



**February 2011**  
**Guide to Harmonic Balance Simulation in ADS**

**© 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).

About Harmonic Balance Simulation . . . . .	7
Overview of Harmonic Balance . . . . .	7
Simulation Setup . . . . .	9
Setting Frequency . . . . .	9
Setting Order and MaxOrder . . . . .	9
Initial Guess Selection . . . . .	11
Solving Convergence Problems . . . . .	13
Convergence Mode . . . . .	13
Solver Type . . . . .	13
Setting Status Level and Understanding Output in the Status Window . . . . .	15
Parameter Access . . . . .	17
Circuit Operation and Verification with Transient Simulation . . . . .	18
Harmonic Balance Controller Setup . . . . .	19
Newton Solver Issues . . . . .	22
Linear Solver Issues . . . . .	22
Sweeps as Convergence Tools . . . . .	24
Transient Assisted Harmonic Balance - TAHB . . . . .	27
Changing the DC Convergence Algorithm . . . . .	35
Device Models . . . . .	36
Fourier Truncation Error . . . . .	36
Harmonic Balance Assisted Harmonic Balance . . . . .	38
Changing the Tolerances . . . . .	39
ADS Dialog Boxes . . . . .	43
Additional Parameters . . . . .	49
Convergence Mode . . . . .	49
Krylov Solver . . . . .	49
Arc Length Continuation . . . . .	50
Memory Requirements . . . . .	51
Parameter Index . . . . .	53
Harmonic Balance Background . . . . .	55
How the Harmonic Balance Simulator Operates . . . . .	57
Newton's Method . . . . .	58

# About Harmonic Balance Simulation

Harmonic balance is a highly accurate frequency-domain analysis technique for obtaining the steady state solution of nonlinear circuits and systems. It is usually the method of choice for simulating analog RF and microwave problems that are most naturally handled in the frequency domain. Once the steady state solution is calculated, the harmonic balance simulator can be used to do the following.

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

The harmonic balance method assumes that the input stimulus consists of a few steady-state sinusoids. Therefore the solution is a sum of steady state sinusoids that includes the input frequencies in addition to any significant harmonics or mixing terms.

This document provides details and instructions on setting up harmonic balance simulations. It also includes troubleshooting techniques for nonconvergent circuits. It does not cover oscillators, small-signal, or noise simulations.

## Overview of Harmonic Balance

In harmonic balance, the objective is to compute the steady state solution of a nonlinear circuit. In the simulator, the circuit is represented as a system of  $N$  nonlinear ordinary differential equations, where  $N$  represents the size of the circuit (number of nodes and branch currents). The sources and the solution waveforms (all node voltages and branch currents) are approximated by truncated Fourier series. Therefore, a successful simulation will yield the Fourier coefficients of the solution waveforms.

A circuit with a single input source will require a single tone harmonic balance simulation with a solution waveform (e.g., the node voltage  $v(t)$ ) approximated as follows:

$$v(t) = \text{Real} \left\{ \sum_{k=0}^K V_k e^{j2\pi kft} \right\}$$

where  $f$  is the fundamental frequency of the source, the  $V_k$ 's are the complex Fourier coefficients that the harmonic balance analysis computes, and  $K$  is the level of truncation (number of harmonics) called Order. For details on setting the order, refer to *Setting Order and MaxOrder* (adshbapp).

A circuit with multiple input sources will require a multitone simulation. In this case, the steady state solution waveforms are approximated with a multidimensional truncated Fourier series as follows:

$$v(t) = \text{Real} \left\{ \sum_{k_1=0}^{K_1} \sum_{k_2=0}^{K_2} \dots \sum_{k_n=0}^{K_n} V_{k_1, k_2, \dots, k_n} e^{j2\pi(k_1 f_1 + \dots + k_n f_n)t} \right\}$$

where  $n$  is the number of tones (sources),  $f_{1\dots n}$  are the fundamental frequencies of each source, and  $K_{1\dots n}$  are the number of harmonics for each tone. The number of mixed terms that occur with multiple tones in a circuit is controlled by the MaxOrder setting. For details on setting MaxOrder, refer to *Setting Order and MaxOrder* (adshbapp).

The truncated Fourier series representation of the solution transforms the system of  $N$  nonlinear differential equations into a system of  $N*M$  nonlinear algebraic equations in the frequency domain, where  $M$  is the total number of frequencies including the fundamentals, their harmonics, and the mixing terms. This system of nonlinear algebraic equations is solved for the Fourier coefficients of the solution via Newton's Method. This method is the outer solver of the HB simulator (also referred to as the nonlinear solver). Newton's method iterates successively from an initial guess to arrive at the solution.

The system of nonlinear algebraic equations represents a statement of Kirchhoff's Current Law (KCL) in the frequency domain. According to KCL, the sum of the currents entering a node must equal the sum of the currents leaving that node. The amount by which the KCL is violated at each iteration of Newton's method is known as the KCL residual. Newton's method (as well as Harmonic Balance) achieves convergence when the KCL residual is driven to a small value.

Newton's method generates a matrix problem (linear system of equations) at each iteration. This matrix is known as the Jacobian. An inner solver in harmonic balance (also referred to as the linear solver) is used to factor the Jacobian matrix.



# Simulation Setup

There are three main parameters to set when doing an HB simulation: Frequency, Order, MaxOrder. Additionally, two simulation setup parameters will be determined automatically (and in an optimal manner) by the simulator. These are Convergence mode and Solver. If convergence is achieved and accurate results are obtained, then you don't need to go further. If the circuit does not converge, see *Solving Convergence Problems* (adshbapp).

## Setting Frequency

The **Frequency** parameter is found on the HB controller's Freq tab. It appears as Freq[i] on schematic, where  $i=1, \dots, \text{number of tones (sources)}$  in the circuit. For a single tone HB simulation, set the Frequency to the fundamental frequency of the source used in the circuit. For example, in a circuit with input source at 850 MHz, set Freq[1]=850 MHz.

When doing a multitone simulation, additional Frequencies need to be set on the controller corresponding to the fundamental frequencies of the additional sources. It is strongly recommended to set Freq[1] to the most nonlinear tone. The most nonlinear tone is typically the one with the largest power. For example, consider a two tone HB analysis to determine mixer conversion gain with an LO source at 1850 MHz, and an RF source at 2.1 GHz. Since the LO is the more nonlinear tone, it should be set to be the first fundamental, i.e., Freq[1]=1850 MHz, while the RF should be set to Freq[2]=2.1 GHz. Next consider a mixer intermodulation distortion analysis (same LO at 1850 MHz and RF at 2100 MHz). In this case, use a VAR component to define FrqSpacing=100k, and set the HB controller with Freq[1]=LO, Freq[2]=RF+FrqSpacing/2, Freq[3]=RF-FrqSpacing/2. An example of these circuits and simulations can be seen in *Harmonic Balance for Mixers* (cktsimhb).

If the frequency of the input source is not the fundamental or a related harmonic of a Frequency parameter on the controller, then the frequency of the input source is not used in computing the steady state solution. For example, in a circuit with three sources (1 GHz, 900 MHz, and 940 MHz) in which only two of the three are specified on the HB controller (1 GHz and 900 MHz), the third source is turned off. When this occurs, the following warning message is generated:

```
Warning detected by HPEESOF5SIM during HB analysis `HB1'. For source `SRC1',
(1xfreq[3])=9.4e+08 is 4e+07 Hz away from the closest analysis frequency at
9e+08. The maximum frequency difference for analysis time step is 900 Hz.
This spectral component is turned off for this simulation.
```

## Setting Order and MaxOrder

The **Order** parameter is found on the Freq tab, and it determines the number of harmonics used in the truncated Fourier series representation of the HB solution. The Order and Frequency parameters are set at the same time. The default value for Order is 3. For a single tone simulation, set the Order to the desired level of Fourier series truncation. The Order needs to be sufficiently large so that the HB simulator can compute

its solution waveforms to an adequate degree of accuracy. For example, in the circuit with input source at 850 MHz and Order set to 3, the following three harmonics will be used in HB: 850 MHz, 1700 MHz, and 2550 MHz. However, three harmonics are sufficient only for mostly linear circuits generating sinusoidal-like signals. For mildly nonlinear circuits, the Order should be set to 7 or more. Highly nonlinear circuits with waveforms containing sharp edges and spikes will require many more harmonics (sometimes in the hundreds).

For multitone simulations, the Order needs to be specified for each tone. It is recommended to use a higher Order for the more nonlinear tones. For example, in the above mixer example, the Order for the LO tone should be at least 7, while the RF Order can be left at 3.

The parameter **Maximum mixing order** (MaxOrder on schematic), also found on the Freq tab, determines how many mixing products are to be included in a multitone simulation. A mixing term, or mixing product, is a combination of two or more fundamentals or their successive harmonics. Mixing products will occur when there are multiple sources in a circuit. Since the number of mixing terms can grow very large, it is limited in ADS by the following:

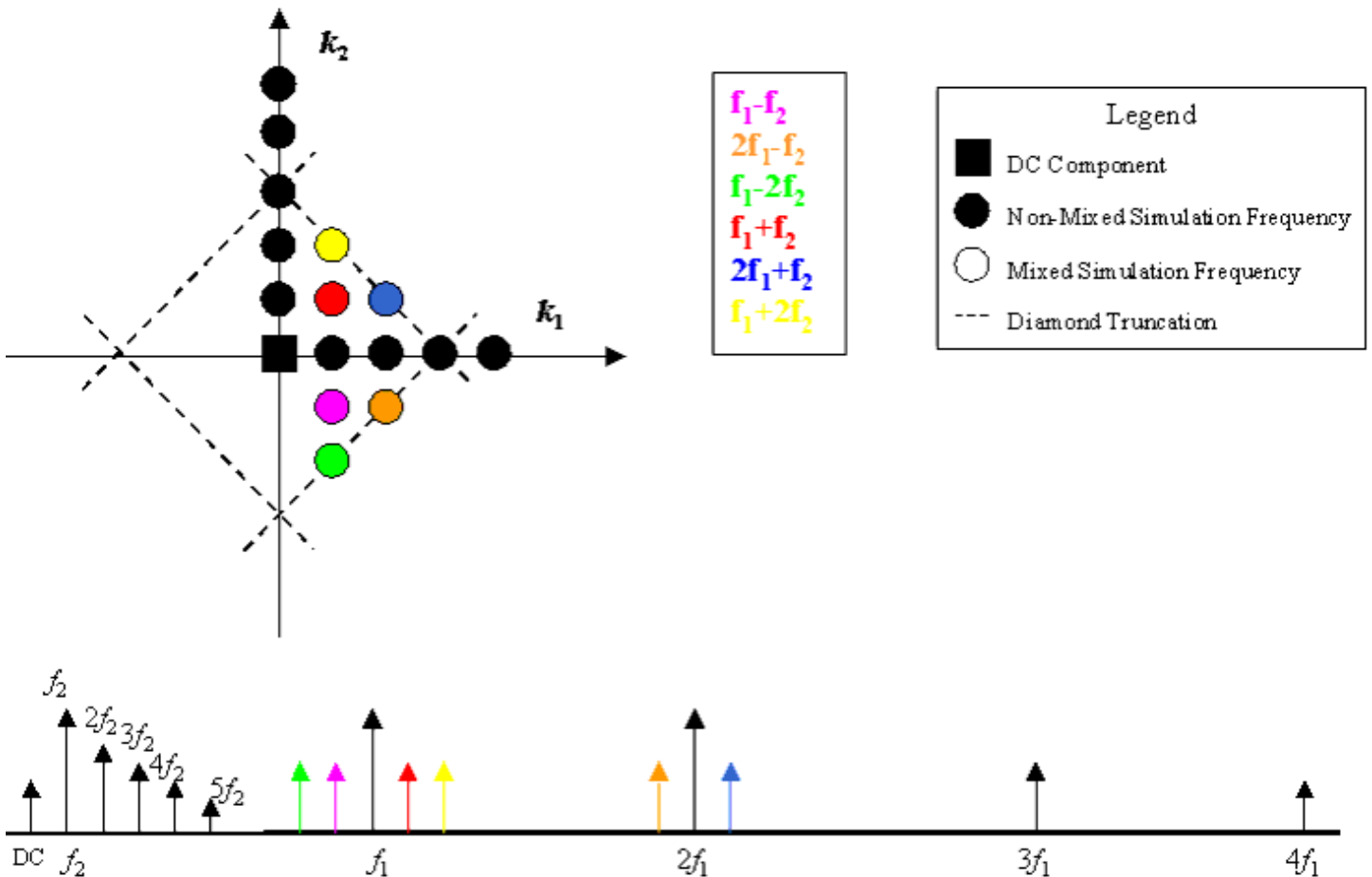
$$|k_1| + |k_2| + \dots + |k_n| \leq \text{MaxOrder}$$

where  $k_j$  is the harmonic for the  $j^{\text{th}}$  tone in the circuit. The Maximum mixing order can be set when there are two or more frequencies in the simulation. This parameter does not affect a single tone simulation, and is therefore disabled on the Freq tab. The table below gives a specific example with the first fundamental at 1.9 GHz with Order[1]= $K_1$  =4, the second fundamental at 2.1 GHz with Order[2]= $K_2$  =5, and Maximum mixing order=3. The DC term is always included as one of the simulation frequencies; however, it is not listed in the table.

Source	Frequency	Order	Non-Mixed Simulation Frequencies
Fund1 (f1)	1.9 GHz	4	f1=1.9GHz, 2f1=3.8GHz, 3f1=5.7GHz, 4f1=7.6GHz
Fund2 (f2)	2.1 GHz	5	f2=2.1GHz, 2f2=4.2GHz, 3f2=6.3GHz, 4f2=8.4GHz, 5f2=10.5GHz

Order	Mixing Term	Frequency	Maximum Mixing Order
2	f1+f2	4.0 GHz	3
2	f1-f2	0.2 GHz	3
3	2f1+f2	5.9 GHz	3
3	f1+2f2	6.1 GHz	3
3	f1-2f2	2.3 GHz	3
3	2f1-f2	1.7 GHz	3

This can also be represented in a plot of  $k_2$  vs.  $k_1$ . Consider the same two-tone case as above with  $K_1$  =4 and  $K_2$  =5, and Maximum mixing order=3. The HB simulator uses a diamond truncation method to determine which spectral components it will retain and use for simulation. This can be seen in the following figure. Note that all of the points in the plot of  $k_2$  vs.  $k_1$  will be used in the simulation for those particular values of Order and Maximum mixing order. The dashed lines are there to emphasize the diamond shape.



If Maximum mixing order is 0 or 1, no mixing products are simulated. If Maximum mixing order is not given, then it will be set to the smallest fundamental order, e.g., a diamond truncation is used to determine the mixing products. Make certain that in a multi-tone simulation, the tones are *not* defined more than once. For example, a 1 GHz tone with 3 harmonics (Order set to 3) means that 2 GHz and 3 GHz are already defined. In a multi-tone environment, such as one with a 1 GHz tone and 200 MHz tone, each with Order set to 3 and Maximum mixing order set to 5, mixing products at 1.2 GHz, 1.4 GHz, and 1.6 GHz are already defined. None of these should be redefined as fundamental frequencies in the Harmonic Balance controller. When tones are redefined, the simulator still runs and gives a warning message in the status window:

More than one mixing term has landed on frequency \*,

where \* is the value of the mixed frequency.

## Initial Guess Selection

By default, the Harmonic Balance simulator uses a DC solution as an initial guess when starting the simulation. However, a transient initial guess can provide a much better starting point for harmonic balance. This is especially true when simulating circuits that are highly nonlinear and contain sharp-edged waveforms (such as dividers). In this case, a transient simulation often provides a good initial guess for the starting point of harmonic

balance. It is recommended to use a transient initial guess when simulating frequency dividers.

## Automated TAHB

Transient assisted harmonic balance is automated and will be used if the simulator detects a divider in the circuit. By default, if the simulator does not detect a divider, then it will not use TAHB. It can also be turned on or off from the Initial Guess tab on the Harmonic Balance simulation controller. In this case, select the box labeled *On*, and the simulator will generate its own transient initial guess. The transient simulator will use intelligent defaults and determine a steady state solution as the initial guess for harmonic balance. It is not required to set any of the transient parameters on the Initial Guess tab. However, you may set the transient parameters only when TAHB is set to *On*. In that case, the settings can be activated from the Advanced Transient Settings dialog. Transient assisted harmonic balance can be turned off by selecting *Off* on the Initial Guess tab. For more details on setting the additional transient parameters, see *Transient Assisted Harmonic Balance - TAHB* (adshbapp).

# Solving Convergence Problems

This section discusses the different types of convergence problems that can occur when using the Harmonic Balance simulator. It also includes the remedies for these possible convergence problems. The parameters used for convergence are mentioned in this section, and are thoroughly described in *Additional Parameters* (adshbapp).

Prior to fixing a convergence problem, you should have some familiarity with the Convergence mode and Solver type parameters. Realize that the nonlinear outer solver, Newton solver, and Convergence mode parameter are one in the same. Similarly, the linear solver, inner solver, and solver type are one in the same. By default, these parameters are automatically adjusted by the simulator to achieve both speed and robustness. Convergence mode and Solver type are found on the Solver tab of the HB controller. Although it is not recommended to modify the Convergence mode or Solver type, you have the option to do so.

## Convergence Mode

There are three choices for the nonlinear (outer) solver that can be selected by setting the **Convergence mode** (ConvMode on schematic). The maximum number of iterations for the nonlinear solver is controlled by the parameter **Max. Iterations** (MaxIters on schematic). Max Iterations can be set to Fast, Robust, or a Custom value. The Convergence mode parameter options are described below.

- **Auto** This is the default mode setting. It is both fast and robust. 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 appears in the status window when this occurs, and it includes the tolerance level up to which convergence was achieved.
- **Robust** This option enables an advanced Newton solver. This mode is extremely robust, and ensures maximal KCL residual reduction at each iteration. The Robust convergence mode usually simulates slightly slower, yet works well for very nonlinear circuits (i.e., those with very high power levels). It is recommended that the maximum number of iterations (MaxIters) be set to Robust when this mode is selected. Another option is to select Custom and enter a value of 50 or higher.
- **Fast** This option enables the basic Newton solver. It is fast and performs well for most circuits. For highly nonlinear circuits, this mode may have difficulties converging. It is then recommended to switch to the Robust convergence mode.

## Solver Type

When using harmonic balance, you can allow the simulator to choose a solver automatically, or select one of two linear (inner) solver techniques: Direct or Krylov. The linear solver is used to solve the matrix problem generated at each iteration of the Newton (outer) solver. The matrix size will be determined by both the size of the circuit and the

total number of frequencies (fundamentals, their harmonics, and mixing products).

## Auto Select Solver

This option allows the simulator to choose which solver to use. The Auto select solver is enabled by default. 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 the Direct or Krylov solver in a manner transparent to you. The selection choice heavily depends upon the amount of available RAM. The simulator will determine roughly how much RAM is available. Alternately, you can specify the amount of RAM to allocate; however, if this is not enough for the simulator, then it will either allocate more RAM or report an error. Furthermore, if the Krylov solver is chosen by the simulator, several options for that solver also are set automatically.

It is possible to override the choice of solver given by the Auto select solver option. This can be done simply by selecting Direct or Krylov from the Solver tab on the HB controller. When simulating large circuits, that is, those with many devices and harmonics, it is recommended to use the Krylov solver.

## Direct Solver

The Direct solver uses direct matrix factoring methods (such as Gaussian elimination) to invert the Jacobian matrix. This solver is recommended for small circuits with relatively few devices, non-linear components, and number of harmonics. For large circuits, the direct solver will be slow and inefficient. This is because the computation time of the direct solver grows with the cube of the matrix size. For example, in a single-tone HB simulation, doubling the circuit size or doubling the number of harmonics (the Order) will slow the simulation run time by a factor of 8. Also, since the direct solver requires an explicit storage of the Jacobian, its memory requirements grow with the square of the matrix size. For example, the factorization of a Jacobian with a size 500 will require 2500 times as much RAM as one with a size of 10.

## Krylov Solver

An alternate approach to solving the matrix problem is to use a Krylov subspace iterative method such as GMRES. The Krylov method is intended for solving large circuits with many devices, non-linear components, and number of harmonics. (A large circuit can be roughly described as one in which a simulation using the direct solver exceeds 100 MB of memory usage or the memory capacity of the computer, whichever occurs first.) The Krylov solver does not require the explicit storage of the Jacobian matrix, but rather only the ability to carry out matrix-vector products. As a result, Krylov solver's memory requirements grow linearly with the matrix size, rather than quadratically as in the direct method. Thus, Krylov solvers offer substantial memory usage savings for large circuit

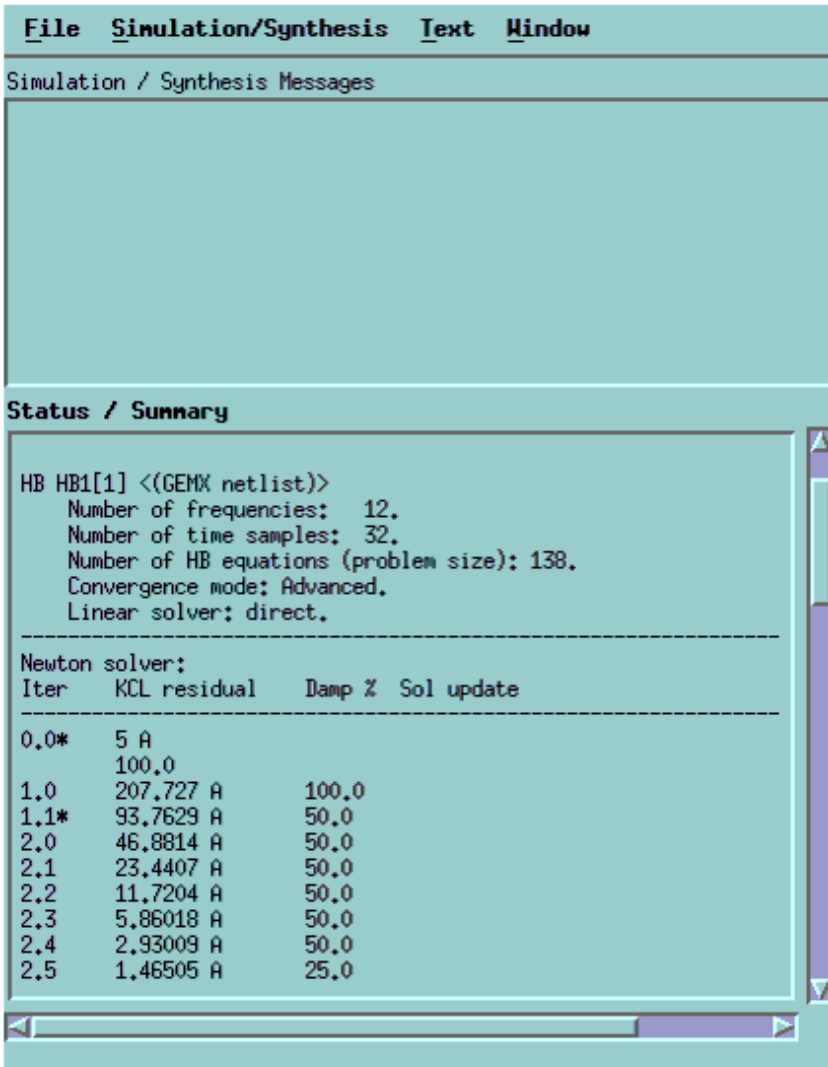
problems. Since the Krylov method solves the matrix problem to a loose tolerance, it is also much faster than the direct solver (but less robust). The computation time of the Krylov solver grows slightly faster than linear with the matrix size. For example, doubling the circuit size or doubling the number of harmonics will increase the simulation run time by slightly more than a factor of 2.

## Setting Status Level and Understanding Output in the Status Window

During an HB simulation, the simulator prints information describing the simulation progress in the status server window. The **Status level parameter** (found on the Freq tab) controls the amount of detail in this information. Reading and understanding this information is critical to solving convergence problems.

The default status level is set to 2; however, when solving a convergence problem, it is best to set the status level to 4. For each Newton iteration the L-1 norm of the KCL residuals throughout the circuit is printed.

The KCL residual indicates how well the circuit has converged up to that point. A steadily decreasing residual implies successful convergence. For example, for an HB simulation at default (strict) tolerances, this residual should reach levels of pico amps at the end. A snap shot of the ADS Status Server Window is shown in the following figure.



When using the Krylov solver, it is useful to print additional information by setting the status level to 5, as shown in the following illustration.

```

-----
Newton solver:
Iter   KCL residual   Damp % Sol update
-----
11     115.983 mA     100.0
-----
Linear solver:
Iters  Residual
-----
12     1.189e-03
-----
Krylov solver (target tol = 0.00119):
  Iter   Residual
-----
0       1.000e-00
1       3.276e-01
2       2.180e-01
3       1.208e-01
4       6.767e-02
5       3.017e-02
6       1.818e-02
7       1.220e-02
8       4.739e-03
9       3.219e-03
10      6.449e-04
-----
    
```



```

Newton solver:
Iter      KCL residual      Damp% Sol update      Linear solver:
                                         ITERS      Residual
-----
12        51.3821 mA        100.0                 10         6.449e-04
-----
Krylov solver (target tol = 0.001):
Iter      Residual
-----
0         1.000e+00
1         5.178e-01
2         3.442e-01
3         2.976e-01
4         2.138e-01
5         9.809e-02
6         7.323e-02
7         3.645e-02
8         8.977e-03
9         7.924e-03
10        1.130e-03
11        1.130e-03
11*       7.830e-04

```

- The target tol indicates the desired Krylov solver tolerance.
- The residual at each Krylov solver iteration indicates how well the Krylov solver has converged up to that point. When the Krylov solver is performing well, the residual decreases quickly, and the Krylov solver reaches the target tolerance in fewer iterations.
- The Newton solver lines include a summary of the linear solver performance: the total number of Krylov iterations and the achieved Krylov tolerance (this information is also printed for status level set to 4).
- The Sol update (solution update) is largest amount of voltage change between two successive outer solver (Newton) iterations for all solution waveforms. This will get printed toward the end of the simulation, or in the case of a swept simulation, it will get printed at the end of each sweep point.
- Because the Robust convergence mode (for the Newton solver) was used in this example, the damping percentage of the solution update is also printed.

When using the Auto solver, set the status level to 5 to see the relevant circuit statistics, memory computations, and the chosen parameter settings.

After increasing the status level, it is recommended to insert an Options controller and check the box (found on the Output tab) labeled **Issue Warnings** (GiveAllWarnings=yes on schematic). Be sure to watch for these warning messages in the status window and act upon them accordingly.

## Parameter Access

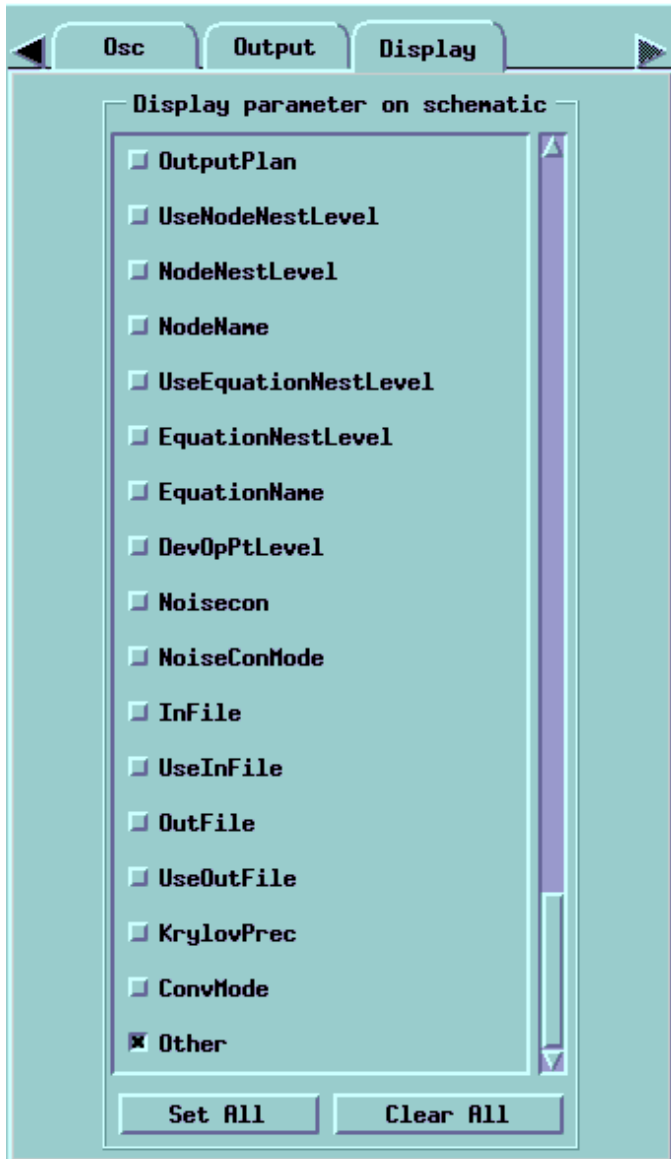
The most frequently used parameters can be accessed from the various HB controller tabs. A second group of parameters which are used less frequently can be accessed through the Harmonic Balance Display tab. Choose to display the parameter on the schematic and edit its value on the schematic. The final group of parameters are the hidden parameters. To activate these parameters, use the entry on the Display tab called

Other. The format is

```
Other=HiddenParameter1=value HiddenParameter2=value
HiddenParameter3=value....
```

The figures below show the Display tab from the HB controller and an example of how to use the Other parameter. Note that once the Other parameter has been selected to be displayed, it may be edited on the schematic. For example,

```
Other=RedRatio=0.8 NormCheck=0
```



## Circuit Operation and Verification with Transient Simulation

It is important to verify that the circuit is operating properly, as intended by the designer. Performing a transient simulation prior to a harmonic balance simulation will enable you to check for unstable circuits and circuits with multiple solutions. After running a transient

analysis, check to see if the waveforms blow up or have several spikes and sharp edges. In the case that the waveforms have these conditions, Harmonic Balance may require hundreds or even thousands of harmonics which in turn will significantly increase simulation run time and memory usage.

## Harmonic Balance Controller Setup

When a circuit does not converge, it is important to check that the controller is set up correctly and with appropriate controller parameter settings.

### Order

The Harmonic Balance solution is approximated by a truncated Fourier series. When convergence problems begin to occur, the first parameter to examine is the Order, which is the number of harmonics. The lower the Order, the greater the error due to Fourier truncation in the solution representation. The Order needs to be sufficiently large to represent nonlinear signals such as those with sharp transitions or square waves. If increasing the order causes the simulation speed to dramatically slow down or there is an excessive usage of memory, then it is best to switch from the direct solver to using the Krylov solver.

By setting the status level to 4 or more, an HB truncation error warning may be given in the status window upon a successful completion of an HB simulation. The warning contains a sorted table of the five waveforms in violation of the HB truncation error check with largest HB truncation errors. Note that the HB truncation error check is not the same as the circuit convergence check for the KCL residual; in fact, the HB truncation error warning can be generated only once the circuit has converged. The HB truncation error may not be distributed evenly across all of the computed harmonics.

If fewer than five waveforms violate the HB truncation error check, only those will be printed. If there are no violating waveforms, then the HB truncation error warning is not printed at all. Increasing the order will reduce the number of violating waveforms. An example of the warning message for HB truncation error is shown below:

Warning detected by HPEESOFSIM during HB analysis `HB1'.

An HB truncation error may be present.

- o The HB truncation error is due to using a finite order (number of harmonics) in the representation of the circuit signals.

Waveform	Trunc error		Tolerance
v2	6.576e-03	>	5.941e-06
v3	1.780e-03	>	1.043e-06

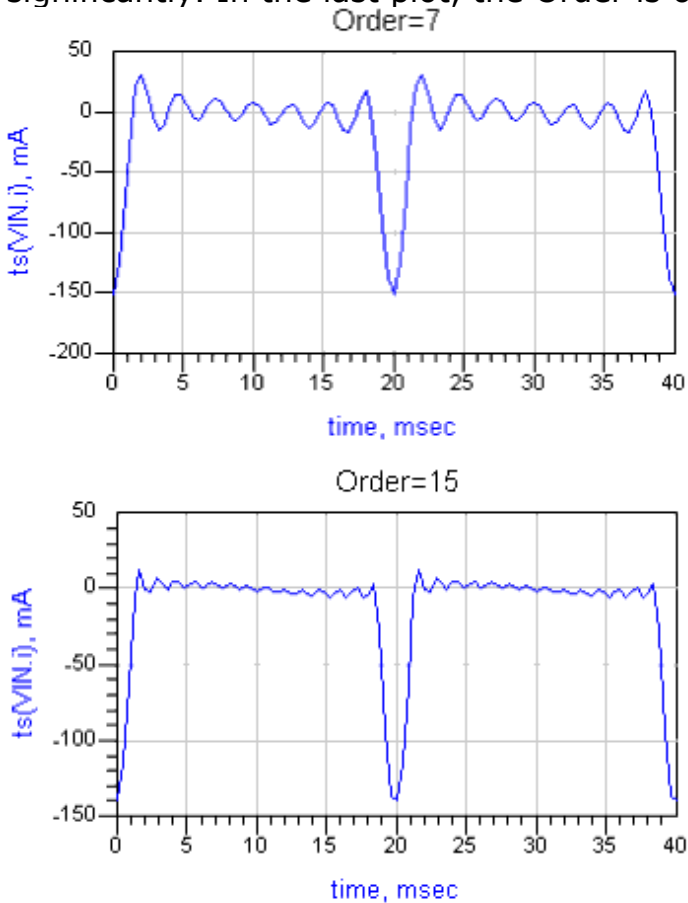
- o Number of waveforms violating the HB truncation error check:  
2 out of 2 waveforms.

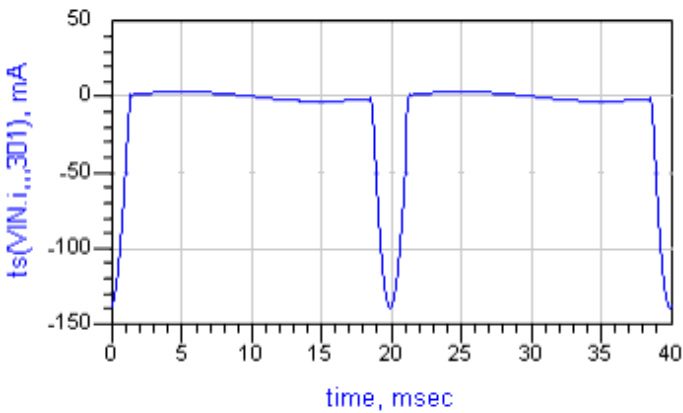
- o Estimated max HB truncation error: 6.576e-03 @ waveform v2.
- o The maximal HB truncation error estimate is greater than the

achieved tolerance of  $5.941e-06$  for this waveform.

- o A time-domain plot of the v3 waveform may show the error as Gibbs ripples.
- o To reduce the error, increase the order (number of harmonics) and re-simulate.

It is recommended to create a time domain plot of the solution waveforms with large HB truncation errors (or a plot of any other solution waveform which has sharp features, spikes, or square waves) to get an idea for how much to increase the Order to reduce the truncation error. The truncation error in the plot is seen as Gibbs ripples. An increase in the Order will reduce the truncation error, decrease these ripples, and decrease the number of waveforms violating the HB truncation error check. The plots shown in the following figure give an example of the HB truncation error and show how it is reduced when increasing the Order. When the Order=7, there are large Gibbs ripples in the output waveform. When the Order is increased to 15, the amplitude of the ripples diminishes significantly. In the last plot, the Order is 63 and the HB truncation error is negligible.





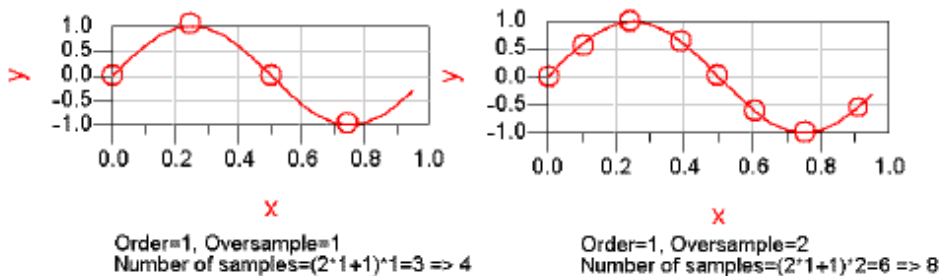
For maximum computational efficiency when simulating with the Krylov solver, set the Order=7, 15, 31, etc. This suggestion is based on the fact that the Krylov solver's computational complexity depends on the size of the FFT.

## Fundamental Oversample

In Harmonic Balance, nonlinear devices are evaluated (sampled) in the time domain, then converted to the frequency domain with the FFT (Fast Fourier Transform). When the time domain sampling rate is greater than twice the largest harmonic frequency, this is known as oversampling. See the diagram below for a waveform that is sampled with oversample set to 1 (no oversampling), and one that has oversample set to 2.

### Oversampling

Number of samples =  $(2^{\text{Order}+1}) \cdot \text{Oversample}$ , rounded to the nearest power of 2.



For each period of the input tone, the simulator will take x number of samples, determined by the values for Order and Oversample.

For a single-tone HB simulation, increasing the Fundamental Oversample parameter (found on the Params tab) can help convergence by ensuring that rapid transitions and sharp features in the device waveforms are more precisely sampled. As a rule of thumb, try Fundamental Oversample=2, 4, 8.

In a multi-tone HB simulation, it is possible to set the oversample for each tone. To do this, click *More* next to the Fundamental Oversample parameter. A new dialog box will appear enabling you to enter the Oversample values for each fundamental in the multi-tone simulation. Similarly to the single-tone case, try Oversample=2, 4, 8.

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.

## Newton Solver Issues

The default Convergence mode is the Auto mode. This mode is preferred since it is fast and robust. The Auto mode works well on a wide range of circuits, including those which are fairly linear to those which are highly nonlinear and contain sharp edge waveforms. It also works well for circuits containing a large number of transistors, and for circuits that seem to go into arc-length continuation or source stepping in only a few number of iterations.

The alternate convergence modes are Fast and Robust. The Fast mode simulates quickly and works well for most circuits. The Robust convergence mode usually simulates slightly slower, yet works well for very nonlinear circuits (i.e., those with very high power levels). The Robust mode solver should exhibit more robust convergence than the Fast mode solver. If the KCL residual in the status window output fails to continue decreasing or exhibits a bouncing pattern (alternates between decreasing and increasing), the Robust convergence mode may also help.

All three convergence modes need an initial guess. The default initial guess is based on a DC solution. Certain circuits may not converge from this starting point.

The initial guess can be changed using the hidden parameter **InitGuess**. By default, InitGuess=0 (DC initial guess). Set InitGuess=1 to use zero voltages and currents for the initial guess.

## Linear Solver Issues

If convergence issues occur while using the Direct solver, some parameters (located on the Harmonic Balance controller's Solver tab) can be modified to assist with convergence. If the Direct solver fails to converge, then set **Matrix Re-use** to Robust. This parameter controls how frequently the Jacobian is constructed and factored rather than being reused. Selecting Robust means that the Jacobian will be computed at each iteration and will not get reused for future iterations. A "\*" next to an iteration number in the status window output indicates a re-computation of the Jacobian for that iteration.

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 the parameter **Matrix Bandwidth**, and its default value is Fast. This 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, set Matrix

Bandwidth to Robust. Typical values for the Custom entry range from 10 to 0.

If convergence issues occur while using the Krylov solver, increase the status level to 5 and monitor the KCL residual and the Krylov solver residual in the status window. If the Krylov solver converges very slowly, its iterations may be terminated before the linear problem can be solved to an acceptable degree of accuracy. In such a case, the following message will appear in the status window output:

```
<name_of_Krylov_solver> terminating due to insufficient rate of convergence.
```

It is recommended to set the Krylov Restart Length (GMRES\_Restart on schematic) parameter to Custom and enter a value between 10 and 100. The default setting is Robust. However, if there is a limited amount of memory available, then select Low Memory. This parameter determines the number of iterations after which the Krylov solver is restarted. Also, to prevent the Krylov solver from stopping too soon due to "insufficient rate of convergence", increase the Krylov Convergence Ratio (KrylovConvRatio on schematic). This is the amount by which the norm of the Krylov solution must decrease from one iteration to the next. The default is 0.9 and it should not be larger than 1.0.

As a last resort, it is recommended to change the Krylov preconditioner. A **preconditioner** is used to increase the rate of convergence of the Krylov linear solver by reducing the number of iterations performed. Thus, preconditioning is essential to making the Krylov solver effective. Click *Advanced Krylov Parameters* to choose the preconditioner.

The default preconditioner is DCP. Some of the Krylov solver's convergence problems arise due to the limitations of the DCP. There can be multiple reasons for these problems, such as strong nonlinearities in the circuit generating an ill-conditioned linear problem at each Newton iteration. As a result the Newton direction becomes inaccurate so that the nonlinear solver fails to converge. When the Krylov solver has trouble converging, it is recommended to change the preconditioner to BSP or SCP. The BSP typically is more efficient for medium to large size problems, while SCP works better for very large problems. Changing the preconditioner should be done only when an error message appears in the status window giving specific instructions to change the preconditioner.

The three types of preconditioners used by the simulator are summarized below. You must select one when using the Krylov solver:

- **DC Preconditioner (DCP)** This is the default preconditioner, which is effective in most cases, but fails for some highly non-linear circuits. It uses a DC approximation on the entire circuit.
- **Block Select Preconditioner (BSP)** This 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. For the circuits that converge with DCP, the overhead introduced by the BSP preconditioner is small. For circuits that fail with the DCP, using the BSP option often achieves convergence at the cost of additional memory usage.
- **Schur-Complement Preconditioner (SCP)** This 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 known as DMRES. The complex technology of the SCP preconditioner results in a memory usage overhead. This overhead is due to construction of a knowledge base enabling the SCP to be much more efficient in the later use of the harmonic balance solution process.

## Sweeps as Convergence Tools

Parameter sweeps can be used to formulate a customized continuation method geared toward the particular circuit problem. Continuation methods provide a sequence of initial guesses that generate a sequence of solutions that approach the final desired solution.

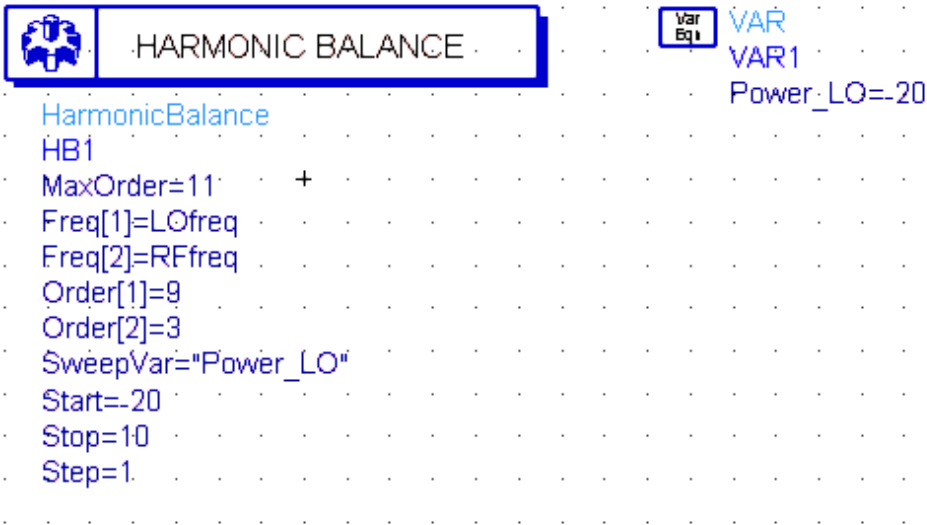
There are two main ways to perform a parameter sweep in ADS. The first way is to use the Sweep tab within the HB controller. This is the most efficient way to perform sweeps, and thus is the preferred way. The second way is to include a Parameter Sweep controller, which is a separate controller from the HB controller. For single parameter sweeps (in which the swept parameter is *not* frequency), use the Sweep tab on the HB controller. For multi-dimensional sweeps, use the Sweep tab for the inner-most sweep parameter, and use the Parameter Sweep controller(s) for the outer-most sweep parameter(s). Frequency should always be selected as an outer-most sweep parameter even for multi-tone simulations.

When a single point HB simulation does not converge, a parameter sweep can be used as a convergence tool. Performing a sweep around a single point that does not converge helps to determine if there is a range of values for which the circuit can converge. Selecting which parameter to sweep is the first step. It is best to choose a parameter that can be set to a value for which the circuit will easily converge. Some examples are the source amplitude or power, a bias voltage or current, or any component parameter that controls the amount of nonlinearity in the circuit. Find the parameter value for which the circuit converges and make this the start point of the sweep. The actual parameter value for which the circuit does not converge should be the end point of the sweep. Perform a swept simulation up to the point for which the circuit converges, and save the solution to be used as an initial guess for single point simulation that does not converge. Simulate the single point with this initial guess. This may give the Newton solver a better initial guess than the DC solution.

In most cases, a linear sweep will work best. When performing a sweep, be sure that the Regenerate Initial Guess for Param Sweep (Restart) is not checked (i.e., Restart=no). This ensures that the sweep will be used as a continuation, or in other words, the solution from the previous sweep step is used as an initial guess for the next step. Having more sweep points will give a greater chance for success, but will result in a longer computation time.

Two diagrams are shown, one for each sweep method. The following figure shows the Harmonic Balance controller sweeping the variable *Power\_LO* from -20 dbm to 10 dBm in steps of 1 dBm. A VAR equation needs to be included to initialize the parameter that is to be swept. The value of this parameter in the VAR equation can be set to an arbitrary number, since the value of the sweep start will override this value.



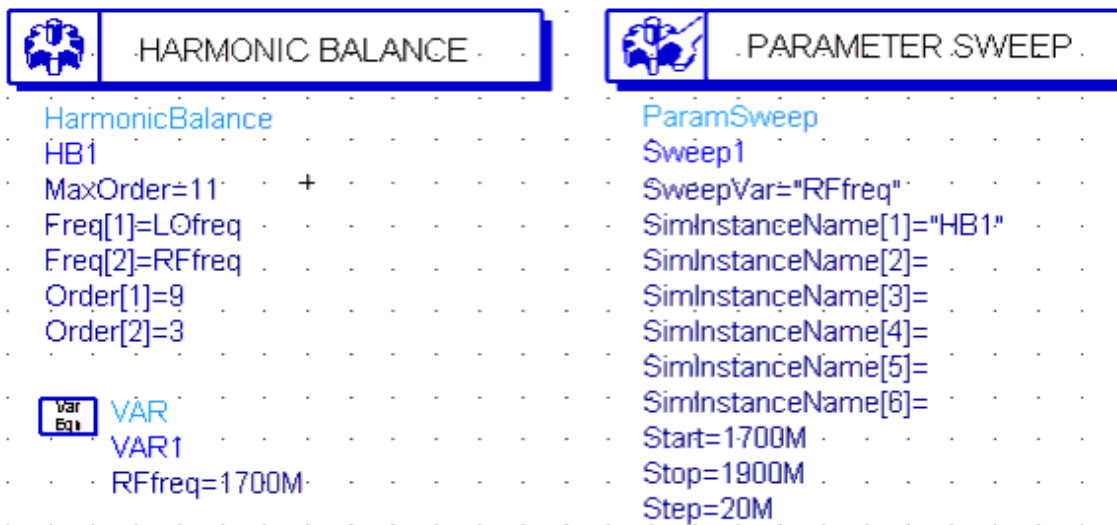


**HARMONIC BALANCE**

HarmonicBalance  
HB1  
MaxOrder=11 +  
Freq[1]=LOfreq  
Freq[2]=RFfreq  
Order[1]=9  
Order[2]=3  
SweepVar="Power\_LO"  
Start=-20  
Stop=10  
Step=1

**VAR**  
VAR1  
Power\_LO=-20

The following figure shows a parameter sweep setup using a Parameter Sweep controller. The parameter being swept is RFreq, from 1700 MHz to 1900 MHz in steps of 20 MHz. For sweeping frequency, it is recommended to use a Parameter Sweep controller, and not the Sweep tab on the HB controller. When using a Parameter Sweep controller, the SimInstanceName must be set to the instance name of the Harmonic Balance controller, as shown for the parameter `SimInstanceName[1]= "HB1"`.



**HARMONIC BALANCE**

HarmonicBalance  
HB1  
MaxOrder=11 +  
Freq[1]=LOfreq  
Freq[2]=RFfreq  
Order[1]=9  
Order[2]=3

**VAR**  
VAR1  
RFreq=1700M

**PARAMETER SWEEP**

ParamSweep  
Sweep1  
SweepVar="RFreq"  
SimInstanceName[1]="HB1"  
SimInstanceName[2]=  
SimInstanceName[3]=  
SimInstanceName[4]=  
SimInstanceName[5]=  
SimInstanceName[6]=  
Start=1700M  
Stop=1900M  
Step=20M

When a swept Harmonic Balance simulation does not converge, you can

- adjust convergence parameters and keep restarting the swept simulation from the very beginning, or
- split the sweep into two parts, or
- perform a single point simulation at the value for which the swept parameter does not converge to determine if the simulation will converge for just that one particular point in the sweep.

The first option is feasible for small circuits that simulate quickly. The second option is preferred for larger circuits with longer simulation run times.

For example, consider sweeping the input power from -20 dBm to 10 dBm. If the circuit

does not converge, reduce the range of the sweep so that the last point is the one for which the circuit will still converge (this is the first sweep). Suppose the circuit converged only up to 5 dBm. The 5 dBm solution can be saved in an output file: select the parameters Write Final Solution and enter the name of the file for the output to be saved. Adjust parameters such as Order, Oversample, and number of iterations; then try a second sweep from 5 to 10 dBm and see if the circuit will go beyond 5 dBm. The 5 dBm output file should be used as an initial guess for this second sweep: select Use Initial Guess and enter the name of the file.

As a more detailed example, consider sweeping the RF frequency in a mixer circuit (with the Fast convergence mode and Krylov solver) from 0.5 GHz to 1.5 GHz, using 11 sweep points (0.1 GHz step size). Suppose this circuit can only converge up to the RF frequency point of 1.0 GHz and fails at 1.1 GHz. At this point, it is recommended to follow these steps:

1. Break the sweep into two parts (the first part will be a sweep over the range of frequencies for which the circuit converges, and the second part will be the remaining sweep points).
2. Simulate the first part to generate an initial guess which can be used for the second sweep.
3. Adjust parameters to achieve convergence for the second part of the sweep.

For this mixer example, it is desired to have 11 sweep points between 0.5 GHz and 1.5 GHz. This means that spacing between sweep points is  $(1.0 \text{ GHz})/10 = 0.1 \text{ GHz}$ . The frequency sweep points are then placed at: 500 MHz, 600 MHz, 700 MHz, 800 MHz, 900 MHz, 1000 MHz, 1100 MHz, 1200 MHz, 1300 MHz, 1400 MHz, and 1500 MHz. Setup the simulation for the first sweep with a VAR block to define  $f_{start1}=0.5G$ ,  $f_{step1}=1.0G/10$ ,  $f_{stop1}=f_{start1}+5*f_{step1}$ , and  $np1=6$ . Since the simulation does not converge beyond 1 GHz, the first sweep is done up to that point, which is 5 frequency points after the start i.e.,  $f_{stop1}=f_{start1}+5*f_{step1}$ . The total number of points for the first sweep is 6 ( $np1=6$ ). The remaining 5 points will be used for the second half of the sweep.

Instantiate a Parameter Sweep controller, and set  $Start=f_{start1}$ ,  $Stop=f_{stop1}$ , and  $Num. \text{ of Points}=np1$ . The sweep step size will be determined by the  $Num. \text{ of Points}$ , and will be equivalent to the value for  $f_{step1}$ . It is not necessary to specify the step size parameter when specifying the  $Num. \text{ of Points}$  parameter. Next, on the Harmonic Balance controller's Params tab, select Write Final Solution and enter the name of the file that will be used as an initial guess for the second sweep. Run the HB simulation. After it completes, add the following equations to the VAR block -  $f_{start2}=f_{stop1}+f_{step1}$ ,  $f_{stop2}=1.5G$ ,  $np2=5$ . We want the second sweep to start from the point at which the original sweep failed, thus,  $f_{start2}=f_{stop1}+f_{step1}$ . There are five remaining points, so  $np2=5$ . Go back to the HB controller, and on the Initial Guess tab, unselect Write Final Solution. Now select Use Initial Guess and enter the name of the file that was written during the first sweep. Next, return to the Parameter Sweep controller, set  $Start=f_{start2}$ ,  $Stop=f_{stop2}$ , and  $Num. \text{ Of Points}=np2$ . (Or deactivate the Parameter Sweep controller and instantiate a new one with the sweep var as RFFreq but with the start and stop with the values for the second sweep). It is not necessary to enter the step size since that is determined by using the  $Num. \text{ Of Points}$ , and the Start and Stop. The next step is to adjust certain parameters to achieve convergence. Recall that the nonconverging simulation was using the Fast convergence mode. For the second half of the sweep, it is then recommended to use the

Robust convergence mode (found on the Solver tab) with the Krylov solver. For this mixer circuit, convergence was achieved using the Robust mode. Alternatively, the entire sweep could have been performed using the Robust convergence mode from the beginning rather than performing two sweeps. However, this approach is less efficient than the two part sweep due to the overhead computation required by the Robust mode in the first part of the sweep.

### **Note**

After doing a HB analysis, you may want to do an HB noise analysis. A saved final solution may be used as the initial guess for other simulations such as noise analysis (of the same circuit) so that the node voltages and branch currents do not have to be recalculated.

## Transient Assisted Harmonic Balance - TAHB

The DC solution is the default initial guess; however, a transient solution can be a better initial guess for the Newton solver. The size of the initial KCL residual (seen from the status window output) is a measure of the quality of the initial guess (the smaller the KCL residual, the better the initial guess). A better initial guess such as TAHB can yield several orders of magnitude improvement in the initial KCL residual. 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.

### Automated TAHB

By default, the HB simulator will determine if there is a divider in the circuit. If so, a transient simulation will be performed to create an initial guess for the HB simulation. TAHB can also be turned on or off from the Initial Guess tab by selecting *On* or *Off*, respectively. When a divider is detected or if TAHB is set to *On*, the transient simulator will use intelligent defaults and determine a steady state solution as the initial guess for harmonic balance. It is not required to set any of the other parameters on the Initial Guess tab. The transient parameters can be set only when TAHB is set to *On*.

### Setting Advanced Transient Parameters

Advanced transient parameters can be set when TAHB is set to *On*. Some are included on the Initial Guess tab in the Transient Assisted Harmonic Balance section under Advanced Transient Parameters. The first parameter is the StopTime, e.g., the ending time for the transient simulation. The default is 100 cycles of the commensurate frequency. (For a one tone analysis, this would be Freq[1]). If the transient simulator detects steady state, then the simulation will end one period after that time point, and thus earlier than the StopTime. The one period after the time point gets transformed to the frequency domain for harmonic balance. If transient does not reach steady state, then the last period before the StopTime will get transformed to the frequency domain, and harmonic balance will use that for the initial guess. If convergence is not achieved in this

case, then it is recommended to increase the StopTime so that transient runs longer than 100 cycles.

The second parameter is MaxTimeStep. The default is  $1/(2*4*\text{Maximum frequency})$ . In a single tone analysis, the maximum frequency would be  $\text{Freq}[1]*\text{Order}[1]$ . Be sure to set MaxTimeStep small enough to accommodate the largest frequency. The simulator will display the values that it determined for StopTime and MaxTimeStep in the status window.

The third parameter is Min Time for detecting steady state (SteadyStateMinTime on schematic). This is 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 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 fourth parameter is IV\_RelTol. This is the transient current and voltage relative tolerance, and the default is  $1e-3$ . It is specific to the transient analysis and will override the relative tolerances on the Options controller if one is found in the schematic. If an Options controller is included in the schematic, be sure to set the IV\_RelTol to a reasonable value, such as  $1e-3$ . Otherwise the transient simulation would run for a very long time since it would use the relative tolerances on the Options controller which are set for harmonic balance.

Any transient parameter that is not found on the tab can be set using the Transient Other parameter. For example, one could set the following transient convolution parameter  $\text{ImpMaxFreq}=10 \text{ GHz}$ .

When TAHB is set to *On*, a user-supplied initial guess is not able to be specified, and thus it is not interpreted at all. If one is given and TAHB is set to *Auto* (the default), the user-supplied initial guess takes the highest precedence.

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

An initial guess generated by a single tone transient simulation may be used for a multi-tone harmonic balance simulation. This approach is strongly recommended, and will often result in much better convergence than with a multi-tone transient simulation. The parameter *Use only Freq[1] for transient* instructs transient simulator to perform a one tone simulation, and to only use the value of  $\text{Freq}[1]$  on the harmonic balance controller for determining the StopTime and MaxTimeStep. The default is yes. If there are multiple frequencies on the HB controller and this parameter is yes, then sources in the circuit at the other frequencies will be turned off for the transient portion only. Also,  $\text{Freq}[1]$  should be set to the frequency of the most nonlinear tone.

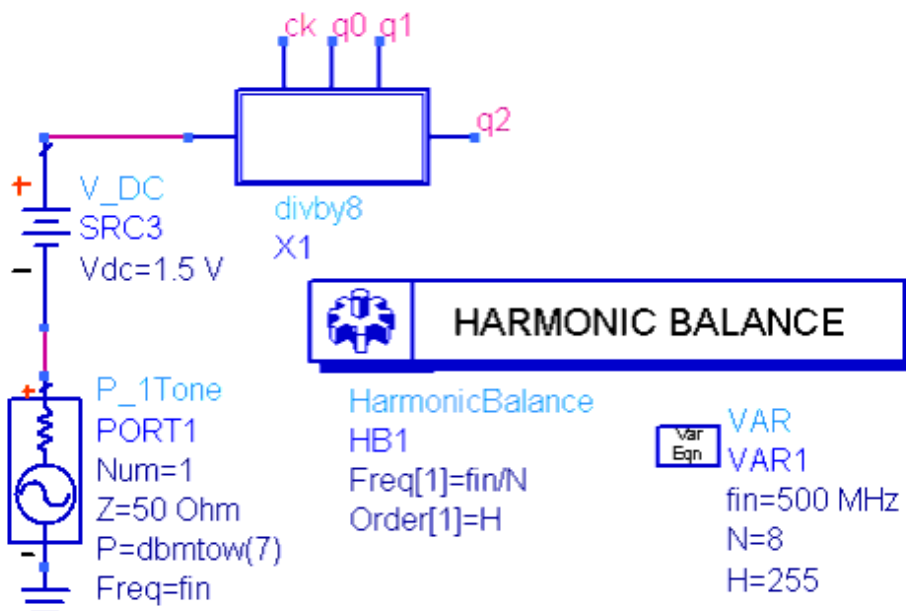
## Swept and Optimizations Simulations with TAHB

Any optimization or statistical analyses such as optimization, yield, Monte Carlo, DOE, and yield-optimization are not supported with automated TAHB. The simulation will use the standard harmonic balance in those cases.

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 ParamSweep controller, a transient simulation will be done for the first sweep point only, unless the Regenerate Initial Guess for ParamSweep (Restart) parameter is set to yes, e.g., the check box is selected. In the case that the Restart is set to yes, a transient simulation will be done at each sweep point and thus generate a new transient initial guess for each sweep point of the harmonic balance simulation.

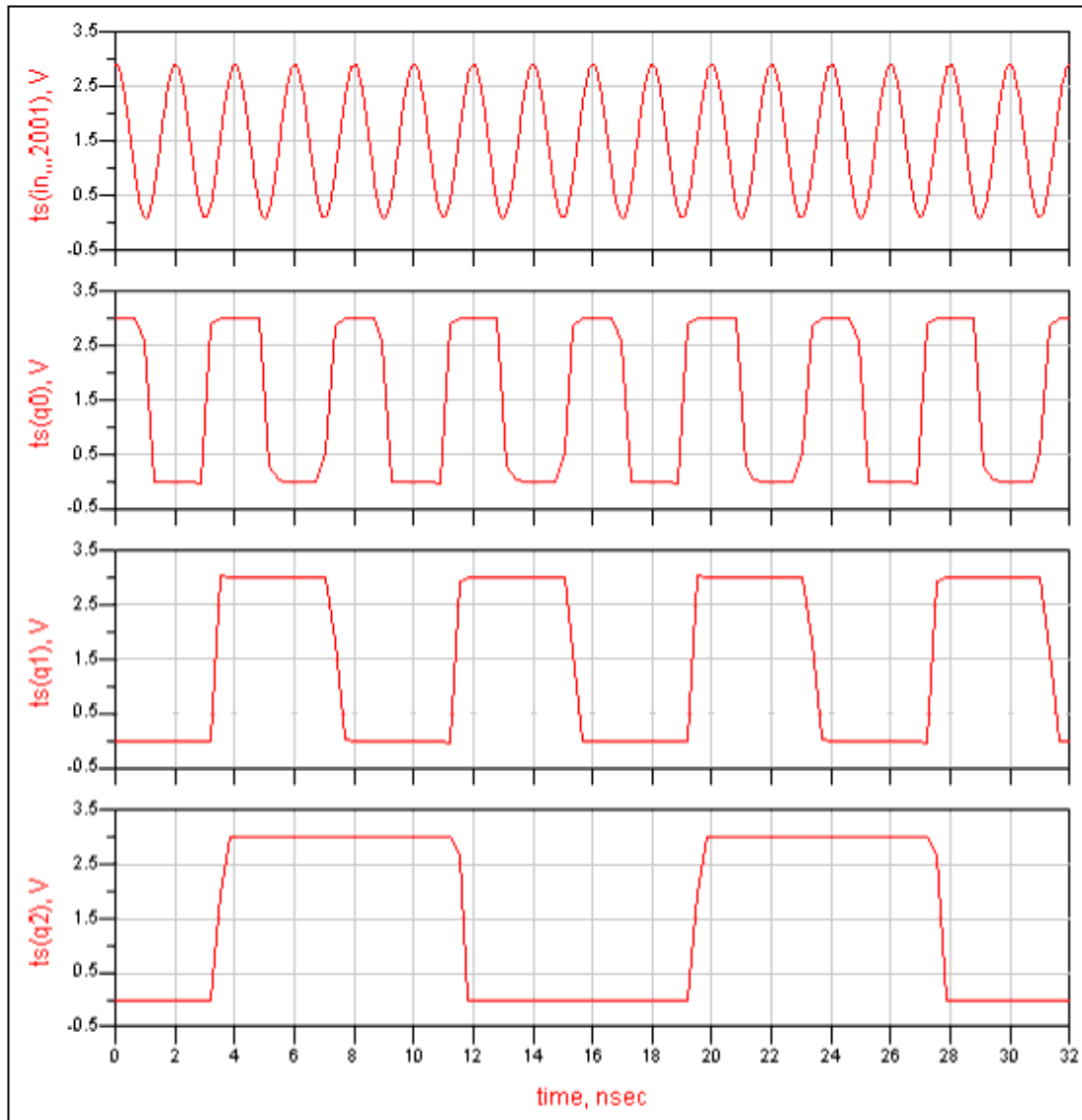
## Divide-by-8 Example

Consider a divider with a divide ratio of 8, that may eventually follow an oscillator. It is recommended to use TAHB to simulate this circuit. One reason for using TAHB is because a divider, mathematically, has more than one solution, however, only one is physical. TAHB will provide the real solution. Another reason for using TAHB is that a phase noise analysis on the future oscillator can be determined. The simulator will determine that this is a divider, and generate a transient initial guess for harmonic balance. Set the Freq[1] to final divided frequency, and set the order large enough to get a good representation of the square-like waveforms. For divider type circuits, this can be very large - Order[1]=255 in this example. See the circuit diagram with the harmonic balance simulation setup in the following figure.



The results are shown in the next figure. The first waveform is the source input at 500 MHz. The waveforms which follow are the output after division by 2 (250 MHz), division by 4 (125 MHz), and ultimately division by 8 (62.5 MHz). The cross() function in data display was used to verify the correct frequencies of the output waveforms. The cross function computes the zero crossing going in positive slope direction, as indicated by the value of the second argument. The 1.5 V DC offset is also accounted for. To ensure accuracy of the computed mean value frequency, the first few periods are not included, as seen with the

vector range notation [::] applied to the cross function.



FreqIn	FreqDiv2	FreqDiv4	FreqDiv8
500.1 M	250.0 M	125.0 M	62.50 M

**Eqn**  $\text{FreqIn} = 1 / \text{mean}(\text{cross}(\text{ts}(\text{in}) - 1.5, 1)[3::\text{sweep\_size}(\text{cross}(\text{ts}(\text{in}) - 1.5, 1)) - 1])$

**Eqn**  $\text{FreqDiv2} = 1 / \text{mean}(\text{cross}(\text{ts}(\text{q0}) - 1.5, 1)[2::\text{sweep\_size}(\text{cross}(\text{ts}(\text{q0}) - 1.5, 1)) - 1])$

**Eqn**  $\text{FreqDiv4} = 1 / \text{mean}(\text{cross}(\text{ts}(\text{q1}) - 1.5, 1)[2::\text{sweep\_size}(\text{cross}(\text{ts}(\text{q1}) - 1.5, 1)) - 1])$

**Eqn**  $\text{FreqDiv8} = 1 / \text{mean}(\text{cross}(\text{ts}(\text{q2}) - 1.5, 1)[1::\text{sweep\_size}(\text{cross}(\text{ts}(\text{q2}) - 1.5, 1)) - 1])$

## Manual TAHB

If it is desired to perform a manual TAHB, there are two ways of doing so. The first way is

to use the steady state detector in transient and allow the simulator to automatically capture the steady state portion for the initial guess. The second way is to manually adjust the transient StartTime and the StopTime to capture the steady state portion of the waveform.

## Running Transient and Generating the Initial Guess

For TAHB, a transient simulation is done first. Lets take the first method and use the steady state detector. On the Freq tab of the transient controller, select Detect Steady State. It is required to give at least one frequency and order (Freq[1] and Order[1]) parameter, and set the Frequency on the Freq tab to the same Freq that will be used in the Harmonic balance simulation. The Order on the Freq tab of the Transient simulation controller can be set to be the same or larger than the Order used in the HB controller. In the box labeled Compute HB Solution, it is optional to check the box Apply Window. In the same box, check off Write initial guess for HB, and specify the name of the file. The transient simulator will report whether or not steady state was reached, and if so, the time at which it was reached and frequency of oscillation (when simulating an oscillator). The simulator will stop once steady state has been reached and transform just the last period of the solution. Thus, the transient simulation can end earlier than the StopTime, if steady state has been reached.

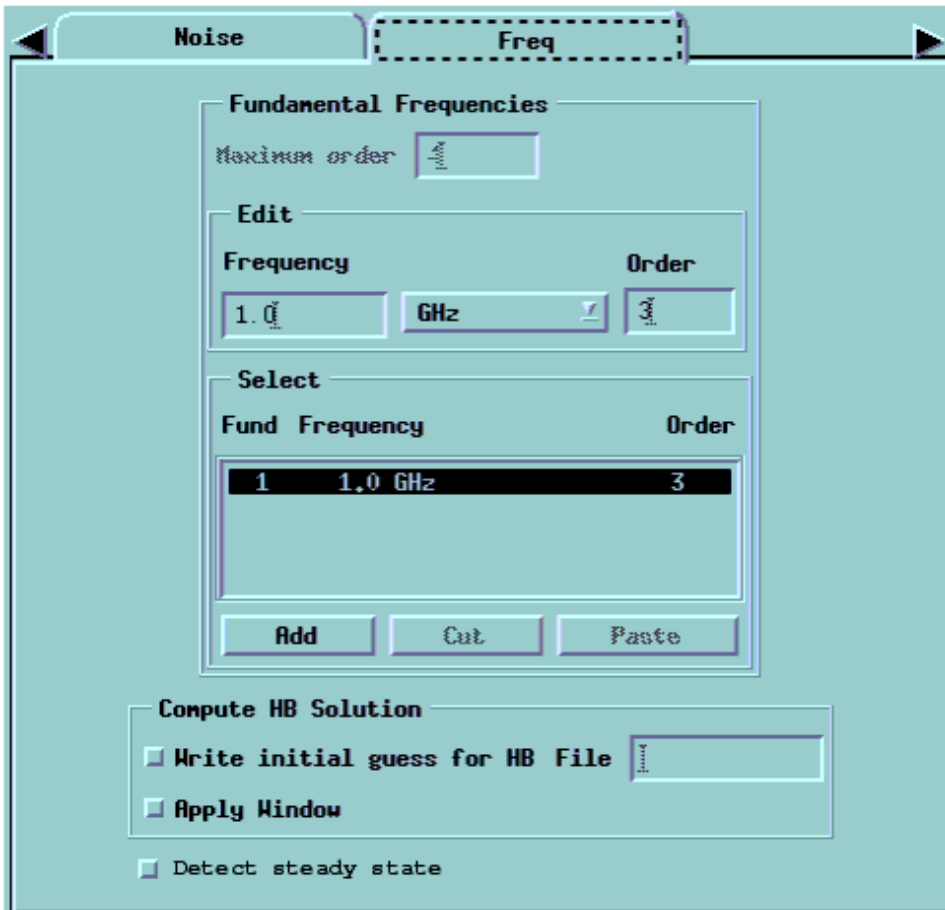
Next, lets consider the second approach of adjusting the transient start and stop time to capture the steady state portion of the waveform. The transient analysis needs to get as near steady state as possible. The conversion of transient (time domain) initial guess to the frequency domain is done from the start time to the stop time. Observe the transient waveform output in the data display. Set the start time appropriately so that the non-steady state portion is not transformed. For example, if the circuit is very near steady state after 50 nsec, then re-run the transient simulation with start time 50 nsec and end it with enough cycles (suppose 70 nsec is the end point). This way, the transient initial guess will only correspond to the part that is very near steady state (the waveform from 50-70 nsec) and not the part which is far from steady state (0-49 nsec). The quantity (stop time - start time) should be an integral number of commensurate periods. If the circuit topology is changed, then another transient simulation needs to be performed to generate a new initial guess file.

Set Max Time Step small enough to accommodate the largest signal frequency. For example, in a mixer circuit, the largest frequency is the LO+RF, and for a power amplifier it would be the third order frequency ( $2f_1 + f_2$ ). A general rule of thumb is to take 16 time points per signal period, so this means (for the mixer example) Max Time Step =  $1/[16*(RF+LO)]$ . In the case that the circuit has square wave like waveforms or rapid transitions, more points should be taken.

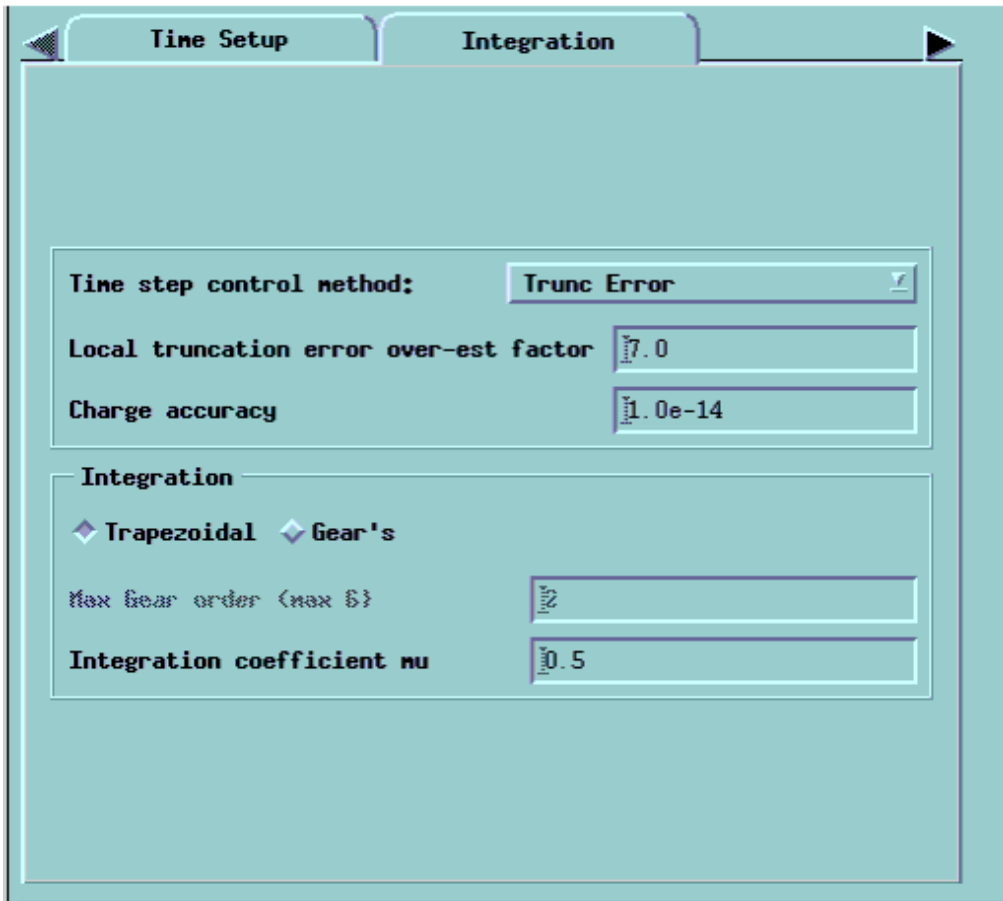
The initial TAHB guess does not need to contain all the HB frequencies, i.e., a multi-tone HB simulation can use a single tone TAHB initial guess. This is often a much more efficient approach because the transient simulation will have a faster run time. For example, you could do a one-tone transient simulation with just the very nonlinear LO, save that solution and then use it as the initial guess in the two-tone HB simulation. This approach works well in the above transceiver example. The exact frequencies do not have to match

between the present analysis and the initial guess solution. (A single tone HB solution done with a 1 GHz fundamental can be used as an initial guess for a single tone HB solution at 1.1 GHz fundamental.) When using an initial guess file, the simulator reads the index information and not the absolute frequency. A single-tone HB simulation done at 2 GHz with an initial guess from a 1 GHz simulation, will use the 1 GHz fundamental value as the initial guess for the 2 GHz fundamental value, and not the 2 GHz second harmonic value.

The following figures illustrate the Freq and Integration tabs on the Transient simulation controller.







## Reading the Initial Guess into HB

After running the transient simulation, you now have the initial guess for the HB simulation. To use the guess, select Use Initial Guess (on the Initial Guess tab) and enter the name of the file from the transient simulation. Now, run the HB simulation.

Note that if the circuit topology is changed, then another transient simulation should be run to generate a new initial guess. Be sure that the transient initial guess is a good one and that it is very near the steady state before doing the HB simulation; otherwise HB will still have trouble converging. Verify the transient initial guess by plotting the results in the data display. TAHB works well for highly nonlinear circuits and mixed signal circuits such as those with dividers, as long as there is a good initial guess.

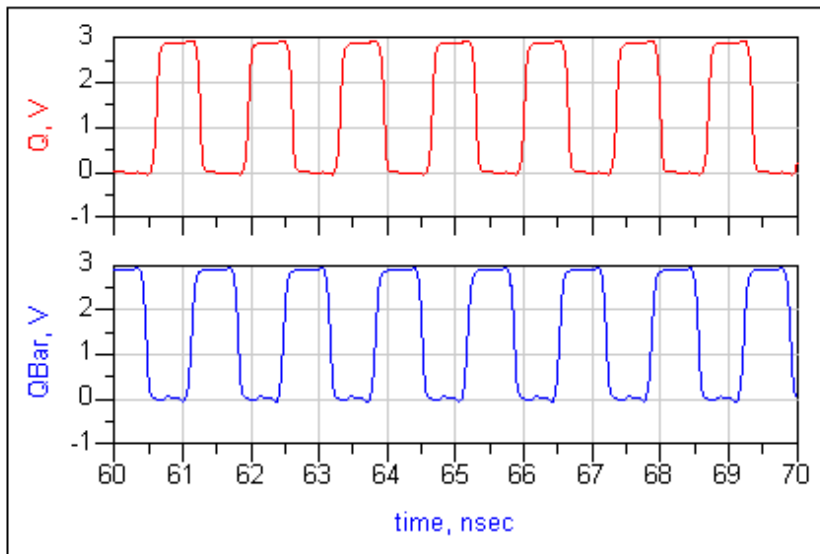
## TAHB for 1-Tone HB Simulation of an Oscillator and Divider Circuit

Consider a 1-Tone HB Simulation of an oscillator and divider circuit that does not converge. This type of circuit will have square-like waveforms with sharp edges or spikes and will require a large number of harmonics to represent the waveforms. Having a good initial guess from a transient simulation will help this type of circuit converge. It is important that the transient initial guess contains the waveforms when they are very near

steady state, and not during circuit startup. Adjust the start and stop times to capture the steady state behavior of the waveform. Run the transient long enough to determine when it approaches steady state.

In this example, the frequency of the oscillator is 738 MHz and the divide ratio on the divider is 2. The transient simulation was run for 90 nsec to determine when the circuit was near steady state. Then it was re-simulated from 60 to 70 nsec since the circuit was very near steady state in that time range. In some cases it may take the circuit longer to reach steady state. It is strongly recommended to plot and verify the transient results before starting the HB simulation. See the waveform plots of the divider in the following diagram.

Transient Initial Guess for Oscillator and Divider Circuit



After generating the initial guess from transient, the single-tone HB simulation was performed. The frequency and order were the same as specified in the transient setup, namely  $\text{Freq}[1]=738$  MHz and  $\text{Order}=31$ . Since the circuit had square wave forms, the transient solution was a very good initial guess for harmonic balance.

## TAHB for 2-Tone HB Simulation of a Large Transceiver Circuit

Consider a 2-Tone HB Simulation of a large transceiver circuit that fails to converge due to a very large initial residual. The reason for this is that the DC initial guess is too far away from the actual solution. The circuit uses two tones:  $\text{LO}=2140$  MHz and  $\text{IF}=260$  MHz. The commensurate frequency (greatest common divisor) is 20 MHz. That is a 50 nsec period (13 cycles of the LO and 107 cycles of the IF). The number of periods required for the transient simulation will depend on how quickly the circuit approaches steady state. The best way to determine how fast the circuit approaches steady state is to plot the transient simulation waveforms. Since the transceiver in this example approaches steady state relatively quickly, 5 periods of the commensurate period is sufficient: thus the transient Stop time should be set to  $5 \times 50$  nsec = 250 nsec. If the commensurate frequency is small and that circuit does not approach steady state quickly, the transient simulation to compute the TAHB initial guess would take a long time. In those cases, it is recommended

to do a one-tone transient simulation using the more nonlinear tone, which is typically the LO.

The number of periods required to reach near steady state depends on the type of circuit. Divider circuits and DC coupled mixers will only need about 2 periods, while circuits with large time constants, and AC coupled mixers may require 20 periods or more.

For the transceiver example, both the IF and LO frequencies should be included on the Freq tab of the transient controller. Since the largest significant tone in this example is  $LO+RF = 2400$  MHz (208 ps period), set the Max Timestep to  $(1/16)^{th}$  of this period, i.e. to 13-15 ps.

When this two-tone TAHB guess is used in the HB simulation, the initial KCL residual shows 12 orders of magnitude of improvement and the circuit converges.

#### Note

In order to save an HB solution to an output file, select **Write Final Solution**. If a file name is not supplied, it is internally generated using the design name, followed by an *.hbs* suffix, and is saved in the Data folder of the workspace. If a file name is supplied, the suffix is neither appended nor required. If this box is checked, then the last HB solution is written out to the specified file. If this is the same file as that was used for the initial guess, then this file is updated with the latest solution. An HB solution can be saved and reused as the initial guess for a noise analysis for the same circuit that generated the output file. This way, the simulator will not have to recalculate the solution.

## Changing the DC Convergence Algorithm

It is important to have a good initial guess for the Newton solver. When doing a Harmonic Balance simulation, the simulator will first do a DC simulation to generate an initial guess. In the case that the DC simulation does not converge, the simulator will halt and send an error message to the status window. If this occurs, deactivate the HB controller, and perform a DC analysis using an alternate convergence mode. The convergence modes for the DC simulation are located on the DC controller's Parameters tab; click *Advanced*. Some typical DC convergence problems and their remedies are described in the *DC Simulation* documentation. Please refer to them for further instruction. Repeat the DC simulation using one of the remedies mentioned in *DC Simulation* (cktsimdc). Once convergence has been achieved, deactivate the DC controller and activate the HB controller. Insert an Options controller with the DC Convergence Mode parameters settings that produced DC convergence, and re-run the Harmonic Balance simulation.

It is possible for some circuits to have multiple solutions. Depending on the DC convergence mode, the simulator may find a solution but that solution may be non-physical. For example, it may determine node voltages which are greater than the supply voltage. In cases like these, follow the above instructions to select a different convergence mode in order to obtain the desired physical DC solution.

In the case that the DC simulation is slow, save that solution and use it as an initial guess for HB so that it does not have to be computed when doing multiple HB simulations for that circuit. If the circuit topology has changed, then DC solution will need to be recalculated by performing a DC simulation.

## Device Models

Some device models may include equations, first and second derivatives with discontinuities. Model problems can cause the KCL residual to hit a threshold and remain stagnant or to exhibit random jumps (sudden increase in value). It is not recommended to use very old models, such as the Berkeley MOSFET Level 1, 2, 3. Also, it is best to use the latest version of the model, especially true for the BSIM3 model.

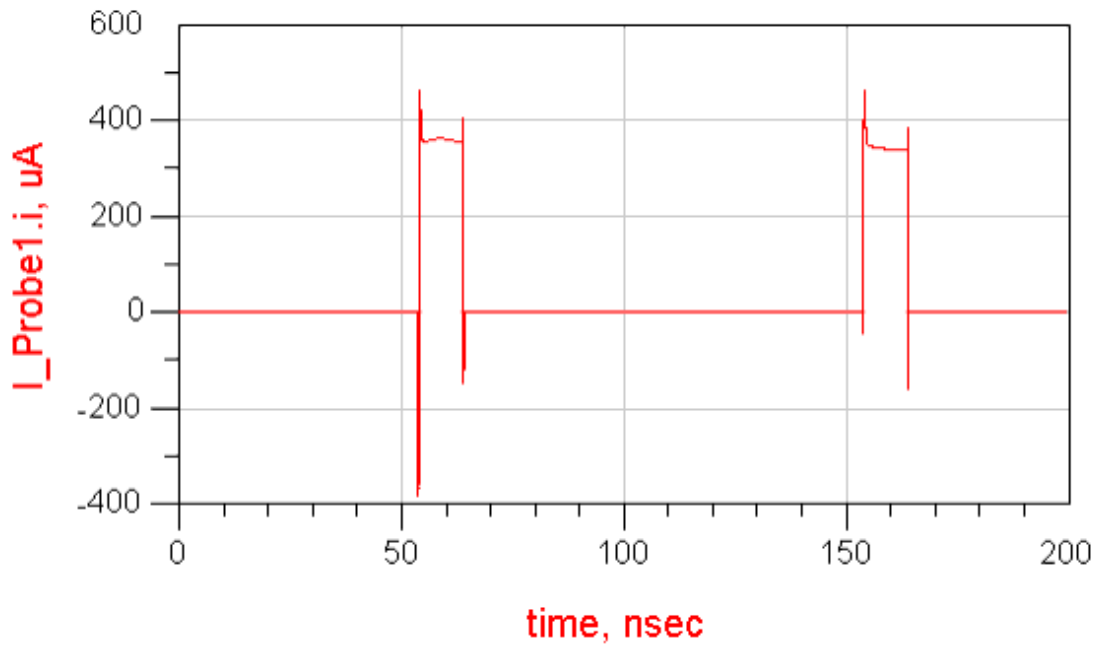
If the convergence problem is suspected to be due to a model, try replacing the model and resimulating. Another thing to try is to disable the devices that use the suspected model and resimulating. Yet another thing to try is to create a small circuit using the model, simulating and ensuring that the model is working properly. When using a particular device model, be sure that they give the expected DC I-V curves - ADS contains schematic design templates for this purpose. In the actual circuit, make sure that the transistors are biased properly, and that the model parameters are set to reasonable values.

SDD based device models need to be checked for equation discontinuities between regions, as well as for using unprotected functions that can blow up (such as exp, sqrt, log). It is not enough to insure continuity and limit the functions only in the operating range of the devices. This is because the Newton solver often takes a path that goes through points which are well outside the device operating range.

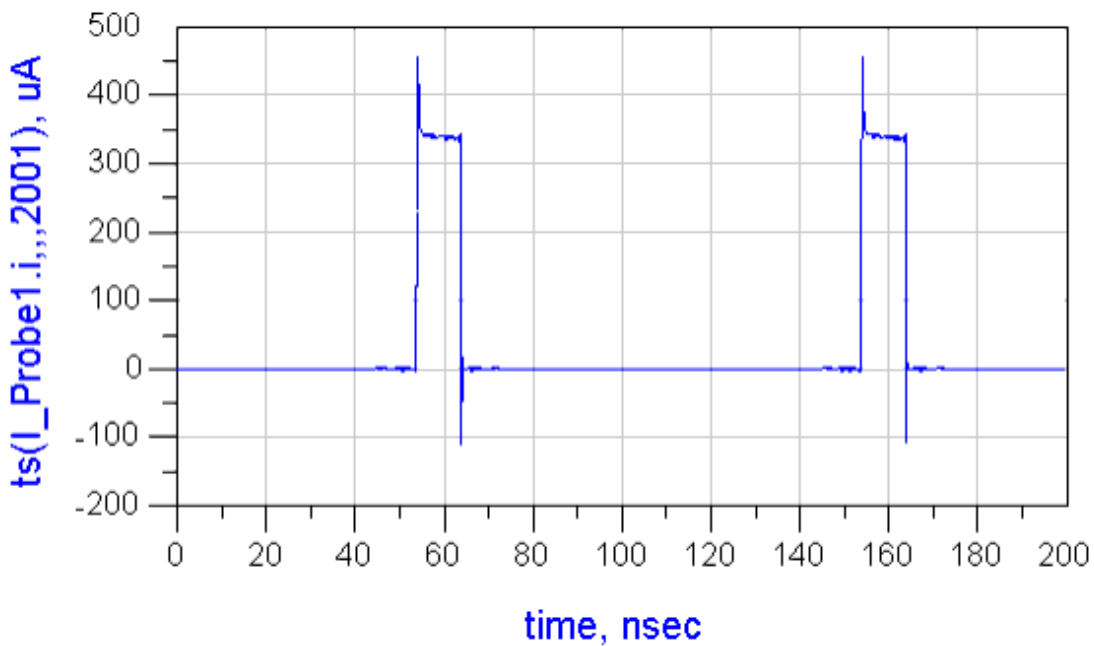
As an example, consider an HB simulation with input power sweep for a circuit which goes into arc-length continuation and fails at an input power of -4 dBm. The circuit contains TOM GaAs models. The reason that this circuit fails is that the TOM GaAs model parameters Gscap and Gdcap are set to 5 which corresponds to a non charge conserving model (physically inaccurate). The convergence remedy is to set Gscap and Gdcap to 6 which selects a charge conserving model. For efficiency, select the Robust convergence mode instead of Fast. The circuit converges up to a 50 dBm input.

## Fourier Truncation Error

There are some circuits with square or pulse type transient waveforms. To represent these waveforms with Fourier series, many harmonics are needed. This is controlled and limited in the HB simulator by the parameter Order as discussed earlier. Circuits with square waveforms can have a difficult time converging in Harmonic Balance unless the Order is sufficiently high. In some cases Harmonic Balance may not converge, and in other cases the Harmonic Balance solution may converge, yet the solution waveforms contain Gibbs ripples. The plots below show the output waveform (charge pump current) of a phase frequency detector and charge pump simulated in transient and Harmonic Balance. To get a meaningful time domain plot, click *Advanced* in the Data Display Plot dialog box, and enter the expression:  $ts(v, , , NumOfPts)$ , where  $v$  is the waveform name and  $NumOfPts$  is the size of the frequency-to-time conversion. When high orders are used, be sure to set  $NumOfPts$  to a sufficiently large number (a few thousand).



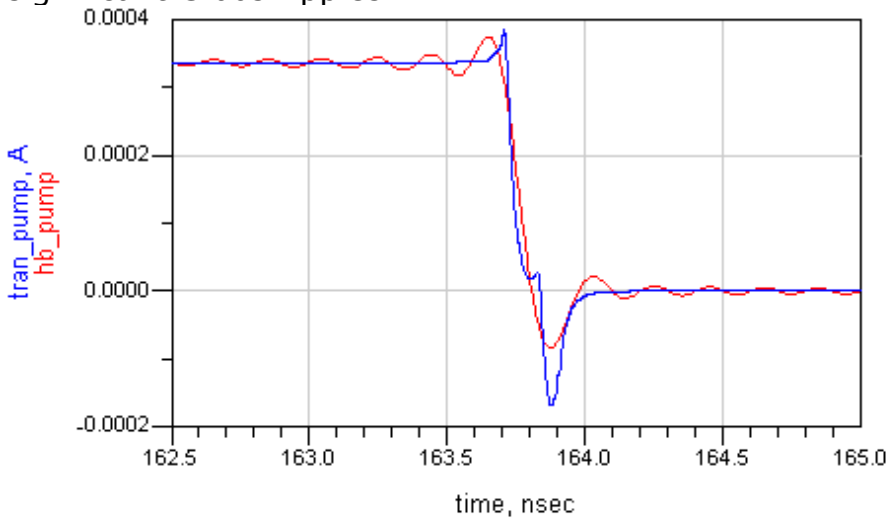
**Transient Simulation**



**Harmonic Balance Simulation, Order=127**

Circuits with such waveforms push the HB algorithm to its limits. In order to achieve convergence for this circuit in HB, the Oversample was set to 2, and the tolerances were very loose ( $V\_AbsTol=10mV$  and  $I\_AbsTol=100\mu A$ ). The circuit converged with

Order=127; but even when simulating with Order=511, the accuracy of the solution is not quite acceptable, as seen in the charge pump current plot below. The transient waveform, `tran_pump`, exhibits a real overshoot. While the HB waveform, `hb_pump`, tries to match this overshoot, due to the limited Order and large HB truncation error it exhibits significant Gibbs ripples:



## Harmonic Balance Assisted Harmonic Balance

When performing a multi-tone harmonic balance simulation, the simulator will decide automatically which tones to use in generating the final HB solution. This method is known as harmonic balance assisted harmonic balance (HBAHB). For example, instead of simulating a 3-tone circuit directly, it is possible to simulate a 1-tone, then use the result as an initial guess for a 2-tone, and finally use that result as an initial guess for the final 3-tone simulation. HBAHB is both fast and robust in particular 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 turned on within the simulator. The simulator will determine the optimal sequence of tones to simulate. Depending upon various conditions, the simulator may decide to first simulate using a 1-tone then use that result for a 3-tone, and skip the 2-tone simulation.

HBAHB can be found on the Initial Guess tab, in the section called Harmonic Balance Assisted Harmonic Balance. There are three choices - Off, On, and Auto. The default selection is Auto which means that the simulator automatically decides whether or not to perform HBAHB, and the optimal sequence of tones to simulate. The On selection means that HBAHB will always simulate each tone up to the last fundamental frequency. In other words, a simulation will be performed for each tone. The Off selection means that HBAHB will not be performed at all. When HBAHB is off, a standard multi-tone HB simulation will take place.

When using HBAHB, select the Auto convergence mode and the Auto solver type. Both of these parameters are found on the Solver tab of the HB controller. This ensures that simulating the intermediate tones will be both fast and robust, and not require user-

intervention to tweak simulation parameters.

If a user-supplied initial guess file is provided, then HBAHB will not be performed to generate the initial guess. However, if the supplied file does not exist, then the simulator will decide whether or not to perform HBAHB. You may also save the final result to a file for later use.

When simulating an HB variant such as noise, the HBAHB will take place prior to simulating noise, and the noise analysis will not be performed when simulating the intermediate tones. When performing a swept HB simulation, HBAHB will take place only for the first, inner-most sweep point.

At any point during the HBAHB simulation, if non-convergence occurs, then the simulator will switch back to performing a standard HB simulation using a DC initial guess as the starting point. In the case that TAHB was enabled, then the transient initial guess will be restored as the starting point.

## Changing the Tolerances

Sometimes a circuit may not converge because the tolerances are too tight. Adjusting the tolerances will help the Newton solver to achieve convergence. When using the Auto convergence mode, it is not necessary to change the tolerance levels. This is because the tolerances can be adjusted automatically by the HB simulator when a circuit is close to achieving convergence but cannot quite satisfy the default (or specified) tolerance levels. Alternatively, if the KCL residual for a circuit stagnates and cannot be further reduced, then the tolerance levels will be automatically adjusted for convergence. A descriptive warning message will be given in the status window when this occurs, and it will indicate the best tolerance level that was achieved for the given simulation setup. The message depends on which tolerance (relative or absolute) had more of an effect on the convergence criteria. Here is an example of the warning message when the current absolute tolerance is automatically adjusted to larger values by the HB simulator:

```
Warning detected by HPEESOFSSIM during HB analysis `HB1'.
  This is the best solution that can be achieved
    for the given simulation setup.
  The simulation has converged up to a current absolute
    tolerance of 3.49787 pA.
  The circuit was NOT able to achieve the target current
    absolute tolerance of 1 pA.
```

Here is an example of the warning message when the current relative tolerance is automatically adjusted to larger values by the HB simulator:

```
Warning detected by HPEESOFSSIM during HB analysis `HB1'.
  This is the best solution that can be achieved
    for the given simulation setup.
  The simulation has converged up to a current relative
    tolerance of 2.41e-06.
  The circuit was NOT able to achieve the target
```

current relative tolerance of 1.00e-06.

When using the Fast or Robust convergence mode as the Newton solver, monitor the KCL residual in the status server window, and adjust the tolerances accordingly. With these convergence modes, the HB simulator will not adjust the tolerance levels automatically. Consider the following example of a circuit that nearly converges to with a few picoamps, but not quite to the default current absolute tolerance of 1 picoamp:

```
HB HB1[1] <5335.ckt>   RFpower=(250e-03->4.5)
Number of frequencies:  11.
Number of time samples: 32.
Number of HB equations (problem size): 19866.
Convergence mode: Basic.
Linear solver: Krylov (GS_GMRES).
Preconditioner: DCP.
RFpower=250e-03      0.00% 1/18
```

```
-----
Newton solver:           Linear solver:
Iter  KCL residualSol update  ITERS  Residual
```

```
-----
0      125 uA
1      27.3103 uA          1      4.165e-10
2      394.436 nA         2      1.431e-02
3      553.875 pA         3      1.025e-03
4      2.3978 pA          4      7.406e-05
5      2.49853 pA         4      1.354e-04
6      2.49881 pA         3      3.217e-04
7      2.49852 pA         3      4.317e-04
8      2.49853 pA         3      4.109e-04
9      2.49853 pA         3      3.398e-04
10     2.4988 pA          3      3.910e-04
11     2.49853 pA         3      3.243e-04
12     2.52273 pA         3      3.616e-04
13     2.49853 pA         3      2.825e-04
14     2.52301 pA         3      2.630e-04
15     2.49853 pA         3      3.826e-04
16     2.49853 pA         3      2.669e-04
17     2.49853 pA         3      4.079e-04
18     2.49881 pA         3      4.320e-04
19     2.52302 pA         3      2.911e-04
```

Switching to source-stepping...

Attempting solution at `sourceLevel' value of 0.5:

```
0 62.5 uA
1 6.57904 uA          1      4.545e-10
2 24.3022 nA         2      3.600e-03
3 77.6941 pA         2      3.110e-03
4 3.41004 pA         3      1.138e-04
5 3.50342 pA         3      1.096e-04
6 3.50328 pA         2      9.598e-04
```

As a side note, when the maximum number of Newton iterations is reached, the simulator switches to a continuation method known as source stepping. In this method, the simulator decreases the input source levels and attempts to converge. If this is successful then the source levels are gradually increased to the final level.

By increasing the current absolute tolerance to 5 picoamps, convergence is achieved in



only a few iterations:

```
HB HB1[1] <5335.ckt>   RFpower=(250e-03->4.5)
Number of frequencies:  11.
Number of time samples: 32.
Number of HB equations (problem size): 19866.
Convergence mode: Basic.
Linear solver: Krylov (GS_GMRES).
Preconditioner: DCP.
RFpower=250e-03      0.00% 1/18
```

```
-----
Newton solver:                               Linear solver:
Iter   KCL residual   Sol update   ITERS   Residual
-----
0      125 uA
1      27.3103 uA           1      4.165e-10
2      394.436 nA          2      1.431e-02
3      553.883 pA          3      1.025e-03
4      2.39792 pA          4      7.406e-05
5      2.4988 pA           4.05239 pV  4      1.354e-04
      RFpower=500e-03      5.88% 2/18
0      125 uA
1      29.5154 uA           2      1.232e-02
2      1.47481 uA          2      4.910e-02
3      3.05574 nA          4      2.831e-03
4      3.95315 pA          5      2.187e-04
5      2.00489 pA          85.8831 pV  5      3.522e-04
      RFpower=750e-03      11.76% 3/18
0      125 uA
1      43.8438 uA           2      4.030e-02
2      1.08621 uA          4      2.400e-02
3      3.16865 nA          5      2.622e-03
4      4.6383 pA           6      7.839e-04
5      3.69573 pA          260.821 pV  6      6.887e-04
```

To adjust the tolerances, insert an Options controller on the schematic and go to the Convergence tab. Without an Options controller, the default tolerances for Harmonic Balance are set by the simulator. The following table shows the tolerances found on the Options controller, their description, and their default values.

Name	Description	Default Value
I_RelTol	Relative current tolerance	10 <sup>-6</sup>
V_RelTol	Relative voltage tolerance	10 <sup>-6</sup>
I_AbsTol	Absolute current tolerance	10-12A
V_AbsTol	Absolute voltage tolerance	10-6V

The controller allows for a choice of three tolerance presets: Strict, Intermediate, and Relaxed. The default tolerances shown above correspond to the strict preset. Looser tolerances will speed up the simulation run time, but may decrease the accuracy of the solution. (In a Transient simulation, the default RelTols are 10<sup>-3</sup>. Placing an Options controller with RelTols set to 10<sup>-6</sup> will slow down the simulation run time by about a factor of 10).

Some additional rules of thumb for adjusting tolerance parameters are:

- If the currents in the circuit are on the order of milliamps or amps, try increasing the relative current tolerance.
- If the currents in the circuit are on the order of microamps or smaller, try increasing the absolute current tolerance.
- If the status window output shows consecutive values in the Solution update (Sol update) column, then try increasing the voltage tolerances.

# ADS Dialog Boxes

The relevant ADS dialog box pages are shown below. All Harmonic Balance and Options parameters that apply:



## HARMONIC BALANCE

### HarmonicBalance

#### HB1

MaxOrder=4	ArcMinValue=	UseCompactFreqMap=	IV_RelTol=
Freq[1]=1.0 GHz	ArcMaxValue=	SweepVar=	AddtTranParamsTAHB=
Order[1]=3	UseAllSS_Freqs=	Start=1	OneToneTranTAHB=yes
StatusLevel=2	MergeSS_Freqs=	Stop=10	OutputTranDataTAHB=
FundOversample=	UseKrylov=auto	Step=1	HBAHB_Enable=Auto
Oversample[1]=	GMRES_Restart=	InFile=	SteadyStateMinTime=
PackFFT=	KrylovUsePacking=	UseInFile=	
MaxIters=	KrylovPackingThresh=	OutFile=	
GuardThresh=	KrylovTightTol=	UseOutFile=	
SamanskiiConstant=	KrylovLooseTol=	KrylovPrec=Auto	
Restart=no	KrylovLooselters=	ConvMode=Auto (Preferred)	
ArcLevelMaxStep=0.0	KrylovMaxIters=	Other=	
MaxStepRatio=100	AvailableRAMsize=	TAHB_Enable=Auto	
MaxShrinkage=1.0e-5	KrylovSS_Tol=	StopTime=	
ArcMaxStep=0.0	RecalculateWaveforms=	MaxTimeStep=	



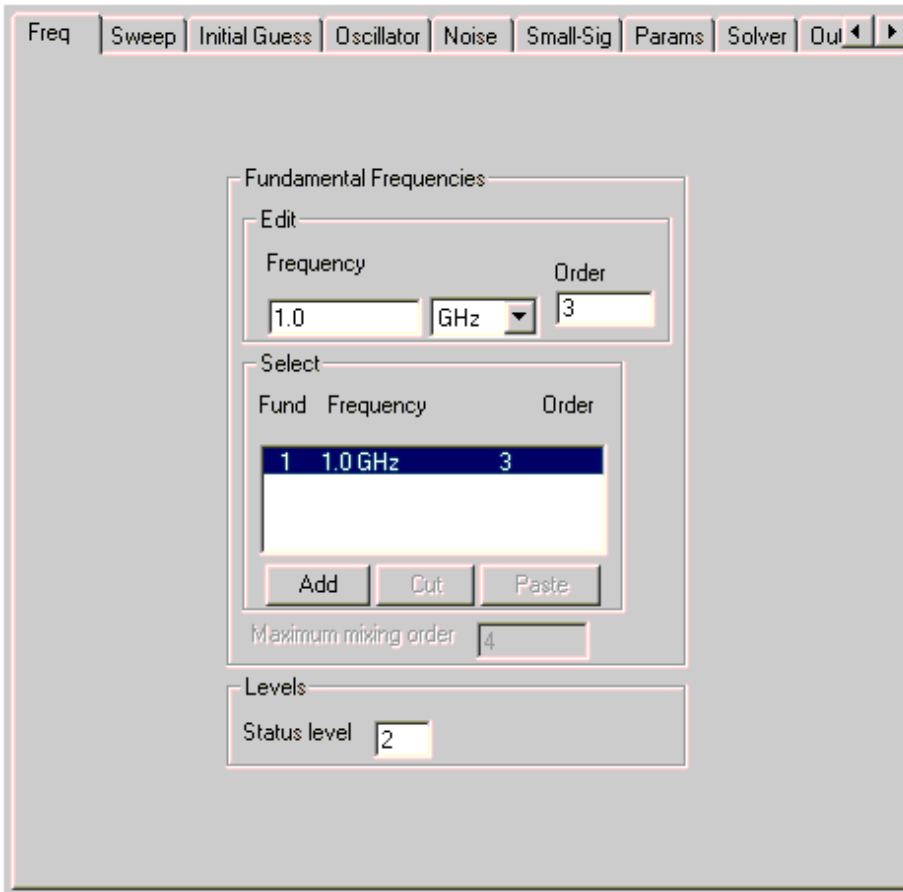
## OPTIONS

### Options

#### Options1

MaxDeltaV=  
 DC\_ConvMode=0  
 V\_RelTol=1e-6  
 V\_AbsTol=1e-6 V  
 I\_RelTol=1e-6  
 I\_AbsTol=1e-12 A  
 GiveAllWarnings=yes  
 MaxWarnings=10

### The Harmonic Balance Freq Tab



### The Harmonic Balance Sweep Tab

Freq Sweep Initial Guess Oscillator Noise Small-Sig Params Solver Out ◀ ▶

Parameter to sweep

Parameter sweep

Sweep Type

Start/Stop     Center/Span

Start

Stop

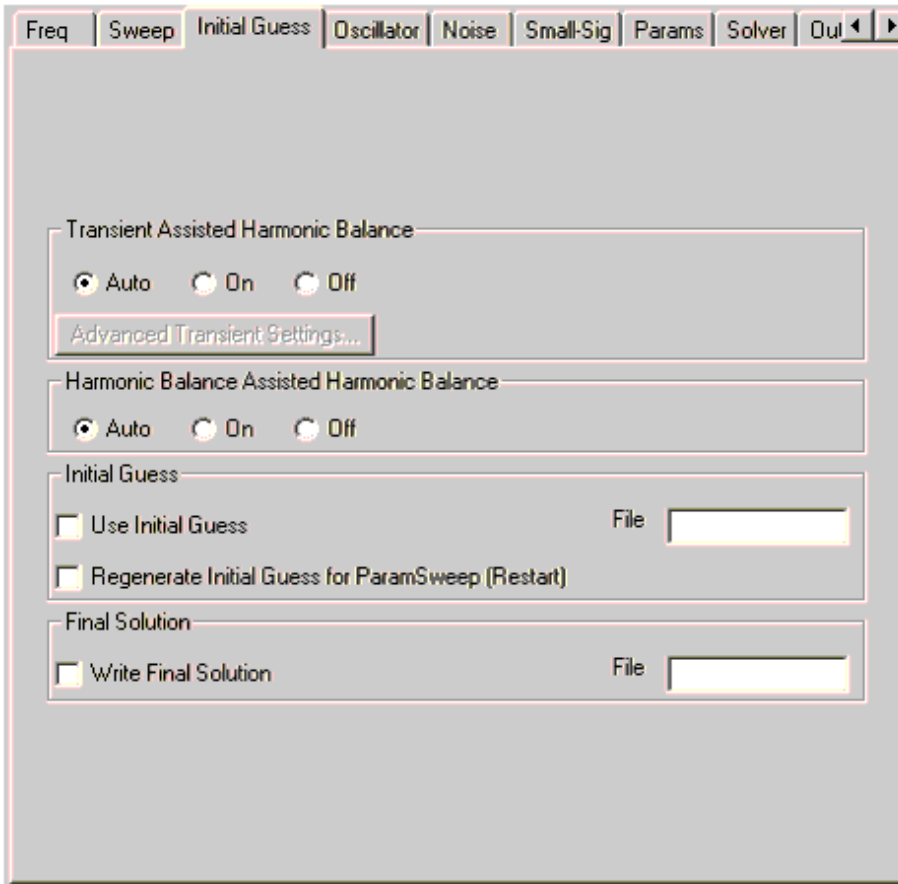
Step-size

Num. of pts.

Use sweep plan

### The Harmonic Balance Initial Guess Tab

**Note**  
 Setting Transient Assisted Harmonic Balance to *On* enables Advanced Transient Settings, which includes *Transient Stop Time*, *Transient Max Time Step*, *Min Time for detecting steady state*, *Transient IV\_RelTol*, *Transient Other*, *Use only Freq(1) for transient*, *Save transient data to dataset*.



## The Harmonic Balance Solver Tab

### Notes

- Setting Convergence Mode to *Advanced* or *Basic* enables Max. Iterations and Advanced Continuation Parameters. The Advanced Continuation Parameters include: *Arc Max Step*, *Arc Level Max Step*, *Arc Min Value*, *Arc Max Value*, *Max Step Ratio*, *Max Shrinkage*.
- Setting Solver Type to *Direct* enables Matrix Re-use, Matrix Bandwidth, and Waveform Memory Reduction.
- Setting Solver Type to *Krylov* enables Krylov Restart Length, Advanced Krylov Parameters, and Waveform Memory Reduction. The Advanced Krylov Parameters include: *Max Iterations*, *Krylov Noise Tolerance*, *Packing Threshold*, *Tight Tolerance*, *Loose Tolerance*, *Loose Iterations*, *Matrix Packing*, and the Preconditioner choices *Auto*, *DCP*, *BSP*, *SCP*.

Freq	Sweep	Initial Guess	Oscillator	Noise	Small-Sig	Params	Solver	Out	▶
------	-------	---------------	------------	-------	-----------	--------	--------	-----	---

**Convergence**

Convergence Mode:  Auto (Preferred)  Advanced (Robust)  Basic (Fast)

Max. Iterations:  Robust  Fast  Custom

[Advanced Continuation Parameters...](#)

---

**Matrix Solver**

Solver Type:  Auto Select  Direct  Krylov

Matrix Re-use:  Fast  Robust  Custom

Krylov Restart Length:  Robust  Low Memory  Custom

[Advanced Krylov Parameters...](#)

---

**Memory Management**

Matrix Bandwidth (GuardThresh):  Fast  Robust  Custom

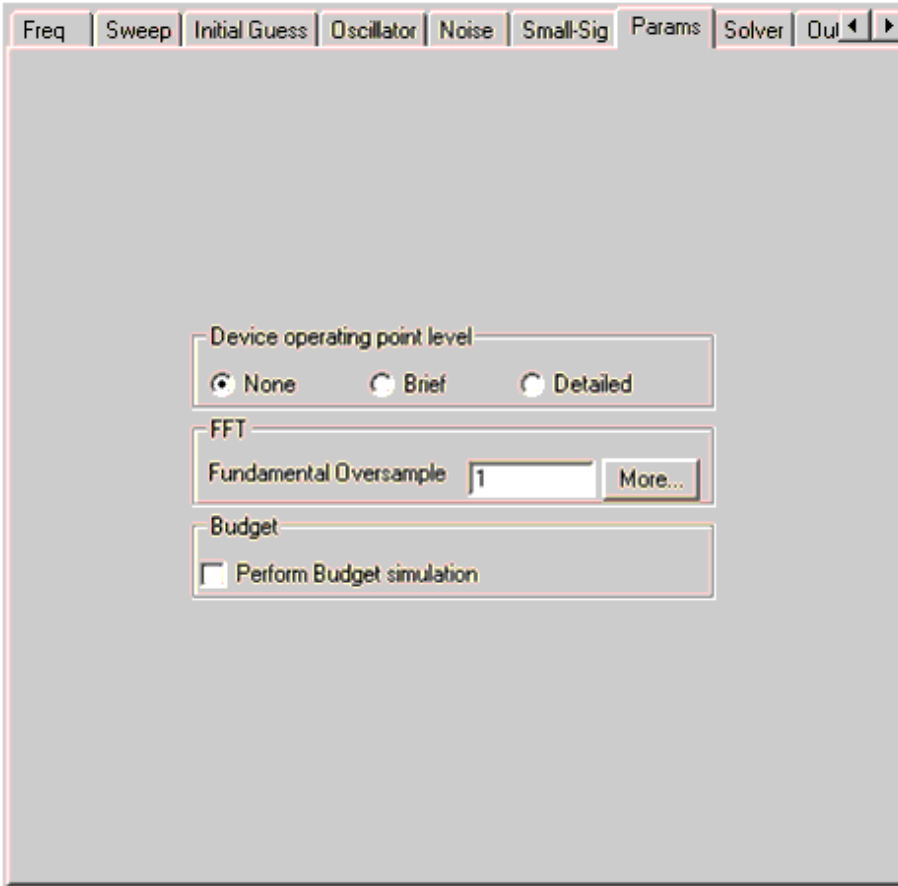
FFT Options:  Minimize memory and runtime  Minimize aliasing

Waveform Memory Reduction:

Use dynamic waveform recalculation

Use compact frequency map

### The Harmonic Balance Params Tab





## Additional Parameters

This section describes some additional parameters which may help nonconverging circuits. These parameters are for the outer (Newton) solver, the inner (Krylov) solver, arc-length continuation, and for memory issues.

### Convergence Mode

The Newton solver has additional settings (hidden parameters) that can be used to enhance its performance and assist convergence

- The hidden parameter **UseOptDamp** turns on the optimal damping search. The default is set to off, or 0. To turn this parameter on, set it to on, or 1. This parameter may be used with auto or advanced convergence modes.
- The parameter **RedRatio** is the KCL residual reduction threshold, default is 0.99. This is the ratio by which the norm of the solution vector must be reduced from one iteration to the next. This parameter has to be smaller than 1. Reducing RedRatio can accelerate the convergence rate at the cost of diminished robustness. This parameter may be used with auto or advanced convergence modes.
- The parameter **NormCheck** is the type of norm used in determining whether the KCL residual has converged. The default is set to 1, which corresponds to the L-1 norm. Other possible values include 0 (L-infinity norm), and 2 (L-2 norm). The L-infinity norm is the most conservative, followed by the L-2 and L-1 norms respectively. The L-2 and L-infinity norms are recommended for highly nonlinear large circuits. For a definition of these norms, consider the vector  $x$  of size  $n$ . Then, the L-1 norm is defined as

$$\|x\|_1 := |x_1| + \dots + |x_n|,$$

the L-2 norm is defined as

$$\|x\|_2 := (|x_1|^2 + \dots + |x_n|^2)^{1/2},$$

and the L-infinity norm is defined as

$$\|x\|_{\text{inf}} := \max |x_i|, \text{ for } 1 \leq i \leq n.$$

This parameter may be used with auto, advanced, or basic convergence modes.

### Krylov Solver

Additional Krylov parameters can be found on the HB controller's Solver tab after clicking *Advanced Krylov Parameters*. It is strongly recommended to use caution when adjusting these parameters, as they are very specific to the internal algorithms.

The parameter **Maximum number of iterations** (KrylovMaxIters on schematic) is the maximum number of Krylov solver iterations allowed. The default is intentionally set to a large value of 150 to accommodate even slowly convergent iterations. Increase this number in cases where poor convergence may be improved and an increase in simulation

run time is acceptable.

The parameter **Krylov Restart Length** (GMRES\_Restart on schematic) sets the number of iterations after which the Krylov linear solver restarts. At this point the algorithm does not need data from previous steps, and the corresponding memory is released. Thus, smaller values lead to lower memory requirements, but might significantly affect convergence. The default is set to Robust, and it is strongly recommended that you avoid decreasing this value unless the problem is extremely large and convergence is carefully monitored. Larger values offer potentially more robust performance, but require more memory. If there are memory constraints, then set this value to 50 or lower by entering the value in the Custom dialog box entry.

The solver achieves full convergence if the Krylov solver residual is less than the tight tolerance, which is set by the parameter **Tight Tolerance** (default value of 0.001). After the number of iterations specified by the parameter **Loose Iterations** (default value of 50), the solver then uses parameter **Loose Tolerance** to achieve partial convergence. The loose tolerance has default value of 0.1. The Krylov solver fails to converge if the residual reduction factor from two adjacent iterations becomes larger than the parameter **KrylovConvRatio**, which has default set to 0.9. The KrylovConvRatio is a hidden parameter, whereas the other three mentioned above are in the HB controller's Solver tab by clicking *Advanced Krylov Parameters*.

A hidden parameter for the DCP preconditioner:

- **PrecMatrixSolver** 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:
  - PrecMatrixSolver=0 (default, the simulator automatically chooses the solver)
  - PrecMatrixSolver=1 (use the general solver)
  - PrecMatrixSolver=2 (use the large-circuit solver)

Some hidden parameters for the BSP and SCP preconditioners:

- **PrecRhsThresh** activate BSP or SCP if Newton residual is smaller than this threshold (default=0.05)
- **\_ScpSchurSolver** selects the Schur solver in SCP, 0=DMRES (default), 1=GMRES
- **\_ScpReuse** 1=re-use the SCP at each Newton iteration (default), 0=reload the SCP at each Newton iteration
- **\_ScpTol** inner SCP Schur solver tolerance (default=0.001)
- **\_ScpStartIter** use SCP from this Newton iteration onward (default=0)

## Arc Length Continuation

The simulator will switch over to arc-length continuation method when it is having a difficult time converging using the Direct solver. This algorithm is very robust. If the simulator goes into this method, it is often the case that the circuit has instabilities or multiple solutions. It is recommended to try all other convergence remedies first before

adjusting arc length parameters. The arc length parameters can be found on the HB solver tab under Advanced Continuation Parameters. The **Max Step Ratio** controls the maximum number of continuation steps (default is 100). **Max Step Shrinkage** controls the minimum size of the arc-length parameter step (default is 1e-5). The **Arc Max Step** will limit the maximum size of the arc-length parameter step during arc-length continuation. The default is 0 which means there is no limit for the Arc Max Step. Analogously, the **Arc Level Max Step** will limit the maximum size of the arc-length step during source level stepping. The default is 0 which means there is no limit for the Arc Level Max Step. Note that arc-length continuation occurs when using the direct solver and source level stepping occurs when using the Krylov solver. **Arc Min Value** and **Arc Max Value** determine the allowed range of the continuation parameter  $p$  during the simulation. The defaults are  $p_{min} - \text{delta}$  and  $p_{max} + \text{delta}$ , respectively, where  $\text{delta}$  is  $p_{max} - p_{min}$ .  $p_{min}$  is lower end of the parameter sweep, and  $p_{max}$  is the upper end of the parameter sweep.

As an example, consider a transceiver circuit in a Harmonic Balance simulation with an input power sweep from -20 dBm to 3 dBm. That circuit simulation fails with the following arc-length message:

```
Value of `pin' went out of range during arc length continuation. The range is
-43 to +26. Try explicitly specifying the range with ArcMin Value and
ArcMaxValue and re-simulating.
```

During arc-length continuation, the continuation parameter (in this case, pin input power) may get out of the allowed range. The allowed range is from -43 dBm to 26 dBm. In this particular case,  $\text{delta} = 3 - (-20) = 23$ ,  $\text{Arc Min Value} = -20 - 23 = -43$ , and  $\text{Arc Max Value} = 3 + 23 = 26$ . The convergence remedy is to extend the continuation parameter range limits by setting Arc Min Value and Arc Max Value.

## Memory Requirements

Solving a circuit with many nonlinear devices, harmonics, and tones, requires a substantial amount of memory. Please make sure the Krylov linear solver is used when simulating such circuits. There are some additional parameters that can be set to reduce the amount of required memory such as Matrix Packing, PackFFT, Recalculate waveforms, and UseCompactFreqMap.

**Matrix Packing** forces the solver to use the technique known as spectral packing, which reduces the memory needed for the Krylov solver, typically by 60-80%. By default, this feature is turned off. It is recommended to turn this on for extremely large problems in which the available RAM would not be able to accommodate the Krylov solver memory requirements. To enable this parameter, check the box next to Matrix Packing on the Solver tab under Advanced Krylov Parameters. In conjunction with Matrix Packing is the parameter **PackingThreshold** which sets the bandwidth threshold for the packing. The default value is 1e-8. Set this to a larger value to increase the memory reduction.

**FFT Options** controls the frequency map packing for multi-tone Harmonic Balance. This

parameter can be selected to either minimize aliasing or minimize memory usage and simulation run time. By default the simulator sets the parameter to minimize memory usage and simulation run time. Minimizing the memory usage 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. For mixers and other applications with a single large dominant (LO) tone, that frequency should be assigned to the first fundamental and FFT Options should be set to minimize aliasing so that any aliased harmonics of this large signal will just land on its own harmonics and not on mixing terms. If you are simulating mixer intermodulation or third-order intercept, it is recommended that this parameter also be set to minimize aliasing to achieve the most accurate results.

Both use **dynamic waveform recalculation** and use **compact frequency map** for memory reduction. When using neither of these, all nodal waveforms are stored. The advantage is that it speeds up simulation time since the waveforms do not need to be recalculated. However, the disadvantage is that storing all the nodal waveforms causes a high memory consumption.

The parameter Use dynamic waveform recalculation (found on the Solver tab of the HB controller) enables reuse of dynamic waveform memory instead of up front storage on all waveforms. By enabling this parameter, only the needed waveforms are stored which requires less memory. However, if two devices are sharing the same node, then the waveform for the second node would need to be recomputed and therefore result in an increase in simulation time. Small circuits might simulate a little slower, but not significantly.

Use compact frequency map (found on the HB controller's Solver tab) enables a spectral compression, typically requiring less memory for individual waveforms. If the memory required for the Krylov solver is greater than the available RAM, then this parameter will get set to yes by the simulator and a warning message will be displayed in the status window. By enabling this parameter, simulation speed will increase, yet there is less memory reduction compared to using RecalculateWaveforms.

# Parameter Index

Display Name on Schematic	UI Tab	UI Name
AddtlTranParamsTAHB	Initial Guess	Advanced Transient Settings - Transient Other
ArcLevelMaxStep	Solver	Advanced Continuation Parameters - Arc Level Max Step
ArcMaxStep	Solver	Advanced Continuation Parameters - Arc Max Step
ArcMaxValue	Solver	Advanced Continuation Parameters - Arc Max Value
ArcMinValue	Solver	Advanced Continuation Parameters - Arc Min Value
AvailableRAMsize	Display	Only editable on schematic
ConvMode	Solver	Convergence Mode (In Convergence Section, Check box)
Freq[i]	Freq	Frequency (In Edit Section)
FundOversample	Params	Fundamental Oversample (In FFT section)
GMRES_Restart	Solver	Advanced Krylov Parameters - Krylov Restart Length
GuardThresh	Solver	Matrix Bandwidth (In Memory Management section)
HBAHB_Enable	Initial Guess	Harmonic Balance Assisted Harmonic Balance
InFile	Initial Guess	File (Initial Guess section)
IV_RelTol	Initial Guess	Advanced Transient Settings - Transient IV_RelTol
KrylovLooseIters	Solver	Advanced Krylov Parameters - Loose Iterations
KrylovLooseTol	Solver	Advanced Krylov Parameters - Loose Tolerance
KrylovMaxIters	Solver	Advanced Krylov Parameters - Max Iterations
KrylovPackingThresh	Solver	Advanced Krylov Parameters - Packing Threshold
KrylovPrec	Solver	Advanced Krylov Parameters - Preconditioner
KrylovSS_Tol	Solver	Advanced Krylov Parameters - Krylov Noise Tolerance
KrylovTightTol	Solver	Advanced Krylov Parameters - Tight Tolerance
KrylovUsePacking	Solver	Advanced Krylov Parameters - Matrix Packing (check box)
MaxIters	Solver	Max Iterations (In Convergence section)
MaxOrder	Freq	Maximum mixing order
MaxShrinkage	Solver	Advanced Krylov Parameters - Max Step Shrinkage
MaxStepRatio	Solver	Advanced Krylov Parameters - Max Step Ratio
MaxTimeStep	Initial Guess	Advanced Transient Settings - Transient Max Time Step
MergeSS_Freqs	Small-Sig	Merge small- and large-signal frequencies
OneToneTranTAHB	Initial Guess	Advanced Transient Settings - Use only Freq(1) for transient
Order[i]	Freq	Order (in Edit section)
Other	Display	Other
OutFile	Initial Guess	File (Final Solution section)
OutputTranDataTAHB	Initial Guess	Advanced Transient Settings - Save transient data to dataset

Oversample[i]	Params	Oversample (in FFT section, click more)
PackFFT	Solver	FFT Options (In Memory Management section)
RecalculateWaveforms	Solver	Use Dynamic Waveform recalculation (In Memory Management section)
Restart	Initial Guess	Regenerate Initial Guess for ParamSweep (check box)
SamanskiiConstant	Solver	Matrix Re-use (Matrix Solver section)
Start	Sweep	Start
StatusLevel	Freq	Status Level (Levels section)
SteadyStateMinTime	Initial Guess	Advanced Transient Settings - Min Time for detecting steady state
Step	Sweep	Step-size
Stop	Sweep	Stop
StopTime	Initial Guess	Advanced Transient Settings - Transient Stop Time
SweepVar	Sweep	Parameter to sweep
TAHB_Enable	Initial Guess	Transient Assisted Harmonic Balance
UseAllSS_Freqs	Small-Sig	Use all small-signal frequencies
UseCompactFreqMap	Solver	Use compact frequency map (in Memory Management section)
UseInFile	InitialGuess	Use Initial Guess
UseKrylov	Solver	Solver Type
UseOutFile	Initial Guess	Write Final Solution

# Harmonic Balance Background

Harmonic Balance is a frequency domain analysis technique for simulating nonlinear circuits and systems. This method assumes the input stimulus consists of a relatively few steady state sinusoids. Therefore the solution can be expressed as a sum of steady state sinusoids that includes the input frequencies in addition to any significant harmonics or mixing terms.

A circuit with a single input source will require a single tone HB simulation. A solution waveform (e.g. the node voltage  $v(t)$ ) in a single tone HB simulation is approximated as follows:

$$v(t) = \text{Real} \left\{ \sum_{k=0}^K V_k e^{j2\pi k f t} \right\}$$

where  $f$  is the fundamental frequency of the source, the  $V_k$ 's are the complex Fourier coefficients that the HB analysis computes, and  $K$  is the level of truncation (number of harmonics).

A circuit with multiple input sources will require a multitone HB simulation. In this case, the steady state solution waveforms are approximated with a multidimensional truncated Fourier series as follows:

$$v(t) = \text{Real} \left\{ \sum_{k_1=0}^{K_1} \sum_{k_2=0}^{K_2} \dots \sum_{k_n=0}^{K_n} V_{k_1, k_2, \dots, k_n} e^{j2\pi(k_1 f_1 + \dots + k_n f_n)t} \right\}$$

where  $n$  is the number of tones (sources),  $f_{1\dots n}$  are the fundamental frequencies of each source, and  $K_{1\dots n}$  are the number of harmonics for each tone. When having multiple tones in a circuit, mixing products will occur.

The circuit simulator converts  $N$  nonlinear differential equations (where  $N$  is the size of the circuit, i.e., the number of nodes and branch currents) into the frequency domain, where it becomes a set of  $N \cdot M$  nonlinear algebraic equations (where  $M$  is the total number of frequencies including the input frequencies, their harmonics, and mixing terms), as shown below:

$$g(v(t)) + \frac{d}{dt}q(v(t)) + y(t) \otimes v(t) = i(t)$$

$$F_k \{g(v(t))\} + j\omega_k F_k \{q(v(t))\} + Y(j\omega_k) V_k = I(\omega_k)$$

where  $F_k$  is the  $k^{\text{th}}$  spectral component of a Fourier transformation, and  $\omega_k = 2\pi f \cdot k$ . The harmonic balance simulator then must simultaneously solve this set of  $N \cdot M$  nonlinear algebraic equations for the  $V_k$  values. The number of nonlinear equations that must be solved has increased by a factor of  $M$  compared to standard time domain simulators. This means that the matrix sizes and memory requirements of harmonic balance increase considerably as  $M$  becomes large. The nonlinear devices are still evaluated in the time domain by using an inverse Fourier transformation to convert the  $V_k$  values into the  $v(t)$

waveform prior to evaluating the nonlinear  $q()$  and  $g()$  functions. The current and nonlinear charge waveforms are transformed into the frequency domain at each iteration so their spectral values can be used in the frequency domain equations. Since the HB simulator uses the Newton's method, the derivatives (nonlinear resistance and capacitance) must also be computed in the time domain and transformed into the frequency domain.

The primary advantage of harmonic balance over time domain solutions is that the linear devices with arbitrary frequency responses can be easily, yet quickly, modeled. Lumped element approximations are no longer required. Time domain convolution has been replaced with simple frequency domain multiplication. This is especially important for RF, microwave and millimeter frequencies, which are often characterized with measured frequency data. An additional benefit is that harmonic balance solutions directly provide the steady state solution without having to wait until the transient solution dies out. For high Q circuits this can be a costly wait. The input stimulus frequencies,  $f$ , can also be arbitrarily widely spaced and may actually be non-commensurate, but the harmonic balance solution can still be quickly obtained. The complexity and cost of the solution does not increase just because there is a low frequency tone (a long period) coexisting with high frequency tones (very small time steps).

The limitations of harmonic balance are that the signal must be quasi-periodic; they must be representable as a sum of a relatively few 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  $M$ ). The Krylov solvers therefore enable harmonic balance to be used on very large circuits and circuits with a large number of tones.

Harmonic balance is usually the method of choice for simulating analog RF and microwave problems, since these are most naturally handled in the frequency domain. Examples of devices and circuits suited to this analysis include power amplifiers, frequency multipliers, mixers, and modulators under large signal sinusoidal drive. In the context of high frequency circuit and system simulation, harmonic balance has a number of advantages over conventional time-domain analysis:

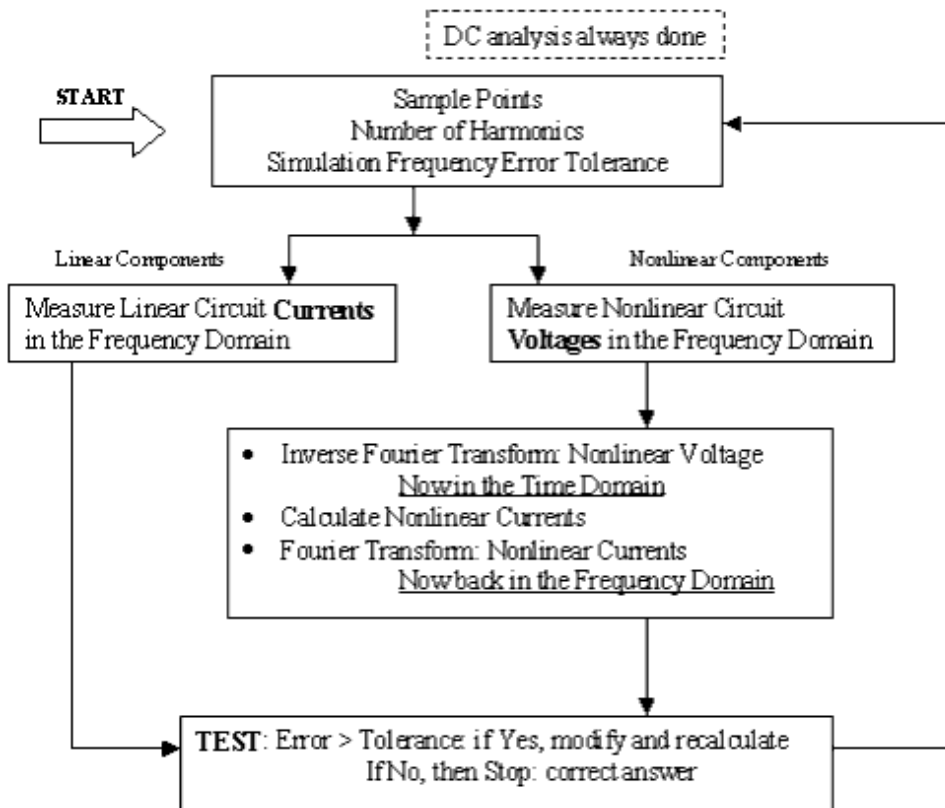
- Designers are usually most interested in a system's steady state behavior. Many high frequency circuits contain long time constants that require conventional transient methods to integrate over many periods of the lowest-frequency. Transient analysis would require integration over an enormous number of periods on the highest-frequency sinusoid.
- The applied voltage sources are typically multitone sinusoids that may have very narrow 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 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.



## How the Harmonic Balance Simulator Operates

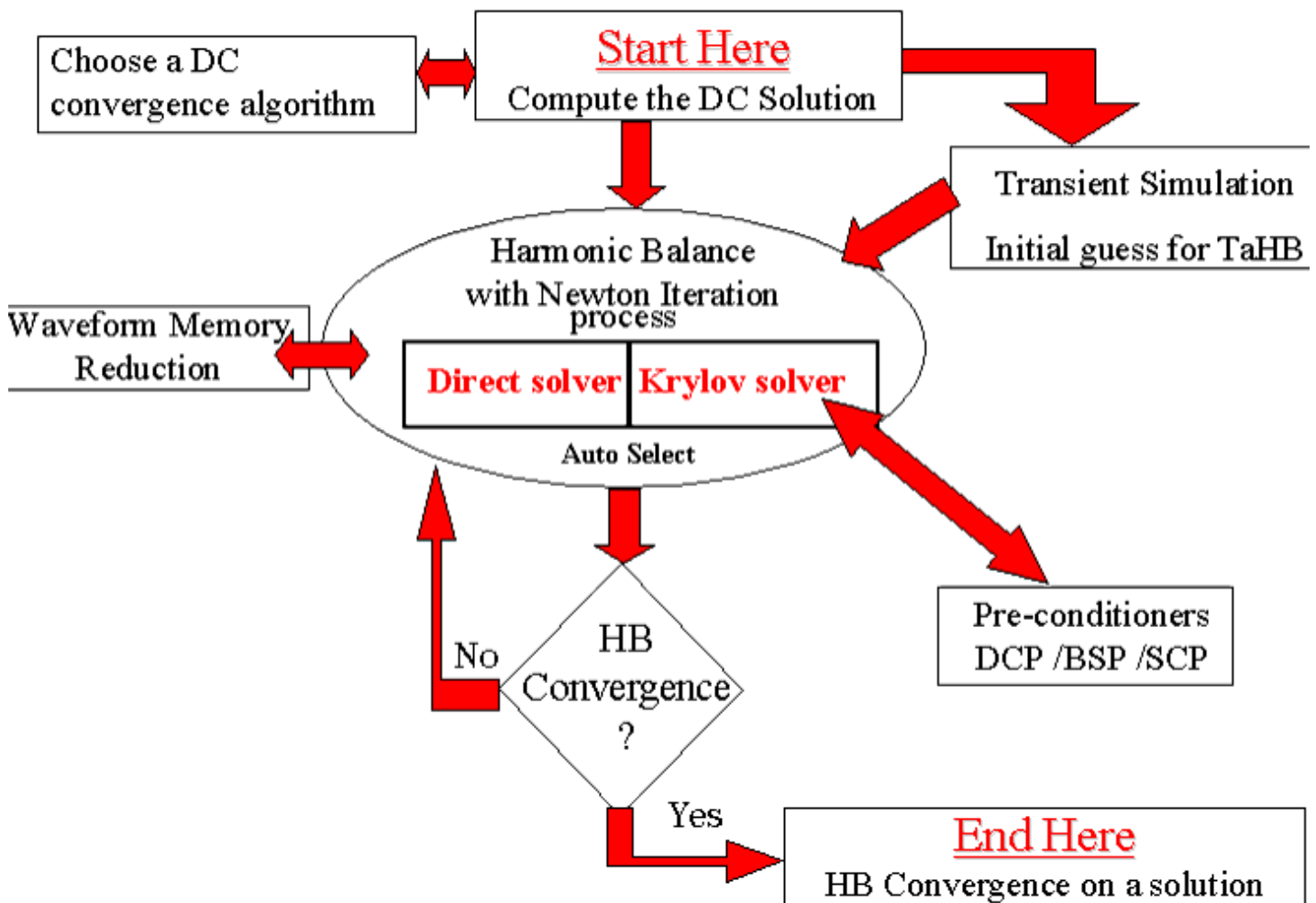
Harmonic balance computes the steady state response of nonlinear circuits excited by single or multiple periodic sources. The harmonic balance method is iterative. It is based on the assumption that for a given sinusoidal excitation there exists a steady-state solution that can be approximated to satisfactory 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 straightforward frequency-domain linear analysis. Currents from nodes into nonlinear elements are calculated in the time-domain. A frequency-domain representation of all currents flowing away from all nodes is available. According to Kirchhoff's Current Law (KCL), the 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 penalized to adjust the voltage amplitudes and phases. If the method converges (that is, the error function is driven to a given small value), then the resulting voltage amplitude and phases approximate the steady-state solution. The diagram below is a flow chart to demonstrate the harmonic balance method.



The following diagram gives a global overview of the HB simulator in ADS.

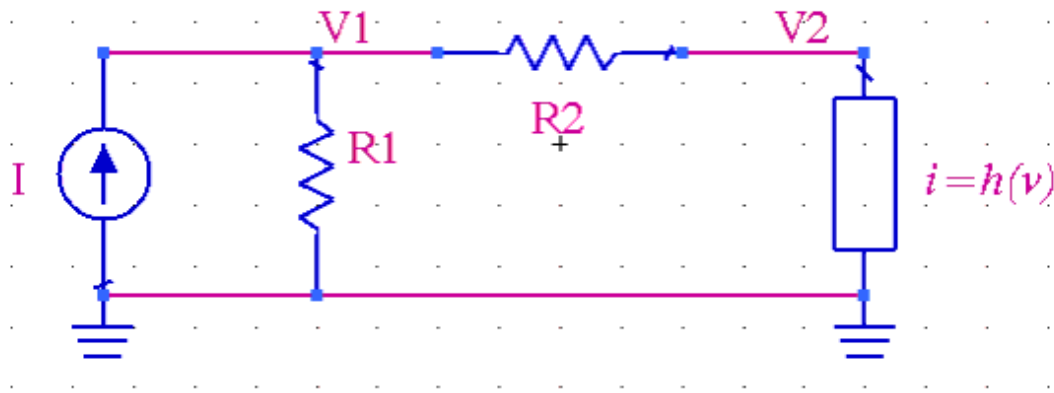
## HB Frequency-domain Circuit Simulation Flow



## Newton's Method

Mathematical theory has shown that there are simple formulas for solving linear and quadratic algebraic equations. However, there are no specific formulas for solving quintic or higher order, non-linear algebraic equations analytically. There are, however, several numerical techniques for solving these types of equations. One of these is Newton's method. In this method, the first step is to make a guess for the root of the equation  $f(x)=0$ . Then use that approximation to get a second. A second to get a third, and so on.

Harmonic Balance uses Newton's method to solve a system of nonlinear algebraic equations, by starting with an initial guess and repeatedly solving the iteration equation. This is done until some convergence criteria are met. Consider the following circuit in which the node voltages and branch currents are solved for using Newton's method.



Using KCL, the following nonlinear algebraic equations are derived:

$$I = V_1 G_1 + (V_1 - V_2) G_2$$

$$(V_1 - V_2) G_2 = h(V_2)$$

For this circuit, let  $I=1\text{A}$ ,  $R_1=6\text{ Ohms}$ , and  $R_2=4\text{ Ohms}$ . The nonlinear resistor  $i$ - $v$  relationship is given as:

$$i = h(v) = \frac{1}{6}v^3 + \frac{1}{12}v^2 + \frac{1}{4}v$$

We will use Newton's method to solve the above system of equations to determine the voltage for nodes  $V_1$  and  $V_2$ . This is an iterative method defined by the following equation:

$$v^{(k+1)} = v^{(k)} - J^{-1}(v^{(k)})f(v^{(k)})$$

where  $v$  is the vector of node voltages,  $f$  is the vector of nodal equations, and  $J$  is the Jacobian of the vector  $f$  at  $v$ . The Jacobian represents the linearized circuit and is defined below:

$$v = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \quad v^{(0)} = \begin{bmatrix} V_1^{(0)} \\ V_2^{(0)} \end{bmatrix} \quad f = \begin{bmatrix} f_1(V_1, V_2) \\ f_2(V_1, V_2) \end{bmatrix} \quad J(v) = \begin{bmatrix} \frac{\partial f_1}{\partial V_1} & \frac{\partial f_1}{\partial V_2} \\ \frac{\partial f_2}{\partial V_1} & \frac{\partial f_2}{\partial V_2} \end{bmatrix}$$

For this problem, we write the circuit equations for  $f$  as follows:

$$f_1(V_1, V_2) = V_1(G_1 + G_2) - V_2 G_2 - I = 0$$

$$f_2(V_1, V_2) = V_1 G_2 - V_2 G_2 - h(V_2) = 0$$

Next the Jacobian is determined:

$$\begin{bmatrix} G_1 + G_2 & -G_2 \\ G_2 & -G_2 - \left( \frac{1}{2}V_2^2 + \frac{1}{6}V_2 + \frac{1}{4} \right) \end{bmatrix} = J(V)$$

Start with the initial guess:

$$\mathbf{v}^{(0)} = \begin{bmatrix} \mathbf{V}_1^{(0)} \\ \mathbf{V}_2^{(0)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In Newton's method, we start with the initial guess,  $\mathbf{v}^{(0)}$  and compute  $f(\mathbf{v}^{(0)})$  and  $J(\mathbf{v}^{(0)})$  to compute  $\Delta\mathbf{v}=\mathbf{v}^{(1)} -\mathbf{v}^{(0)}$ , from which we obtain  $\mathbf{v}^{(1)}$ . The next step is to compute  $f(\mathbf{v}^{(1)})$  and check the convergence for  $f(\mathbf{v}^{(1)})$  and  $\Delta\mathbf{v}$ . If both have converged to within tolerance limits, then the iterations stop and we are done. If convergence was not obtained, then  $J(\mathbf{v}^{(1)})$  is computed and used to determine  $\mathbf{v}^{(2)}$ . Then  $f(\mathbf{v}^{(2)})$  and  $\Delta\mathbf{v}=\mathbf{v}^{(2)} -\mathbf{v}^{(1)}$  are computed and checked for convergence. If convergence is achieved, then Newton's method is complete, otherwise, the Jacobian is computed and the iterations continue. This process is carried out until both of the convergence criteria are met, namely the KCL residual is less than 1pA and the  $\Delta\mathbf{v}$  is less than 1  $\mu\text{V}$ . The values at each iteration are shown in the table below:

Iteration $k$	$\mathbf{v}^{(k)}$	$J(\mathbf{v}^{(k)})$	$f(\mathbf{v}^{(k)})$	$\mathbf{v}^{(k+1)} -\mathbf{v}^{(k)}$
0	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -0.50 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 3.4286 \\ 1.7143 \end{bmatrix}$
1	$\begin{bmatrix} 3.4286 \\ 1.7143 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -2.2551 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.0845 \end{bmatrix}$	$\begin{bmatrix} -0.3091 \\ -0.5152 \end{bmatrix}$
2	$\begin{bmatrix} 3.1195 \\ 1.1991 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -1.4188 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.2268 \end{bmatrix}$	$\begin{bmatrix} -0.1073 \\ -0.1788 \end{bmatrix}$
3	$\begin{bmatrix} 3.0122 \\ 1.0203 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -1.1906 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.0209 \end{bmatrix}$	$\begin{bmatrix} -0.0120 \\ -0.0201 \end{bmatrix}$
4	$\begin{bmatrix} 3.0001 \\ 1.0002 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -1.1669 \end{bmatrix}$	$\begin{bmatrix} 0 \times 10^{-3} \\ -0.2375 \times 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -1.401 \times 10^{-4} \\ -2.336 \times 10^{-4} \end{bmatrix}$
5	$\begin{bmatrix} 3.0000 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -1.1667 \end{bmatrix}$	$\begin{bmatrix} 0 \times 10^{-7} \\ -0.3183 \times 10^{-7} \end{bmatrix}$	$\begin{bmatrix} -1.8785 \times 10^{-8} \\ -3.1308 \times 10^{-8} \end{bmatrix}$
6	$\begin{bmatrix} 3.0000 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} 0.4167 & -0.25 \\ 0.25 & -1.1667 \end{bmatrix}$	$\begin{bmatrix} 0 \times 10^{-15} \\ -0.5551 \times 10^{-15} \end{bmatrix}$	

For this example, it took 6 iterations to arrive at acceptable solutions for  $V_1 = 3\text{V}$  and  $V_2 = 1\text{V}$ . The column labeled  $f(\mathbf{v}^{(k)})$  is the KCL residual and the column labeled  $\mathbf{v}^{(k+1)} -\mathbf{v}^{(k)}$  is the solution update, or  $\Delta\mathbf{v}$ . It is seen from the 6th iteration that the KCL residual is less than 1 pA and that  $\Delta\mathbf{v}$  is less than 1  $\mu\text{V}$ . Convergence was achieved for  $\Delta\mathbf{v}$  at the end of the 5th iteration.

Newton's method can be understood from a graphical point of view as well. The single dimensional case is shown in the diagram below. It is desired to solve for  $v^*$ , such that  $f(v^*) = 0$ . For this method, the first step is to make an initial guess,  $v^{(0)}$ , then linearize about  $v^{(0)}$ , then solve for the next guess  $v^{(1)}$ , and so on. If all goes well, as the  $k$  becomes sufficiently large,  $v^{(k)}$  will asymptotically approach the solution  $v^*$ .

