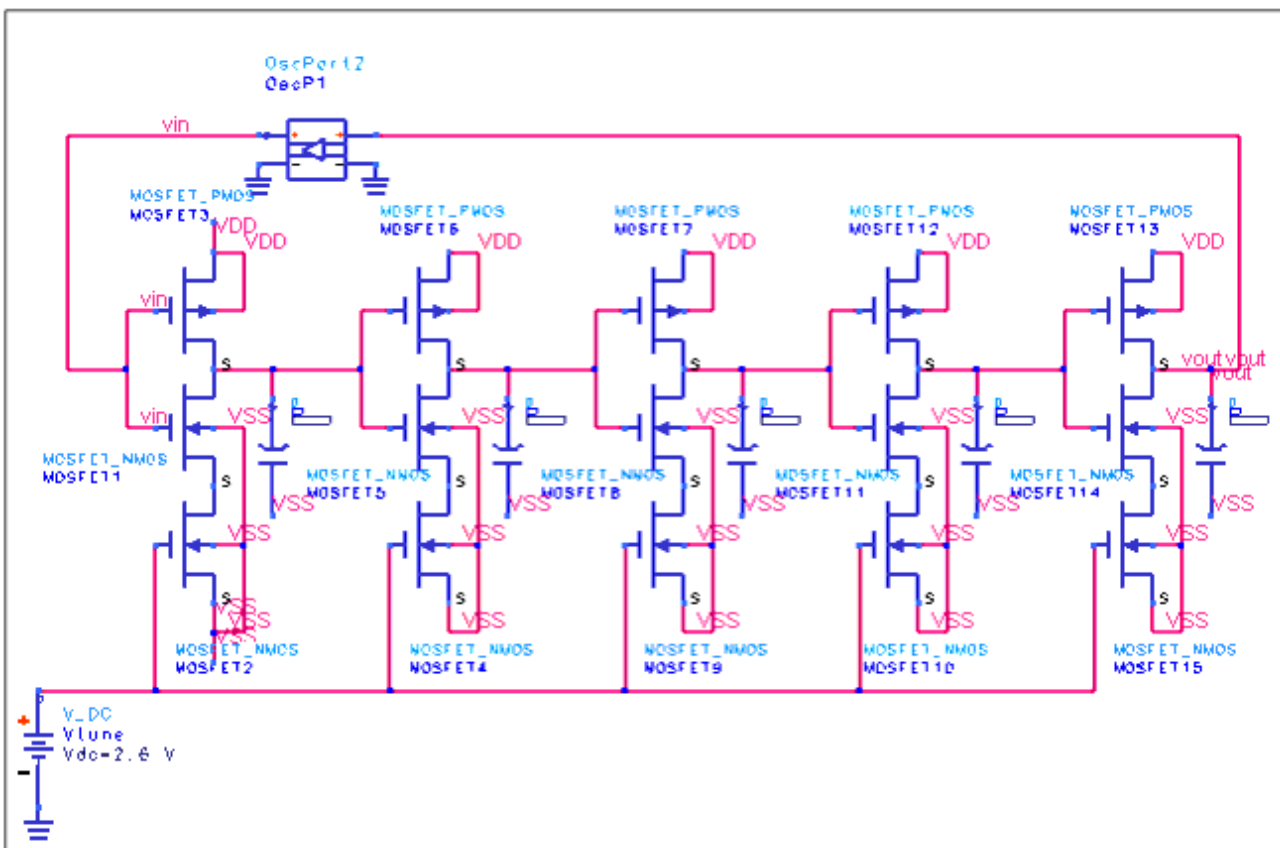
**Note**

If you have purchased and installed the Oscillator DesignGuide, you can access the examples shown in this topic by selecting *DesignGuide > Oscillator > Solving HB Convergence Problems* from the Schematic window.

Harmonic balance simulation in ADS is an excellent way to analyze many oscillators in the frequency domain. Occasionally, you might have an oscillator that converges in a time-domain simulation, but the harmonic balance oscillator algorithm is unable to find the solution. There are two techniques for solving those oscillators:

- Analyze the large-signal loop gain in harmonic balance to find the point of oscillation and using that as an initial guess for the full harmonic balance oscillator analysis.
- Use a Transient Assisted Harmonic Balance.

The oscillator shown in the following figure is the starting point for our test case. This is a five-stage CMOS ring oscillator that oscillates at 86 MHz. The following two techniques that are demonstrated use a ring oscillator, but they can be used with any type of oscillator that won't converge in a standard harmonic balance oscillator simulation.



**Five-stage CMOS ring oscillator**

Transient analysis shows that it works, but if we run a harmonic balance simulation, the analysis fails with the following error message:

Error detected by HPEESFSIM in frequency search during HB analysis

`HB5'.

Cannot find a frequency where loop gain phase equals 0.0 degrees.  
The magnitude of the loop gain is greater than one but additional  
phase shift around the loop is needed.

Warning detected by HPEESOFSIM during HB analysis `HB5'.

Oscillator analysis did not converge. Setting solution to zero.

Warning detected by HPEESOFSIM during HB analysis `HB5'.

More than one mixing term has landed on frequency 0 Hz.

## Using Transient Assisted Harmonic Balance

The simplest method is to use transient assisted harmonic balance. Simply set TAHB to *On* under the Initial Guess tab in the HB controller and leave all other fields blank to use default settings. The HB controller in the schematic is shown in the following figure.



HarmonicBalance

HB2

Freq[1]=861.0 MHz

Order[1]=15

NLNoiseStart=1.0 kHz

NLNoiseStop=10.0 MHz

NLNoiseDec=2

PhaseNoise=yes

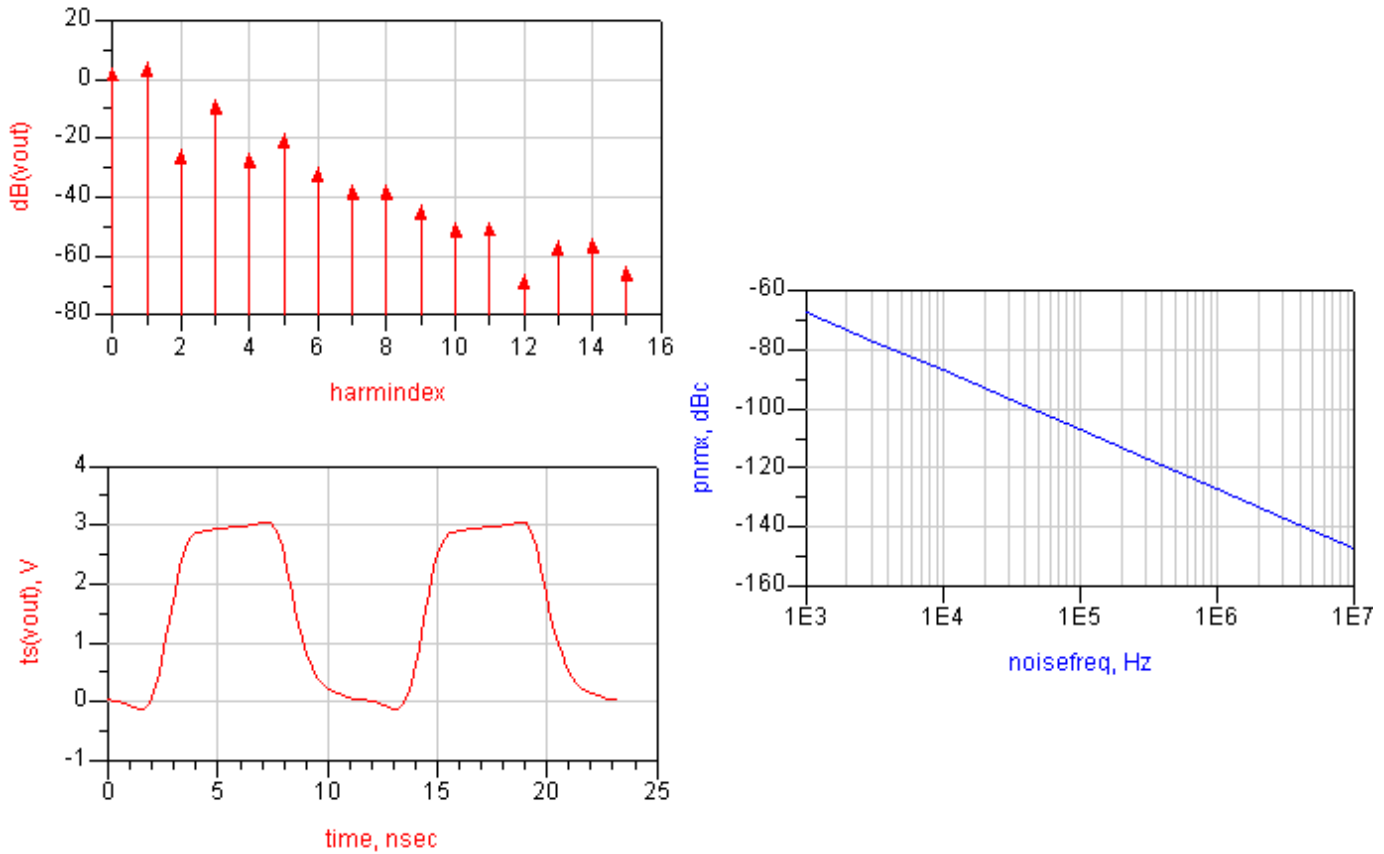
NoiseNode[1]="vout"

OscPortName="Yes"

TAHB\_Enable=On

### Setup for harmonic balance analysis with TAHB set to On

The simulation goes quickly and the results are shown in the following figure.



Results using transient assisted harmonic balance

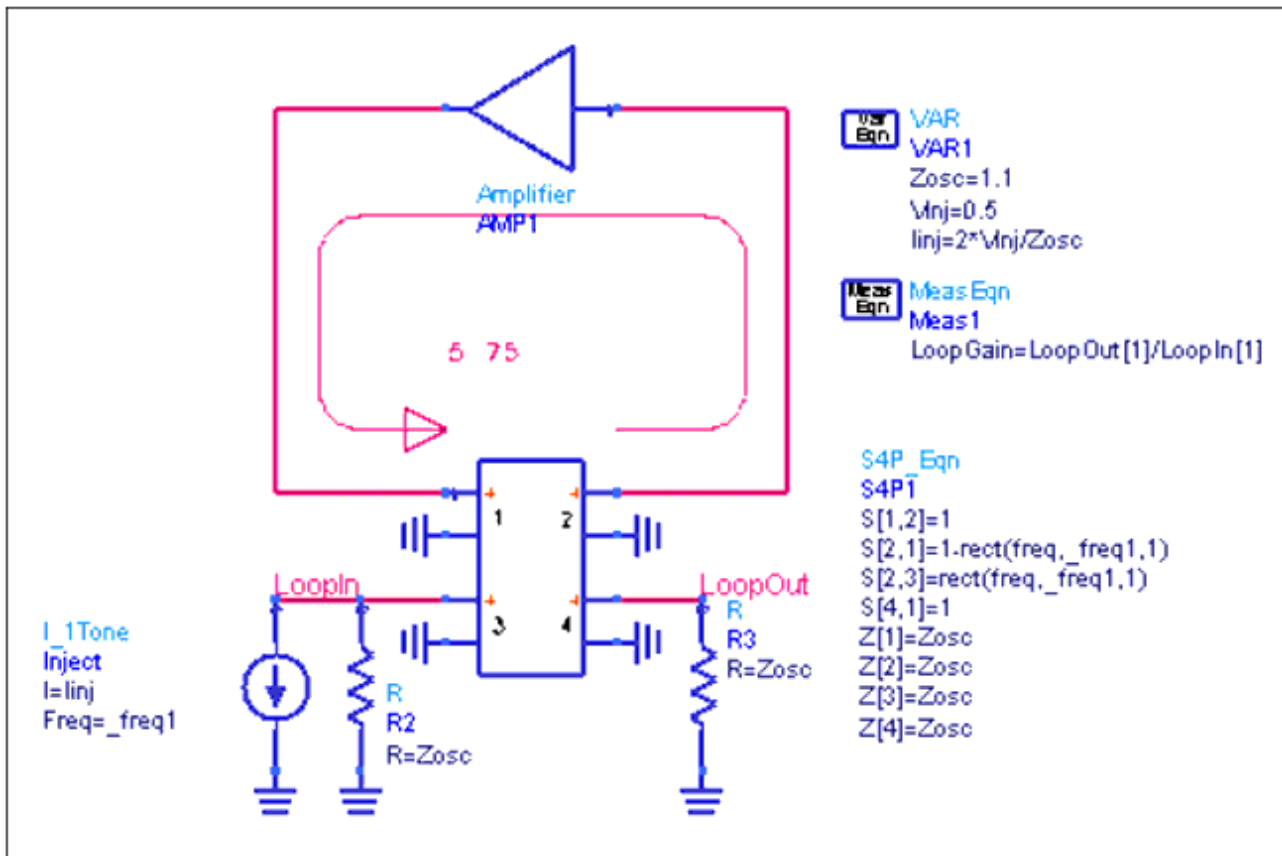
## Large-Signal Loop Gain Analysis

We will take the view that an oscillator is an amplifier with a feedback loop. An initial small signal due to thermal noise or a startup impulse from a power supply goes around the loop and gets amplified and phase shifted by some small-signal gain. This larger signal continues to go around the loop, getting larger. At some point, the nonlinear nature of the amplifier kicks in, and the gain starts to decrease due to gain compression as the signal gets larger. At this point, we have to start thinking of it as a large-signal gain that is dependent on the amplitude of the signal. Ultimately, the signal gets so large that the gain is reduced to unity, and if the signal gets any larger, the gain is less than unity. This is the limiting nature of an oscillator.

At the frequency at which the large-signal gain is unity, the circuit is capable of sustaining the same signal it oscillates. Another requirement for oscillation is that the phase shift around the loop is zero (really a multiple of 360 degrees). So we can say that an oscillator will oscillate at some fundamental frequency with some voltage at that frequency when that signal would set unity gain and zero phase shift around the loop.

The circuit shown in the following figure shows how we can simulate large-signal loop gain in ADS.





### Simulating large signal loop gain

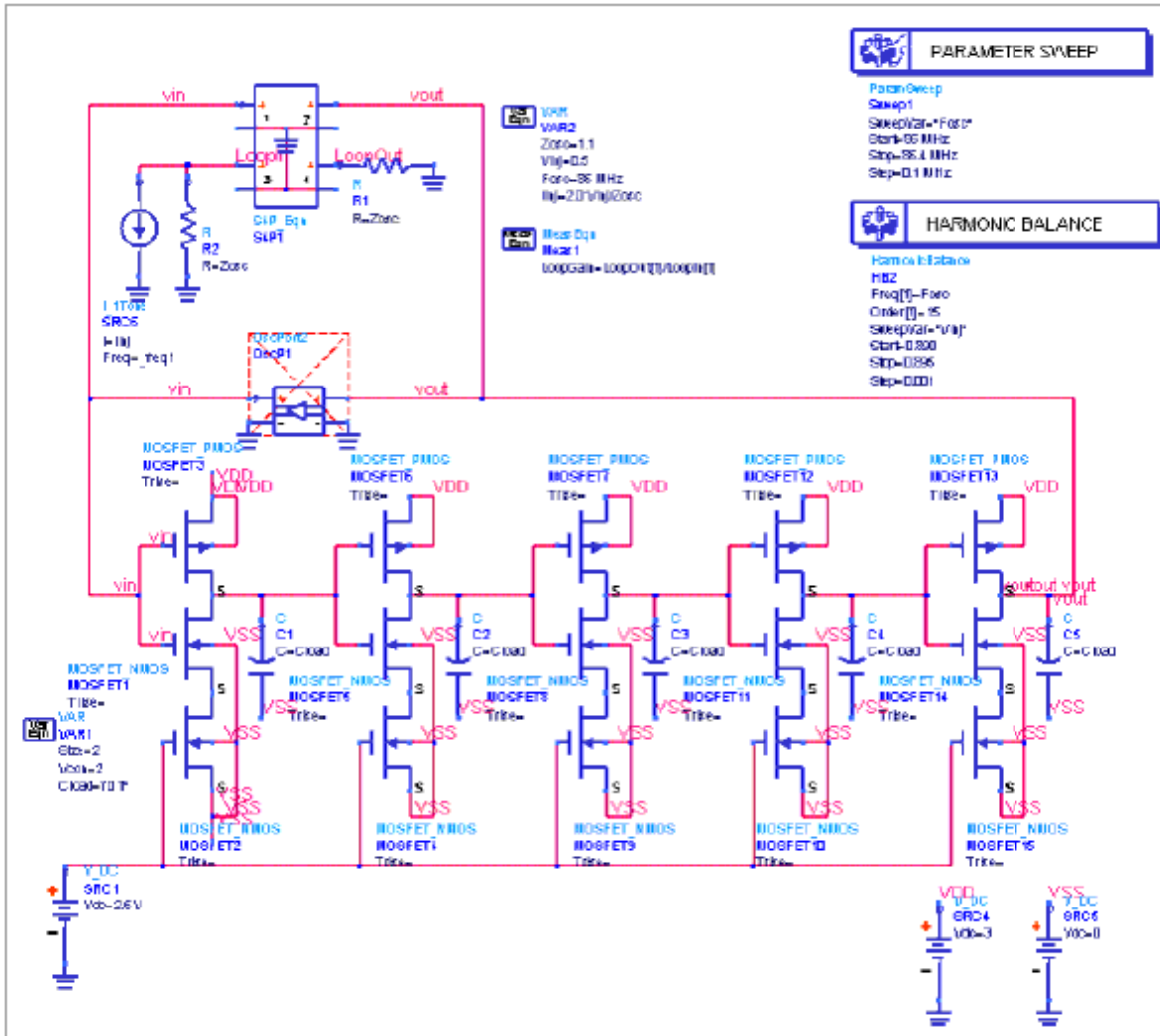
The S4P is defined to have some special properties.:

- $S[2,1]$ , the forward signal flow from port 1 to port 2, is one at all frequencies except the fundamental is zero.
- $S[1,2]$ , the reverse signal flow from port 1 to port 2, is one everywhere.
- $S[2,3]$  defines the injection from the source at port 3 into the loop at port 2; it is set up to pass only the fundamental frequency.
- $S[4,1]$  passes the full signal coming out of the loop to the termination at port 4. This terminates the fundamental but duplicates the rest of the signal (DC and harmonics). In this way, we have opened the feedback loop only at the fundamental frequency, allowing DC and the harmonics to continue to flow around the loop.

We can now inject a large signal tone at the fundamental frequency into the feedback loop and observe the fundamental after it travels around the feedback loop. We can then define the loop gain as the ratio of the voltage at the output to the voltage injected into the loop, both at the fundamental frequency. This analysis capability is built into the OscPort2 component in its large-signal loop gain mode.

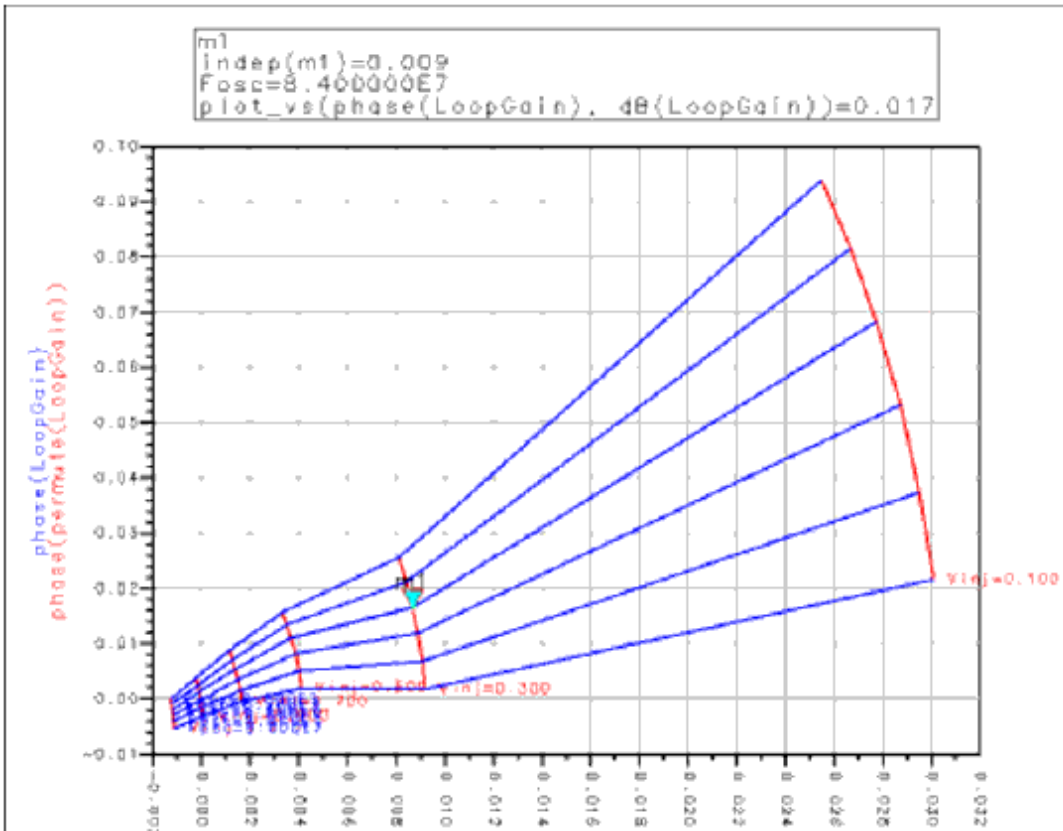
### Swept Large-Signal Loop Gain

We can now start our oscillator analysis. We will use a discrete version of the loop gain measurement rather than use the OscPort2, as it will be needed in the next step. The simulation setup is shown in the following figure. The OscPort2 is disabled, as it is replaced by the S4P circuit for this analysis. A harmonic balance analysis will be done where both the injected voltage  $V_{inj}$  and the fundamental frequency  $F_{osc}$  are swept. This is not an oscillator analysis, because we are specifying the analysis frequencies and driving the circuit with a source.



### Setup for swept large signal loop gain

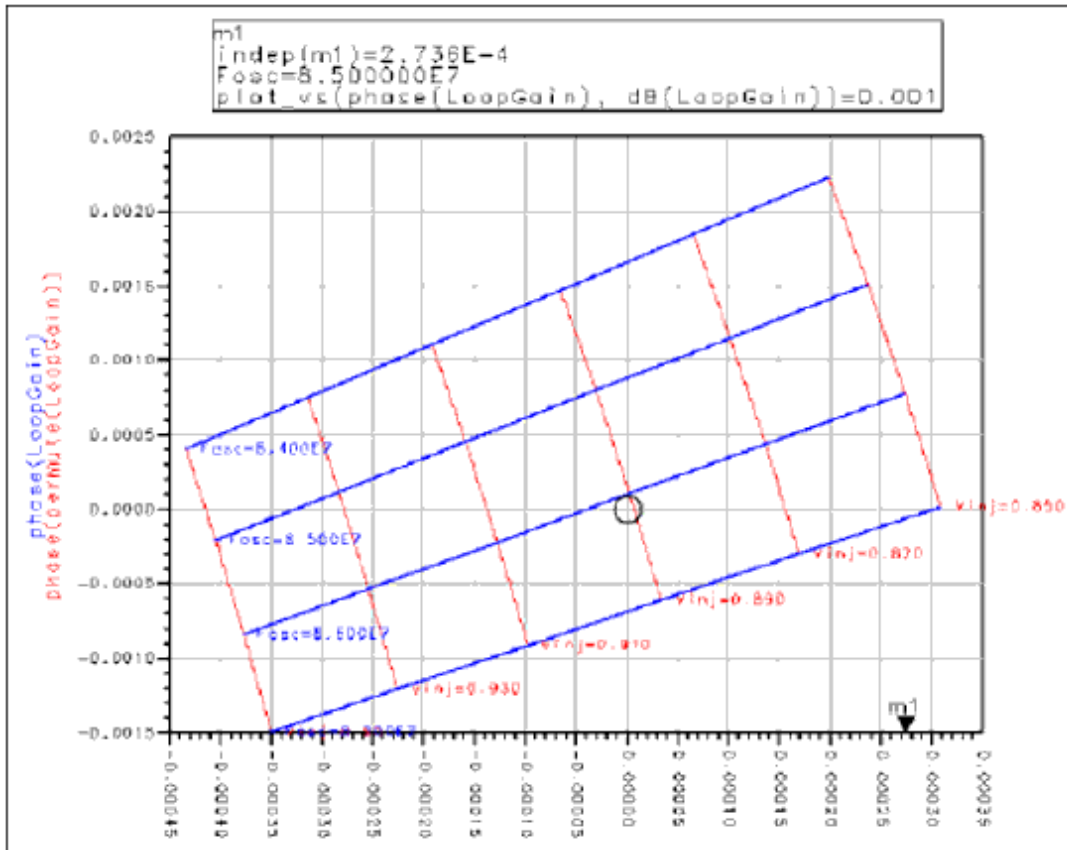
We will start with a fairly coarse sweep. Frequency will be swept from 80 to 90 MHz in 2 MHz, and the injected voltage will be swept from 0.1 to 1.1 V in 0.2 V steps. We are looking for the frequency and voltage that will sustain oscillation: that point at which the loop gain is unity with zero phase shift. In terms of the expressions used in the data display, this is  $dB(LoopGain)=0$  and  $phase(LoopGain)=0$ . A good way of viewing this is shown in the following figure. We plot  $phase(LoopGain)$  versus  $dB(LoopGain)$ , and the point of oscillation will be the (0,0) point on the graph.



#### Phase loop gain vs dB loop gain

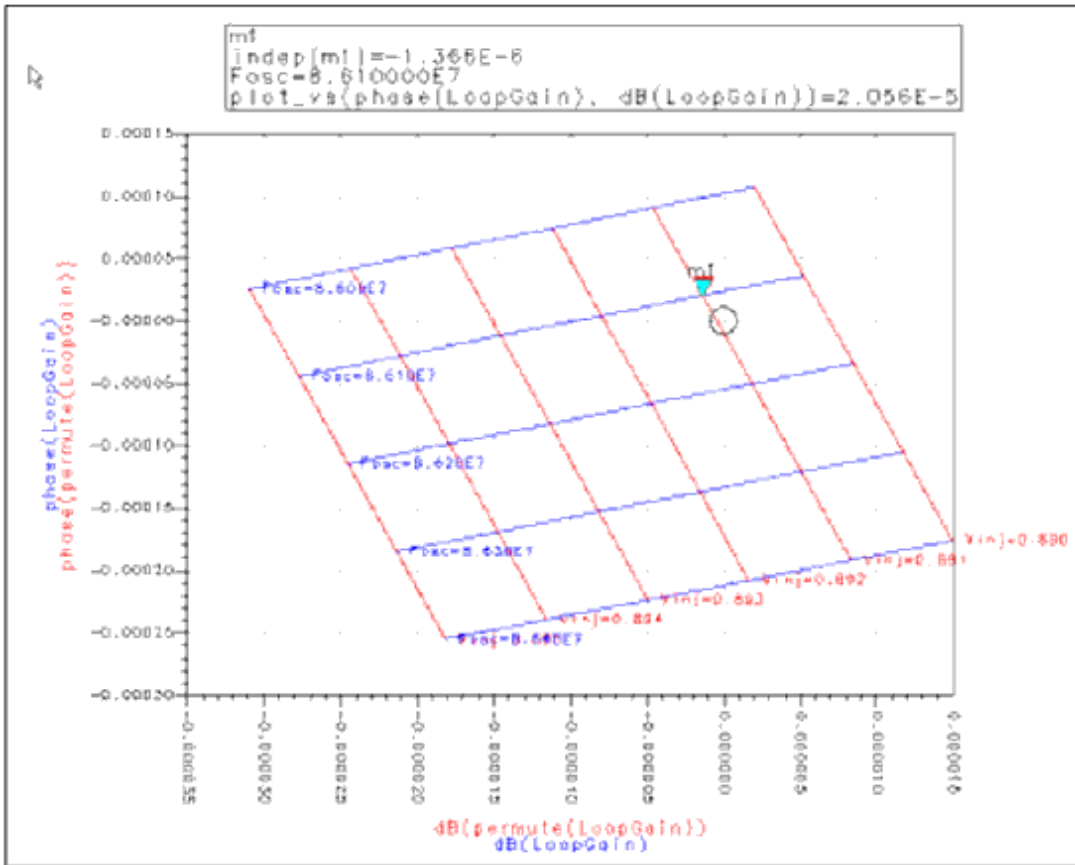
The blue lines are lines of constant frequency and the red lines are lines of constant injected voltage. By using a marker, we can identify which line is which. We can see how the magnitude of the loop gain decreases as the injected voltage increases: the red lines move to the left as the voltage increases.

From this coarse plot, we can narrow down the range of our search for the next step. The frequency range will be swept from 84 to 87 MHz in 1 MHz step, and the injected voltage from 0.85 to 0.95 V in 0.02 V steps. The new results are shown in the following figure. The (0,0) point is circled for clarity.



**Results with lower search range**

We will make one additional refinement on this analysis. The frequency range will be swept from 84.0 to 84.5 MHz in 0.1 MHz steps, and the injected voltage from 0.890 to 0.895 V in 0.001 V steps. These results are shown in the following figure.



### Results with modified frequency and injected voltage

Depending on the nature of the oscillator, more or less sweep refinements might be necessary. If the oscillator appears to be well behaved (plots have a nice grid-like structure), then less refinements may be needed. If the plots are more curvy, or show the possibility of multiple solutions, further refinement might be needed.

### Optimization of Large-Signal Loop Gain

From this last analysis, we have a good estimate of the point at which oscillation will occur. Rather than continuing to manually sweep, we will switch to optimization and let ADS find this point by itself. From the last analysis, we can estimate a bounding box for the oscillation point and restrict the optimization to this range:

$$F_{osc} = 86.1 \text{ MHz opt } \{86.0 \text{ MHz to } 86.5 \text{ MHz}\}$$

$$V_{inj} = 0.89 \text{ opt } \{0.88 \text{ to } 0.90 \}$$

The optimization goals are simple:

$$\text{dB}(\text{LoopGain}) = 0$$

$$\text{phase}(\text{LoopGain}) = 0$$

The circuit used is shown in the following figure. Only the simulation setup is shown; the rest of the circuit is the same as the previous one. A gradient optimization is done using the two goals. The object is to find the frequency and injected voltage at which the oscillation condition is satisfied.

**OPTIM**

Optim  
Optim1  
OptimType=Gradient  
MaxIters=25  
FinalAnalysis="None"  
SaveGoals=yes  
SaveNominal=no  
UseAllGoals=yes

**GOAL**

Goal  
OptimGoal1  
Expr="dB(LoopGain)"  
SimInstanceName="HB2"  
Min=0  
Max=0

**GOAL**

Goal  
OptimGoal2  
Expr="phase(LoopGain)"  
SimInstanceName="HB2"  
Min=0  
Max=0

**HARMONIC BALANCE**

HarmonicBalance  
HB2  
Freq[1]=Fosc  
Order[1]=15

**Var Eqn** VAR  
VAR2  
Zosc=1.1  
Vinj=0.89 opt { 0.88 to 0.90 }  
Fosc=86.1 MHz opt { 86.0 MHz to 86.5 MHz }  
Iinj=2.0\*Vinj/Zosc

**Meas Eqn** MeasEqn  
Meas1  
LoopGain=LoopOut[1]/LoopIn[1]

### Optimization setup

The plot in the next figure shows the optimization results.

| optIter | Optim Goal1 | Optim Goal2 |
|---------|-------------|-------------|
| 0       | 5.243E-6    | 3.680E-5    |
| 7       | .1.936E-8   | 3.447E-9    |

| Fosc     | Vinj     |
|----------|----------|
| 86.1304M | 890.945m |

### Optimization results

The initial error values are shown, and they were fairly good to start with. But seven iterations get the error down below  $10^{-8}$ . The final values are  $F_{osc}=86.1304$  MHz and  $V_{inj}=0.890945$  V.

### Creation of Initial Guess

We run one final simulation with this circuit, setting specifications for VAR, MeasEqn, and Harmonic Balance. The large signal loop gain will be simulated at one point, which includes the values found from optimization. The harmonic balance simulation will save all of its results (the voltage spectra at every node in the circuit) in an output file, using the OutFile parameter. This will be used as initial guess for a full oscillator simulation.

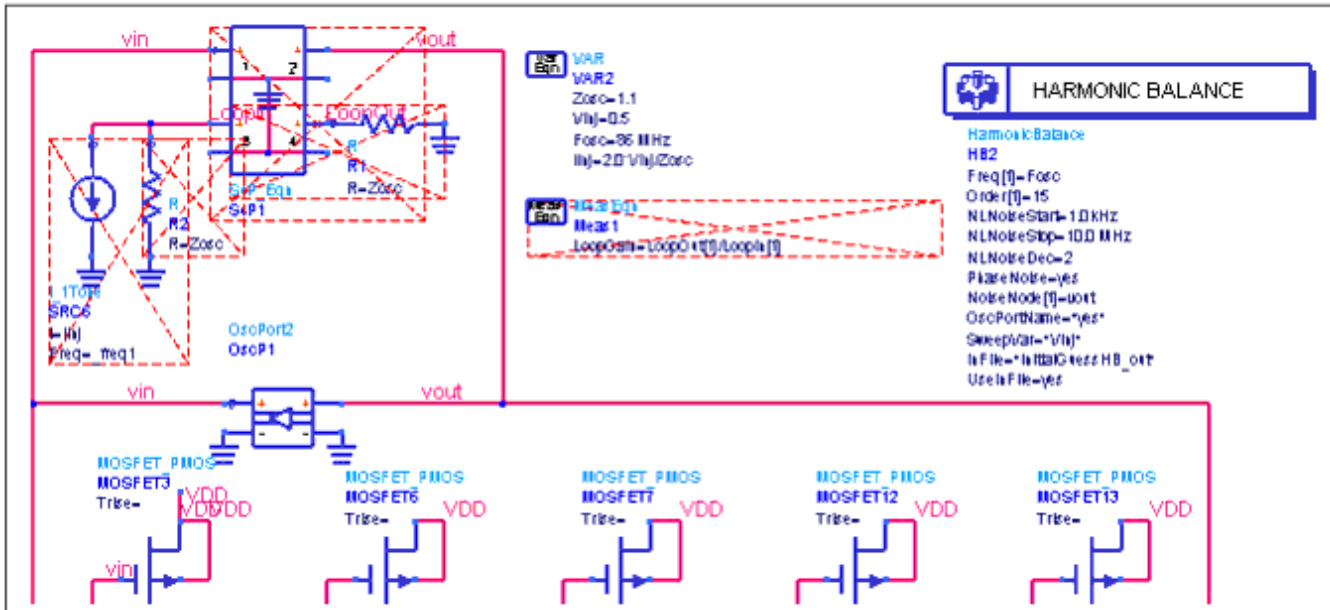
The VAR, MeasEqn, and harmonic balance items are set up as follows:

- **VAR2**  
 $Z_{osc} = 1.1$   
 $V_{inj} = 0/890945$   
 $F_{osc} = 86.1304$  MHz  
 $I_{inj} = Z_{0} * V_{inj} / Z_{osc}$
- **MeasEqn**  
 $LoopGain = LoopOut[1] / LoopIn[1]$
- **HB2**  
 $Fre1[1] = F_{osc}$   
 $Order[1] = 15$   
 $OutFile = fig22\_out$   
 $UseOutFile = yes$

### Final Oscillator Simulation

Now that we have a good initial guess for the oscillator, we can return the automatic oscillator algorithm in harmonic balance. To use this, we must enable the OscPort2 component and disable the S4P and related components that were added to compute the loop gain.

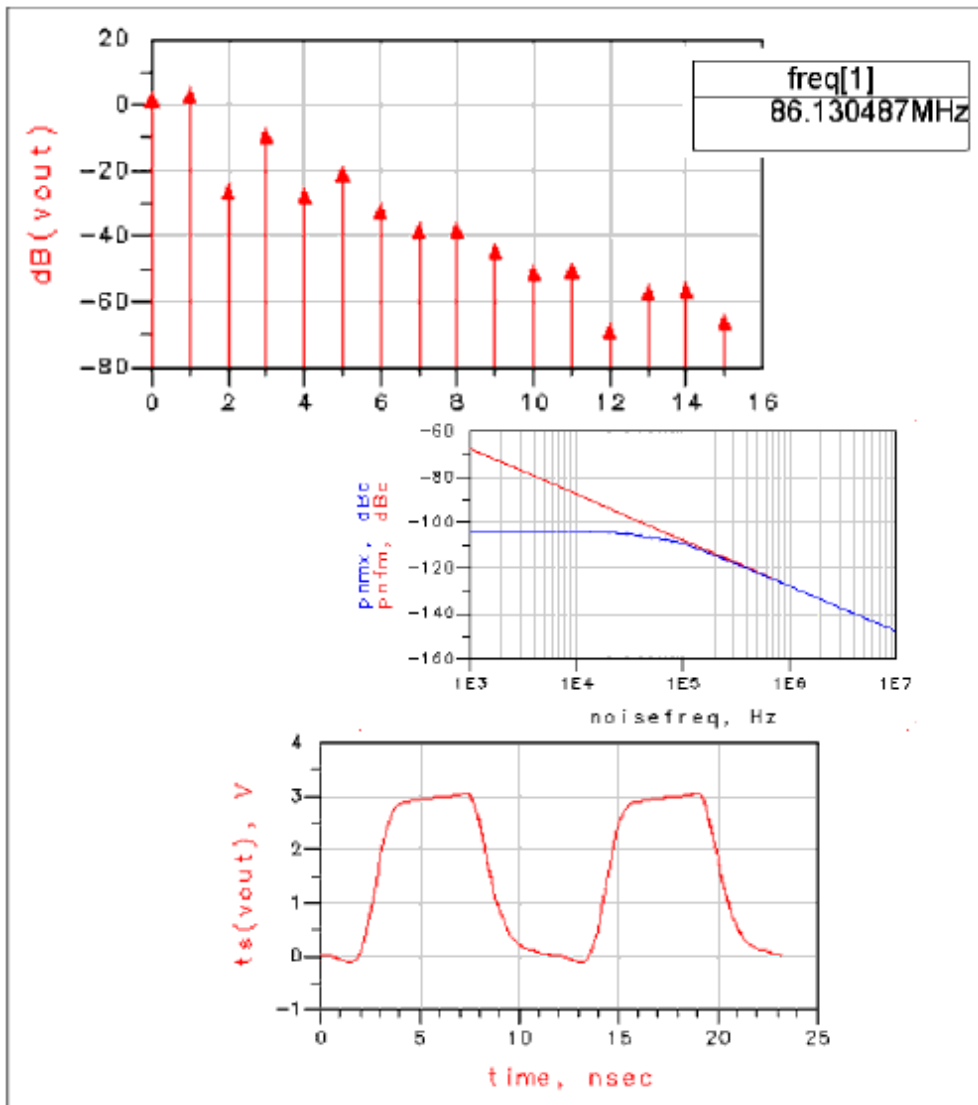
The simulation setup is shown in the following figure. Harmonic balance simulation is done, the oscillator mode is selected and the OscPort name is entered ("yes" means automatically find it). The initial guess is read in using the InFile parameter. It should be set to read the file that was written out in the last step.



#### Setup for final oscillator

The simulation goes quickly, as the simulator has a good initial guess. The initial guess code will complain that some components in the input file don't exist in the current circuit and vice versa, but that doesn't affect its ability to converge. The important nodes (those associated with the nonlinear devices) get the correct initial guess values. The harmonic balance oscillator algorithm refines the frequency and voltages a little, and this time converges. The results are shown in the next figure.





### Results of final oscillation

If we look at the oscillation frequency to many decimal places, we'll see that the loop gain optimization came up with a slightly different value than the oscillator search algorithm found. They are close but vary a little because we did change the circuit a bit.

```
Fosc from optimization:      86.130 415 03 MHz
Fosc input to osc analysis:  86.130 4      MHz
Fosc final answer:          86.130 487 02 MHz
```

### Summary of Simulation Techniques

The preceding shows two methods for solving difficult oscillators in harmonic balance using initial guesses. Initial guess can either be generated from harmonic balance by looking at the large signal loop gain, or from a transient analysis.

If we look at the oscillation frequency to many decimal places, we'll see that the loop gain

optimization came up with a slightly different value than the oscillator search algorithm found. They are close but vary a little because we did make small changes to the circuit.

```
Transient method:
  Fosc from transient:      86.094 601 MHz
Fosc input to osc analysis: 86.1      MHz
Fosc final answer:         86.131 078 MHz
  Loop gain method:
  Fosc from optimization:   86.130 415 MHz
  Fosc input to osc analysis: 86.130 4   MHz
  Fosc final answer:        86.130 487 MHz
```

The harmonic balance oscillator results from the two methods are very similar. The frequencies differ by only 591 Hz, which is a relative difference of  $6.9 \times 10^{-6}$ .

# Harmonic Balance for Mixers

This topic discusses the use of the Harmonic Balance simulator for standard mixer analyses and the small signal option in the Harmonic Balance dialog box that simplifies mixer analyses. It also discusses noise simulations for mixers.

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 sections focusing on setting up noise simulations for mixers describe 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 tab 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 tab in the Harmonic Balance dialog box. With NoiseCons, you can also set parameters to calculate phase noise.

The starting point of this topic assumes you are familiar with the basics of harmonic balance simulation, as described in *Harmonic Balance Basics* (cktsimhb).

For details on harmonic balance for mixers and small-signal mixer analysis, see the following topics:

- [Performing a Basic Mixer Simulation](#) gives the minimum setup for a mixer simulation.
- [Examples of Mixer Simulations](#) 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 Mode Description](#) is a brief explanation of small-signal mixer simulation.

When you are familiar with basic mixer simulation, see the following topics for details on mixer noise simulation:

- [Determining Mixer Noise](#) is an example of how to set up a simulation to calculate noise.
- [Simulating Mixer Noise with NoiseCons](#) illustrates how to perform noise simulations using NoiseCon components in ADS.
- [Small-Signal Noise Simulation](#) is an overview of how nonlinear swept noise is calculated.

## 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 Mixing Order* (cktsimhb).
  - 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* (cktsimhb).

For details about other fields, click *Help* from the dialog box.

## Examples of Mixer Simulations

This section gives detailed setups to perform harmonic balance simulations for:

- [Finding Mixer Output Tones](#)
- [Performing a Small-Signal Simulation of a Mixer](#)
- [Determining Mixer Conversion Gain](#)
- [Determining Mixer Intermodulation Distortion](#)
- [Determining Mixer Noise](#)
- [Simulating Mixer Noise with NoiseCons](#)

### Note

The Harmonic Balance simulation uses the Harmonic Balance Simulator license (sim\_harmonic) which is included with all Circuit Design suites except RF Designer. You must have this license to run Harmonic Balance simulations. You can work with examples described here and installed with the software without the license, but you will not be able to simulate them.

## Finding Mixer Output Tones

The next figure below illustrates an example setup for determining the amplitudes and relationships of mixer output tones.

**Note**

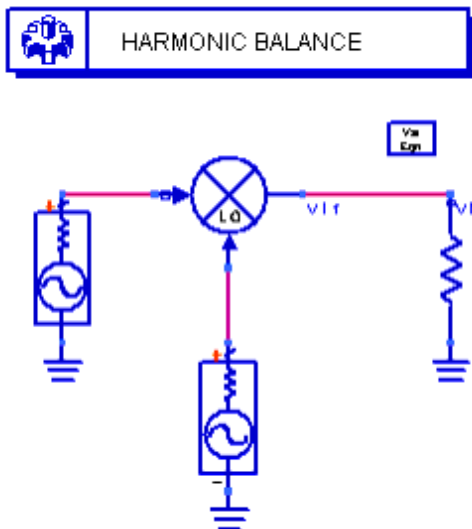
This design, *Mix1*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *Mix1.dds*. Refer also to the ADS examples in */examples/RFIC/Mixers\_wrk*.

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 *dbl1.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.

**Setup for finding output tones**

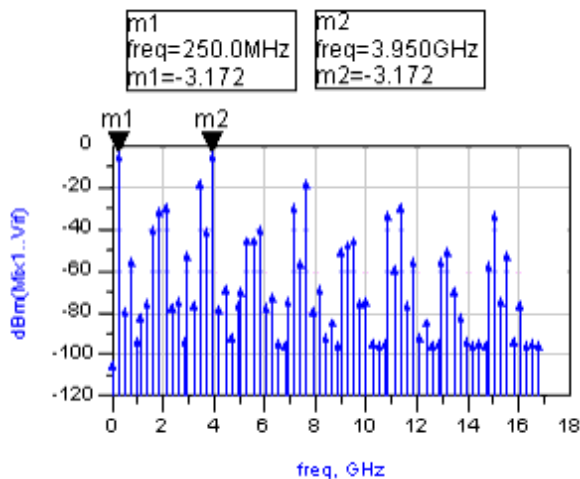
To find the amplitudes and relationships of mixer output tones:

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 = 1850 MHz**
    - **RFfreq = 2100 MHz**
  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 Vif in dBm, then place markers on the two fundamental IF tones, RFfreq - LOfreq (m1) and RFfreq + LOfreq (m2), as shown below:

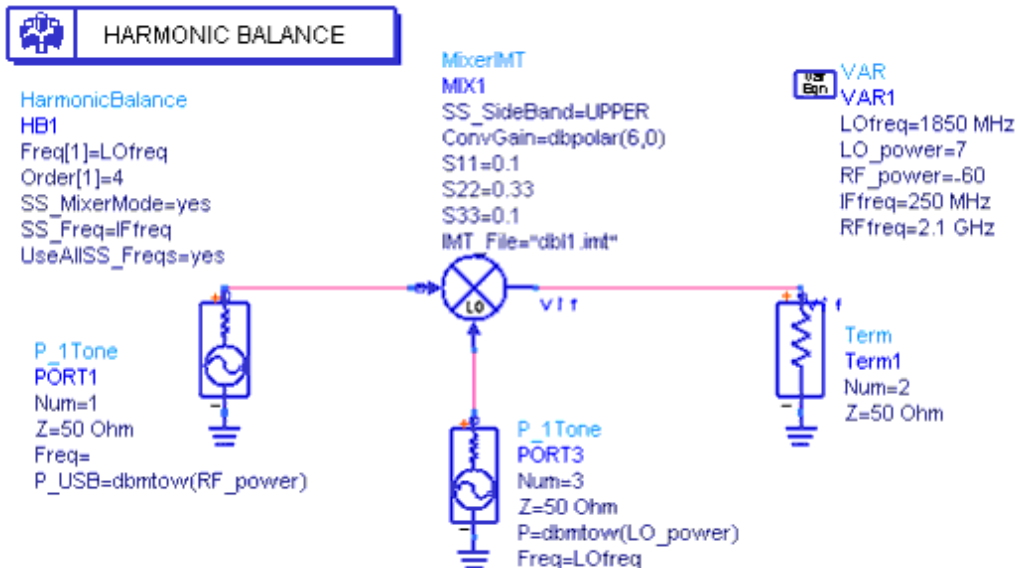


Place more markers to view other frequencies and their magnitudes.

## Performing a Small-Signal Simulation of a Mixer

The following figure, similar to the preceding example, illustrates an example setup for performing a small-signal mixer analysis.

**Note**  
This design, *Mix1\_ssmix*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results of the simulation are in *Mix1\_ssmix.dds*.  
For a detailed example refer to *ADS examples/RFIC/MixerDiffMode\_wrk*.



### 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.

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](#).



#### Note

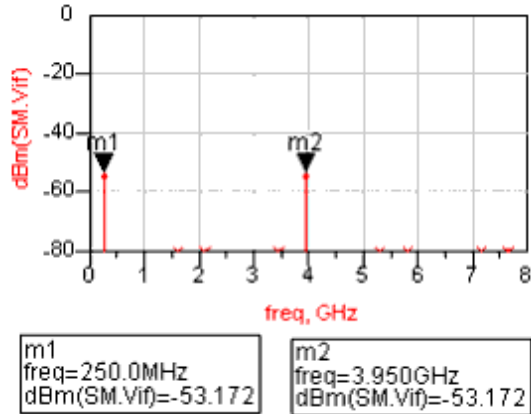
The *Freq* parameter of the P\_1Tone component should be empty, as it will be ignored in this analysis. If you simulate or modify the *Mix1\_ssmix* example shown in the previous figure, be sure to change *Freq = None*.

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 (1850 MHz). The RF input is therefore 2100 MHz.

3. **Simulate.** When the simulation is finished, place a Rectangular plot of  $SM.Vif$  in dBm, as follows:



Markers at 250 MHz ( $f_{IF}$ ) 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).

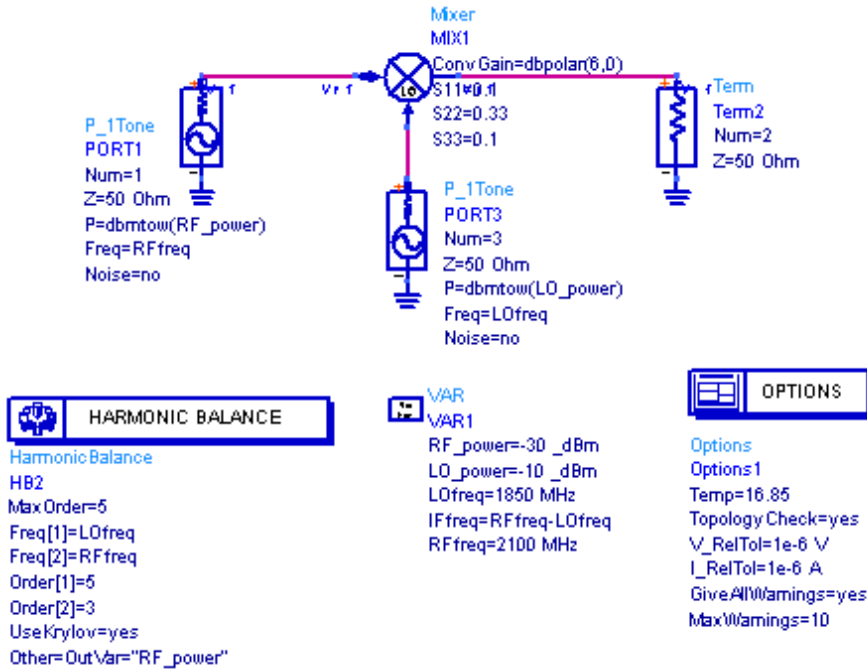
## Determining Mixer Conversion Gain

The following figure illustrates an example setup for determining mixer conversion gain.

**Note**

This design, *Mix1\_convgain*, is in the *Examples* directory under *Tutorial/SimModel\_wrk*. The results are in *Mix1\_convgain.dds*.





### Example determining mixer conversion gain

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](#).
2. Edit the P\_1Tone component at the RF source (PORT1)s:
  - $P = \text{dbmtow}(\text{Power\_RF})$ . Power\_RF will be established by an equation.
  - $\text{Freq} = \text{RFfreq}$
3. Edit the P\_1Tone component at the LO source (PORT2):
  - $P = \text{dbmtow}(0)$
  - $\text{Freq} = \text{LOfreq}$
4. Using the VAR component, enter the following equations (note the use of the underscore before dBm):
  - $\text{RF\_power} = -30 \text{ _dBm}$
  - $\text{LO\_power} = -10 \text{ _dBm}$
  - $\text{LOfreq} = 1850 \text{ MHz}$
  - $\text{IFfreq} = \text{RFfreq} - \text{LOfreq}$
  - $\text{RFfreq} = 2100 \text{ MHz}$

**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**

- $\text{Freq}[1] = \text{LOfreq}$
  - Copy and paste the frequency above, creating  $\text{Freq}[2]$ , the RF frequency. Then edit it so that its frequency is **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, use the Output tab. Refer to *Selectively Saving and Controlling Simulation Data* (cktsim).

7. **Simulate**. When the simulation is finished, do the following:
- Plot  $\text{Mix}(1)$  and  $\text{Mix}(2)$  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 the next figure below, 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  $V_{if}$ , then you select  $V_{if}[7]$ , 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:

$$\mathbf{VIFtone = dBm(mix(Vif,\{1,1\}))}$$

- Write an equation in the Data Display window that expresses conversion gain as the difference between IF power (  $VIFtone$  ) and  $RF\_power$ :

$$\mathbf{MixConvGain = VIFtone - RF\_power[0]}$$

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  $\text{MixConvGain}$  to be optimized. For a discussion of optimization, refer to *Tuning, Optimization, and Statistical Design* (optstat).

The following figure shows the mixing indexes, the equations, and a List plot of  $\text{MixConvGain}$ . This figure, 6.000, is precisely the value assigned to the parameter  $\text{ConvGain}$  in  $\text{MIX1}$ .

| freq     | Vif                  | Vrf             | Mix(1) | Mix(2) |
|----------|----------------------|-----------------|--------|--------|
| 0.000 Hz | 0.758E-18 / 0.000 V  | 0.000 / 0.000 V | 0      | 0      |
| 250 MHz  | 6.00a / 179. V       | 0.000 / 0.000 V | -1     | 1      |
| 500 MHz  | 0.289E-18 / 121. V   | 0.000 / 0.000 V | -2     | 2      |
| 1.35 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 3      | -2     |
| 1.60 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 2      | -1     |
| 1.85 GHz | 0.0134E-48 / 0.000 V | 0.000 / 0.000 V | 1      | 0      |
| 2.10 GHz | 0.000 / 0.000 V      | 11.0m / 0.000 V | 0      | 1      |
| 2.35 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | -1     | 2      |
| 2.60 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | -2     | 3      |
| 3.45 GHz | 0.790E-18 / 104. V   | 0.000 / 0.000 V | 3      | -1     |
| 3.70 GHz | 0.730E-18 / 52.2 V   | 0.000 / 0.000 V | 2      | 0      |
| 3.95 GHz | 20.0m / 214. rV      | 0.000 / 0.000 V | 1      | 1      |
| 4.20 GHz | 0.664E-18 / -4.75 V  | 0.000 / 0.000 V | 0      | 2      |
| 4.45 GHz | 0.203E-18 / -175. V  | 0.000 / 0.000 V | -1     | 3      |
| 5.30 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 4      | -1     |
| 5.55 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 3      | 0      |
| 5.80 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 2      | 1      |
| 6.05 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 1      | 2      |
| 6.30 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 0      | 3      |
| 7.40 GHz | 0.553E-18 / 119. V   | 0.000 / 0.000 V | 4      | 0      |
| 7.65 GHz | 0.598E-18 / -7.73 V  | 0.000 / 0.000 V | 3      | 1      |
| 7.90 GHz | 1.90a / 104. V       | 0.000 / 0.000 V | 2      | 2      |
| 8.15 GHz | 1.47a / 163. V       | 0.000 / 0.000 V | 1      | 3      |
| 9.25 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 5      | 0      |
| 9.50 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 4      | 1      |
| 9.75 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 3      | 2      |
| 10.0 GHz | 0.000 / 0.000 V      | 0.000 / 0.000 V | 2      | 3      |

Eqn VIFtone=dBm(mix(Vif,{1,1}))

Eqn MixConvGain=VIFtone-RF\_power[0]

| freq     | MixConvGain |
|----------|-------------|
| 3.950GHz | 6.000       |

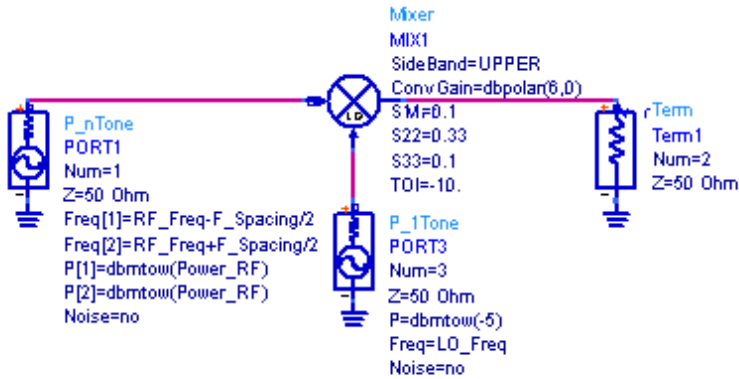
Mixer conversion gain indexes, equations, and plot

## Determining Mixer Intermodulation Distortion

The following figure 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*, is in the *Examples* directory under *Tutorial/SimModels\_wrk*. The results are in *Mix2\_TOI.dds*.

This example is similar to [Performing a Small-Signal Simulation of a Mixer](#), except that it involves three tones as well as equations that determine Freq[2] and Freq[3].



**HARMONIC BALANCE**

Harmonic Balance  
 HB2  
 Max Order=8  
 Freq[1]=LO\_Freq  
 Freq[2]=RF\_Freq+F\_Spacing/2  
 Freq[3]=RF\_Freq-F\_Spacing/2  
 Order[1]=5  
 Order[2]=3  
 Order[3]=3  
 UseKrylov=yes

**OPTIONS**

Options  
 Options1  
 Temp=16.85  
 Topology Check=yes  
 V\_RelTol=1e-6 V  
 I\_RelTol=1e-6 A  
 GiveAllWarnings=yes  
 MaxWarnings=10

**VAR**

VAR1  
 RF\_Freq=2100 MHz  
 LO\_Freq=1850 MHz  
 Power\_RF=-50  
 F\_Spacing=100 kHz

**Mixer intermodulation distortion example**

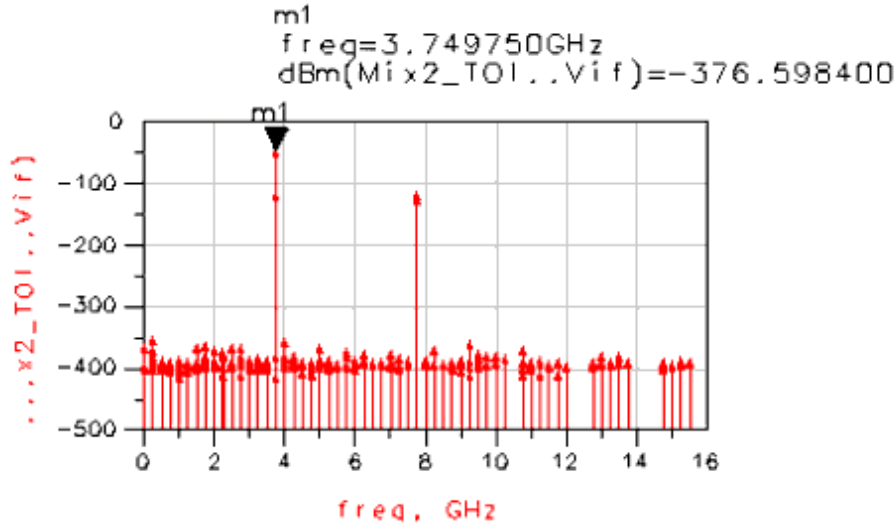
To determine mixer intermodulation distortion:

1. Start with the circuit in the example, [Performing a Small-Signal Simulation of a Mixer](#).
2. Edit the P\_nTone component at the RF source (PORT1) as follows:
  - P[1]= **dbmtow(Power\_RF)**. Power\_RF will be established by an equation.
  - P[2]= **dbmtow(Power\_RF)**

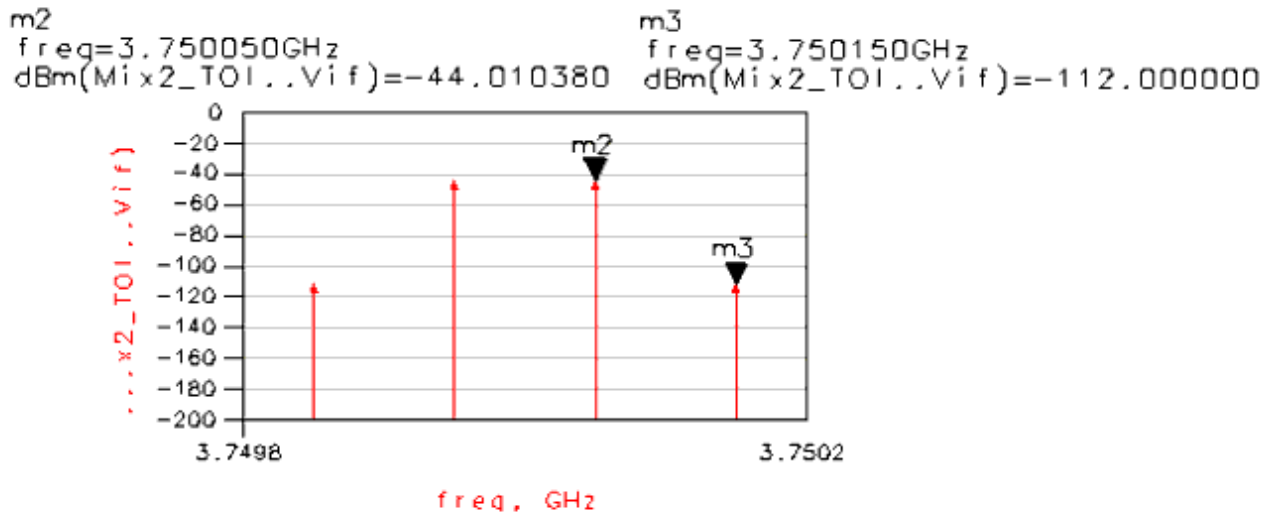
**Note**  
 The indexes [1] and [2] are local to this source component only, and are not related to Freq[1] or Freq[2] in the Harmonic Balance Simulation component.

3. Edit the P\_1Tone component at the LO source (PORT2) as follows:
  - P= **dbmtow(-5)**
  - Freq = **1850 MHz**
4. Enter the following equations in the VAR component:
  - Power\_RF = -50 dBm**
  - RF\_Freq = 2100 MHz**
  - F\_Spacing = 100 kHz**
5. Edit the Harmonic Balance Simulation component, select the **Freq** tab, and edit the following parameters:
  - Maximum order = **8**
  - Frequency = **1850 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 **RF\_Freq+F\_Spacing/2** and its order is **3**.
  - Copy the above to create Freq[3], and edit so that its frequency is **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:
  - Plot Vif, 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 Vif again, this time changing the scale of the x-axis as follows: Select the plot, then choose **Edit > Item Options** (or double-click). When the window opens, click the **Plot Options** tab. Deselect **Auto Scale On/Off**, and make sure that **xAxis** is selected. Set Min = **3.7498e9**, Max = **3.7502e9**, Step = **4e5**, and click **OK**. The following will appear, showing the tones centered around the fundamental IF tone ( $f_{LO} + f_{RF}$ ) at 3.750 GHz:



## Determining Mixer Noise

The next figure below illustrates an example for determining mixer noise. It is based on the example, [Finding Mixer Output Tones](#).

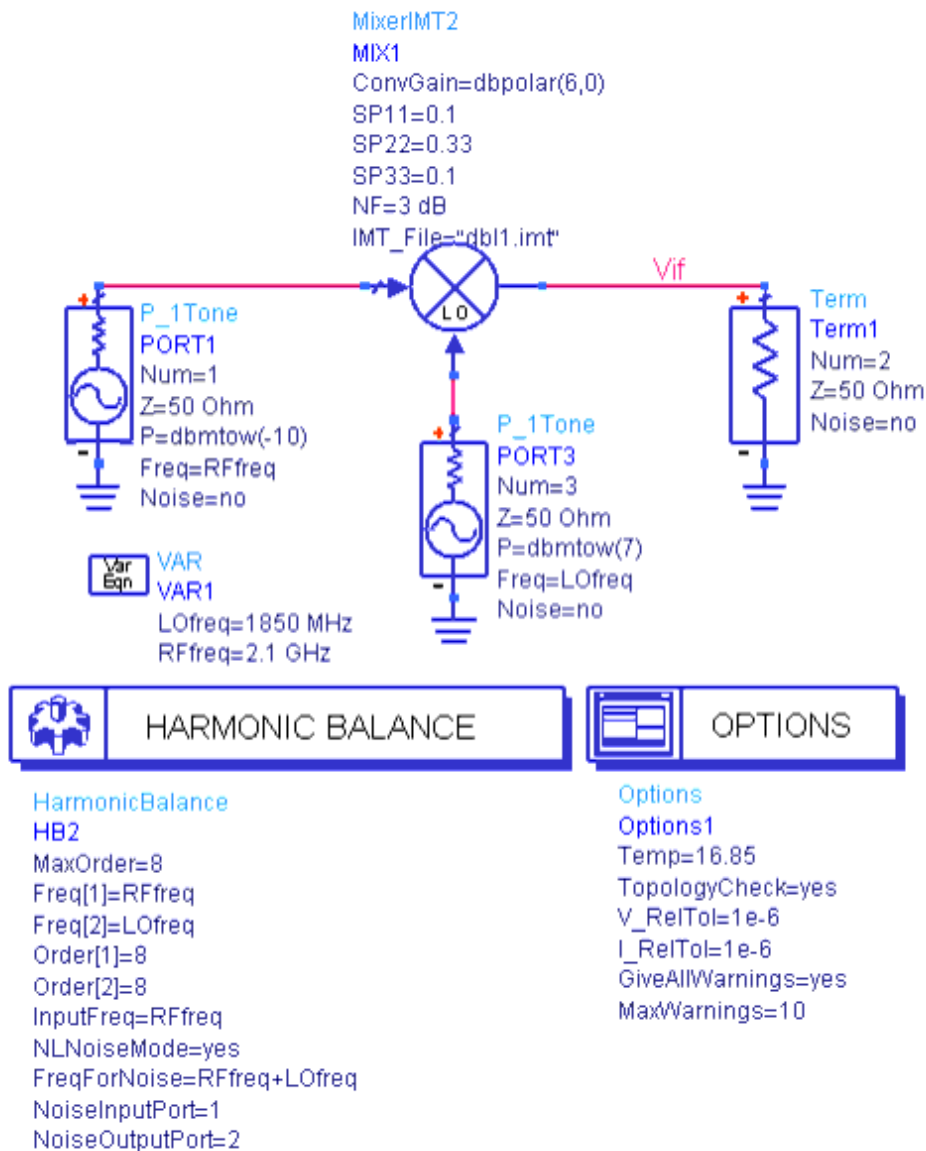
The additional steps in this example configure the simulation for noise analysis. The

MixerIMT2 parameters are the same as those in the previous example, except that NF (double sideband noise figure in dB) 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:

1. Start with [Finding Mixer Output Tones](#).
2. From the **Simulation-HB** 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**



**Determining mixer noise example**

3. Edit the HarmonicBalance simulation component. In the dialog box, select the **Noise** tab, enable **Nonlinear noise**, then click **Noise (1)** to edit the settings in the following fields:
  - Sweep Type = **Single Point**
  - Frequency = **RFreq+LOfreq** ( $f_{RF} + f_{LO}$ ). This is the mixer output frequency.
  - Input frequency = **RFreq** (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**.

**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 figuration.

5. **Simulate**. When the simulation is finished, plot nf(2) and te(2) in dB, as shown below. These are the single sideband noise figure and equivalent noise temperature (in Kelvin) at port 2, respectively.

| noisefreq | nf(2) | te(2)   |
|-----------|-------|---------|
| 3.950 GHz | 6.309 | 949.549 |

The single sideband noise figure of 6.3 dB indicates that the mixer is adding noise based on the MixerIMT2 component setting NF = 3 dB (double sideband).

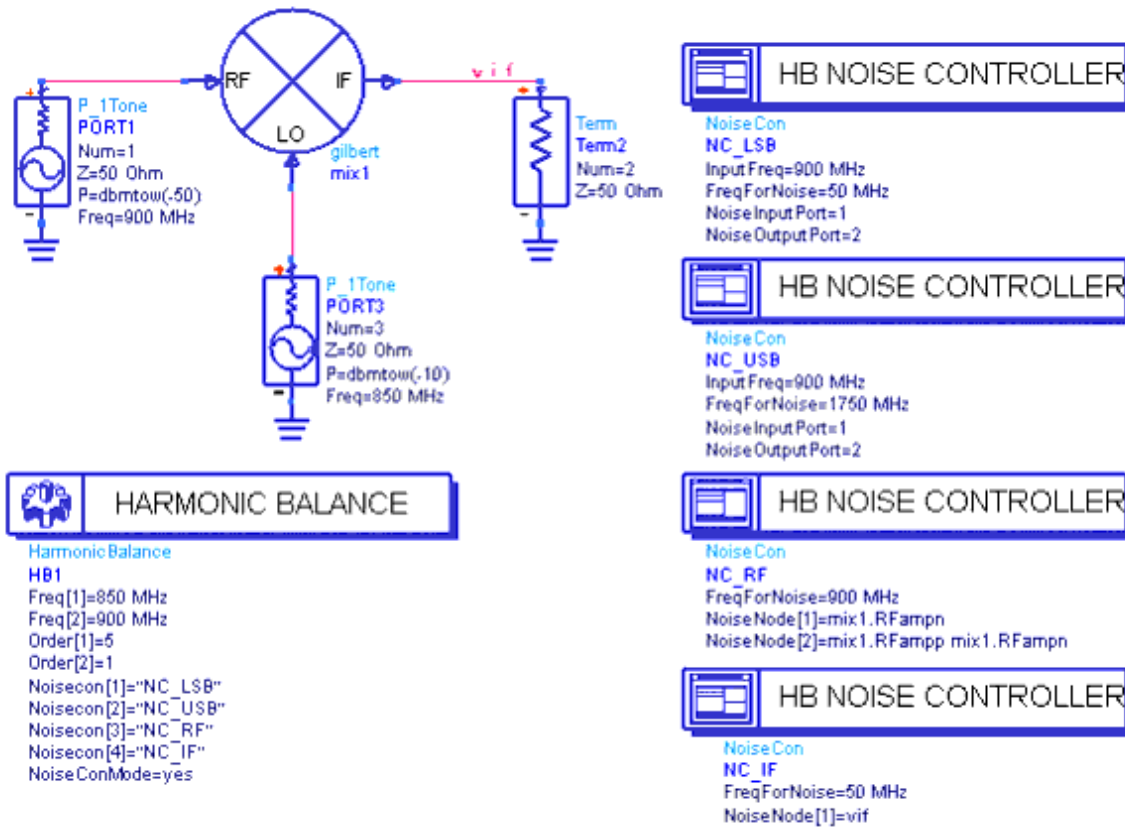
## Simulating Mixer Noise with NoiseCons

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.

The next figure below shows a mixer that will be used to demonstrate the use of NoiseCons components with a mixer.

**Note**

Refer to the ADS workspace *examples/Tutorial/Noisecon\_wrk* for an illustration of how to use NoiseCons for mixer noise simulation. See *mixer* for the design and *mixer.dds* for the results.



Example demonstrating NoiseCons with a mixer

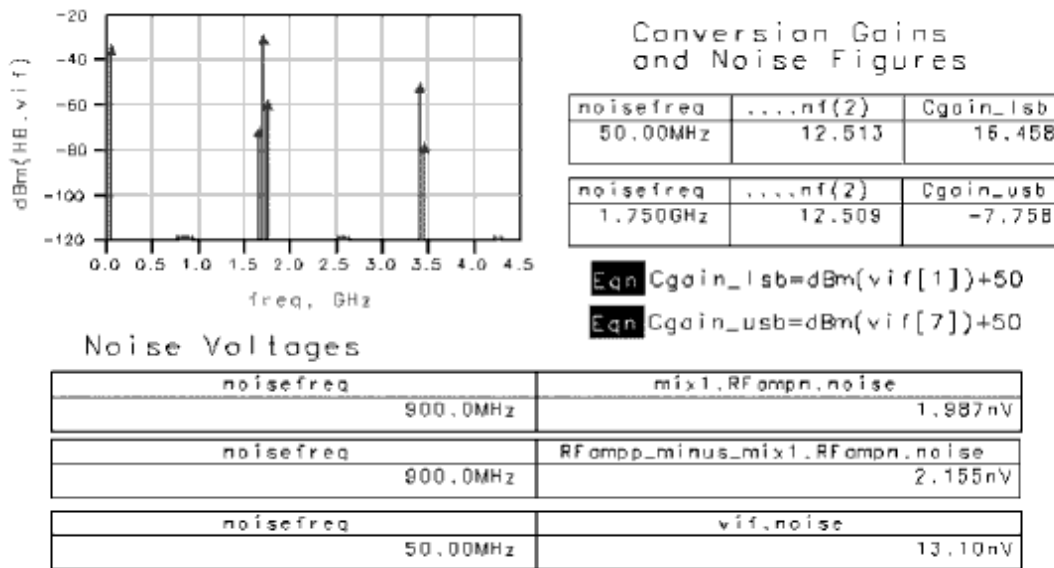
## Mixer Noise Figure

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.





- $NC\_USB.nf(2)$  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.
  - 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.

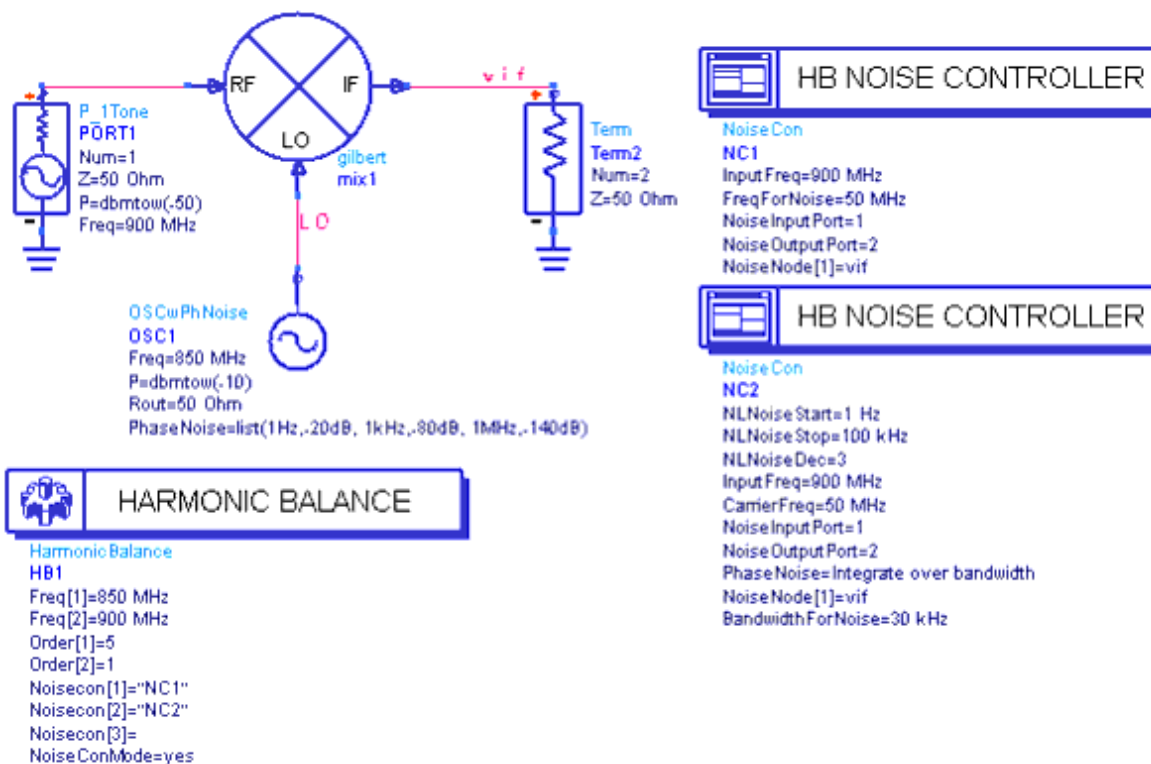
Note the following when viewing the results.

- *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. The figure above, [Determining mixer noise example](#), 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 ADS workspace *examples/Tutorial/Noisecon\_wrk* for an illustration of how to use NoiseCons for mixer noise simulation. See *mixer\_pn* for the design and *mixer\_pn.dds* for the results.



## 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.

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.

$$\text{Eqn } C_{\text{gain\_lsb}} = \text{dBm}(v_{if}[1]) + 50$$

|           |           |
|-----------|-----------|
| noisefreq | NC1.nf(2) |
| 50.00MHz  | 12.513    |

|           |           |
|-----------|-----------|
| noisefreq | NC2.nf(2) |
| 50.00MHz  | 62.865    |

|           |  |
|-----------|--|
| Cgain_lsb |  |
| 16.458    |  |

- $NC1.nf(2)$  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 IEEE standard definition of single sideband noise figure:

$$NF_{ssb} = 10 \log_{10} \left( \frac{(v_n^2(f)/R) + k T_0 (G_1 + G_2 + \dots + G_n)}{k T_0 G_1} \right)$$

where

$k$  is Boltzmann's constant ( $1.389658 \times 10^{-23}$ )

$T_0$  is the IEEE standard temperature for noise figure (290 K)

$G_1$  is the conversion gain of the mixer

$G_2$  is the image conversion gain of the mixer

$G_3, \dots, G_n$  are conversion gains of higher order mixing products

$R$  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  $f_{IF}$ :

$$NF_{ssb} = 10 \log_{10} \left( \frac{\int_{(f_{IF}-B/2)}^{(f_{IF}+B/2)} (v_n^2(f)df)/R + k T_0 B (G_1 + G_2 + \dots + G_n)}{k T_0 B G_1} \right)$$

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  $f_{IF} = |f_{RF} - f_{LO}|$ .

The single sideband noise figure 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 also computes the double sideband noise figure using the following equation:

$$NFdsb = 10 \log_{10} \left( \frac{(v_n^2(f)/R) + k T_0 (G_1 + G_2 + \dots + G_n)}{k T_0 (G_1 + G_2 + \dots + G_n)} \right)$$

This is the value that would be computed by a noise figure meter. It is saved in the dataset as *NFdsb*.

## Small-Signal Mode Description

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.

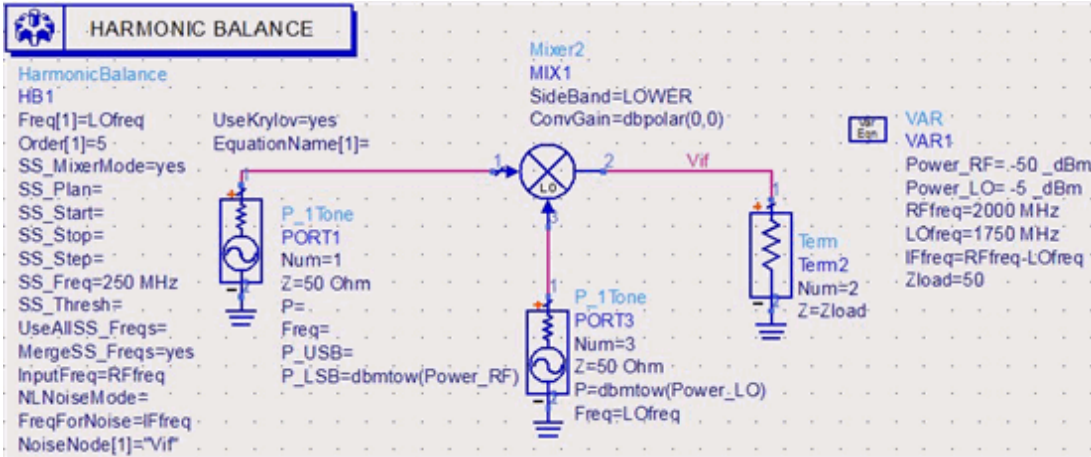
All the time-varying nonlinearities that result from the presence of large signals are simulated, but with the assumption that the small signal does not change these nonlinearities. For example, a mixer will appear to be perfectly linear with respect to the amplitude of the small-signal input, but all 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.

Small-Signal analysis operates on the premise that some tones are large-signal and some tones are small-signal. The small-signal tones are specified relative to a large-signal tone, not in an absolute sense.

Let's consider an example of a simple downconverting mixer



Note that only one large-signal tone exists in the HB controller, `LOfreq=1750 MHz`. The `SS_Freq` is set to 250 MHz, so this means that the actual RF frequencies are:  $1750 \pm 250 \text{ MHz} = 2000 \text{ MHz}, 1500 \text{ MHz}$ .

When either of these frequencies are mixed with the LO frequency, 1750 MHz, the result is 250 MHz, the IF output. Therefore, it is helpful to think of the value of `SS_Freq` to be the IF frequency.

Note that it is not required to specify the frequency, `Freq`, on the `P_1Tone` source, since there is no large-signal tone on that source. It is only necessary to specify the upper and/or lower sideband, `P_USB` and/or `P_LSB`.

## 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 Simulation Description* (cktsimhb).) 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)* dialog box. 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 in the *Noise (1)* dialog box) 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

Advanced Design System 2011.01 - Harmonic Balance Simulation  
of the swept nonlinear noise analysis is always *noisefreq*.



# Transient Assisted Harmonic Balance

This topic describes the automated transient assisted harmonic balance simulation (TAHB). For a harmonic balance simulation, an initial guess of the solution is needed. For circuits that are highly nonlinear and contain sharp-edged waveforms (such as dividers), a transient simulation often provides a good initial guess for the starting point of harmonic balance. In a TAHB simulation, a transient is performed to generate the initial guess for the harmonic balance. Then an HB simulation is performed using the transient initial guess. The simulator automatically generates the transient initial guess when the *Enable* check box is selected.

It is strongly recommended to use this simulation technique for divider circuits and others with sharp edged or pulse-like waveforms. We suggest that you use the TAHB *Auto* mode, which is the default setting, for optimal performance. The simulator will turn on TAHB automatically if the circuit involves a divider. The user can choose the TAHB *On* mode if the circuit does not involve a divider, yet sharp edged or pulse-like waveforms are expected.

If you are not familiar with the harmonic balance simulator, see *Harmonic Balance Basics* (cktsimhb).

The following topics describe details about automated transient assisted harmonic balance:

- [Setting Additional Transient Parameters](#) describes how to set parameters for automated TAHB.
- [Using a One-Tone Transient for a Multi-Tone Harmonic Balance](#) describes how to instruct the transient simulator to perform a one-tone simulation.
- [Using Sweeps and Optimization Simulations](#) describes how to use sweeps and optimization simulations with automated TAHB.
- [Outputting the Transient Data to the Dataset](#) describes how to control transient data output to the dataset.

For details about each individual TAHB parameter, see *Setting Up the Initial Guess* (cktsimhb).

## Setting Additional Transient Parameters

The transient simulator uses intelligent defaults and determines a steady state solution for the initial guess in harmonic balance. For the automated TAHB, you are not required to set any transient parameters. However, you may set the transient parameters by choosing the TAHB *On* mode if desired.

The first parameter to set is *StopTime*. The default for *StopTime* is 100 cycles of the commensurate frequency. In a single tone analysis, this is simply the value of *Freq* [1]. This parameter determines how long transient should run when generating the initial guess. If the transient simulator detects steady state before it reaches the *StopTime*, then

it will end at that point (earlier than the *StopTime* ), then proceed with the harmonic balance simulation. When the transient does not find a steady state solution, then Harmonic Balance uses the last transient data point as an initial guess.

The next parameter to set is *MaxTimeStep*. The default for *MaxTimeStep* is  $1/(2*2*Maximum\ Frequency)$ . In a single tone transient analysis, the maximum frequency is  $Freq[1]*Order[1]$ . Adjust this parameter if a smaller or larger time step is required.

The simulator displays the values that it determined for *StopTime* and *MaxTimeStep* in the status window.

The parameter *Min Detect Steady State Time* sets the earliest point in time that the transient simulator starts checking for steady state conditions. The default is two periods of the fundamental frequency for autonomous circuits, and 10 periods of the fundamental frequency for non-autonomous circuits such as an oscillator. The units for this parameter are in seconds. If your circuit exhibits a large amount of over/undershoot, then this needs to be larger than the default so that the detector will begin to check for steady state after some of the initial transients have settled.

The parameter *IV\_ReITol* is the transient current and voltage relative tolerance. The default value is 1e-3. This is a relative tolerance for transient only. If relative tolerances are set (on Options controller) and the *IV\_ReITol* is given, then it will override the relative tolerances for the Simulator Option's relative tolerances for the transient simulation only. This parameter simply allows for having two sets of relative tolerances, one for transient and the other for harmonic balance.

The *Transient Other* parameter makes it possible to set transient parameters that are not found in the TAHB tab. For example, you can set the following transient convolution parameter *ImpMaxFreq* = 10 GHz using the *Transient Other* parameter. To see all of the transient parameters, please see the *Transient/Convolution Simulation* documentation.

When *TAHB\_Enable* is set to *Auto*, a user-supplied initial guess will be honored and TAHB will be turned off by the simulator. When TAHB is set to *On*, you will not be able to supply an initial guess file. For more information, please see *Transient Assisted Harmonic Balance - TAHB* (adshbapp) in the *Guide to Harmonic Balance Simulation in ADS* (adshbapp).

## Using a One-Tone Transient for a Multi-Tone Harmonic Balance

The *Use only Freq[1] for transient* parameter instructs the transient simulator to perform a one-tone simulation, and to use only the value of *Freq[1]* parameter set for the Harmonic Balance simulation for determining the *StopTime* and *MaxTimeStep*. This parameter is enabled by default. If there are multiple frequencies set for the HB simulation and this parameter is enabled, then sources in the circuit at the other frequencies will be turned off for the transient portion only. It is recommended to perform a one-tone transient simulation for a multi-tone HB simulation. Also, *Freq[1]* should be set to the frequency of the most nonlinear tone.

## Using Sweeps and Optimization Simulations

Statistical simulations such as optimization, yield, Monte Carlo, DOE, and yield-optimization are not supported with automated TAHB. The simulation will terminate prematurely if such simulation types have TAHB set to *On*. If TAHB is set to *Auto*, the simulator will proceed without using TAHB.

Swept simulations are supported with automated TAHB. In ADS, there are two ways to sweep a parameter - using the Sweep tab on the Harmonic Balance controller or using the Parameter Sweep controller. When sweeping a parameter on the Sweep tab, a transient simulation is done for the harmonic balance simulation of the first sweep point only. When sweeping a parameter on the Parameter Sweep controller, a transient simulation is done for the first sweep point only, unless the *Restart* parameter is enabled on the Harmonic Balance controller's Initial Guess tab. When *Restart* is enabled, a transient simulation is done at each sweep point which generates a new transient initial guess for each sweep point of the harmonic balance simulation.

## Outputting the Transient Data to the Dataset

The data from the transient simulation can be output to the dataset by enabling the *Output transient data to dataset* parameter. By default, this parameter is disabled and the transient simulation results are not in the dataset. Recall that the transient was used only to generate the initial guess for harmonic balance. If the transient results are desired in the dataset, then be sure to enable this parameter by checking the box. Proceed with caution for very large circuits since the dataset can grow to be very large if both transient and harmonic balance results are included.

# Harmonic Balance Assisted Harmonic Balance

This topic describes the automated harmonic balance assisted harmonic balance simulation (HBAHB). Using this method when performing a multi-tone harmonic balance simulation allows the simulator to decide automatically which tones to use in generating the final HB solution. For example, instead of simulating a three-tone circuit directly, it is possible to simulate a one-tone, then use the result as an initial guess for a two-tone, and finally use that result as an initial guess for the third tone. HBAHB is both fast and robust, particularly for large multi-tone simulations of mixer circuits to determine gain and IP3.

HBAHB applies only to multi-tone simulations. By default, this feature is enabled within the HB simulator. The simulator determines the optimal sequence of tones to simulate. Depending upon various conditions, the simulator may decide to simulate a three-tone HB simulation with only the first tone, then use that result as an initial guess directly for a three-tone.

If you are not familiar with the harmonic balance simulator, see *Harmonic Balance Basics* (cktsimhb).

The following topics describe details about automated HBAHB:

- [Modes of HBAHB Operation](#)
- [HBAHB, Parameter Sweeps, and Noise](#)
- [HBAHB and TAHB](#)
- [HBAHB and Non-Convergence](#)

## Modes of HBAHB Operation

There are three modes for the HBAHB - *Auto*, *On*, and *Off*. The default mode is *Auto*.

To set the HBAHB mode:

1. Place the HB controller on the schematic and open its setup dialog box.
2. On the Initial Guess tab, in the HBAHB section, select the mode.
  - In the *Auto* mode, the simulator decides whether or not to perform the HBAHB, and the optimal sequence of tones to simulate.
  - In the *On* mode, the HBAHB always performs a simulation at each tone.
  - In the *Off* mode, the HBAHB is not performed at all. When the HBAHB is off, a standard multi-tone HB simulation is performed.

For additional setup information, see *Setting Up the Initial Guess* (cktsimhb).

When selecting Solver parameters for a harmonic balance simulation with the HBAHB set to *Auto*, it is recommended that you set *Convergence Mode* to *Auto* and *Solver Type* to *Auto* (see *Selecting a Harmonic Balance Solver Technique* (cktsimhb)). These settings

ensure that simulation of the intermediate tones is both fast and robust, and that the optimal solver is selected when simulating the different tones.

## HBAHB, Parameter Sweeps, and Noise

When performing a swept simulation, HBAHB occurs only for the first, or innermost, sweep point. When simulating a noise analyses, the HBAHB occurs prior to the noise. This means that the noise analyses is not performed at the intermediate tones.

## HBAHB and TAHB

HBAHB will not be performed if TAHB takes place. The HB analysis will read in the transient solution as the initial guess.

If you provide an initial guess using *Initial Guess* parameters *InFile* and *UseInFile*, HBAHB will not be performed so long as the supplied initial guess file exists. If the file does not exist, the simulator will decide whether or not to perform HBAHB. Details about setting parameters for Initial Guess, HBAHB, and TAHB are located in *Setting Up the Initial Guess* (cktsimhb).

## HBAHB and Non-Convergence

At any point of time during the HBAHB analysis, if non-convergence occurs, the simulation will stop HBAHB and resort to a standard multi-tone HB analyses. A new DC initial guess will also be recomputed. If TAHB was enabled, then the transient initial guess will be restored.