



February 2011
Tuning, Optimization, and Statistical Design

© 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

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

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 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).

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Tuning in Advanced Design System	7
Tuning Benefits	7
Basic Tuning Procedure	7
Tunable and Non-Tunable Parameters	8
Choosing Tuning Parameters	8
Setting Up Parameters Outside of Tuning	8
Tuning Parameters	11
Tuning Examples	15
About Nominal Optimization	22
Introduction	22
Features	23
Minimum Requirements	23
Initiation	24
Single Optimization	25
Introduction	25
Setup	25
Running Optimization	35
Updating Design on the Design	35
Advanced Optimization	36
Swept Optimization Methodology	36
Final Analysis	36
Sequential Optimization Methodology	37
Programmable Optimization Methodology	37
Optimization Cockpit	38
Cockpit Panels	38
Controlling the Optimization	42
Scaling the plots	44
Tuning	45
View-only mode	47
Turning off the Optimization Cockpit	48
Summary of Optimizers	49
Obtaining Global Optimal Results	49
Available Optimizers	49
Error-Function (EF) Formulation	56
Search Methods	61
Sensitivity Analysis	63
Cascaded Optimization Strategies	64
Optimization Examples	65
Single Optimization Example	65
Final Analysis Example	74
Swept Optimization Example	75
Programmable Optimization Example	77
Single Optimization Cockpit Example	79
Swept Optimization Cockpit Example	84
Troubleshooting	86
Understanding How ADS Names Optimization Variables	86
Understanding Optimization Variable Types	86
Failures that Occur in Evaluating Goals	86
Global vs. Local Search	87
Improving Optimization Speed	87
Meeting Troubles in the Cockpit Running Mode	87
Using Statistical Design	88
Statistical Design Minimum Requirements	88
Performing Yield Analysis	89
Specifying Multiple Parameters for Yield Analysis	90
Creating a Measurement Histogram	97
Creating a Sensitivity Histogram	100
Performing Monte Carlo Analysis	101
Performing Yield Optimization	105
Statistical Correlation	110
Using Design of Experiments (DOE)	112
DOE and Computer Simulation	112
Minimum DOE Requirements	112
Specifying Component Parameters for DOE	112
Specifying Multiple Component Parameters for DOE	113
Placing a Simulation Control Component for DOE	114
Setting DOE Goals	114
Setting Job Parameters for DOE	115
Initiating Design of Experiments	118
DOE Terminology	118
DOE Concepts	119
DOE Outputs	123
DOE Basic Example	124
Optimizing Using DOE Outputs	129
Performing the DOE Confirmation Experiment	132
Analyzing the DOE Confirmation Experiment	132
DOE References	133
Available Value Types	134

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Value Types for Nominal Optimization	134
Value Types for Statistical Design	135
Value Types for DOE	135
Using Monte Carlo Yield Analysis	136
Monte Carlo Trials and Confidence Levels	136
Confidence Tables	138

Tuning in Advanced Design System

Advanced Design System's tuning capability enables you to change one or more design parameter values and quickly see its effect on the output without re-simulating the entire design. Multiple traces generated from various tuning trials can be overlaid in the Data Display window. This can help you find the best results and the most sensitive components or parameters more easily.

Note

The tuning feature described in this documentation is available for Advanced Design System and RFIC Dynamic Link schematic only.

This topic includes:

- An explanation of tuning and the benefits of tuning
- A basic tuning procedure
- A description of the tune syntax
- Information on tuning hierarchical networks
- Information on tunable and non-tunable parameters
- A detailed description of each tuning dialog box
- A tuning example for analog/RF systems
- A tuning example for signal processing

When you analyze a network (Simulate > Simulate), a considerable amount of information is compiled by the simulator prior to the actual network simulation. The simulator must set up your network topology, load all the values of the component parameters, and organize your measurement requests.

With the ADS tuning features, you can avoid repeating the pre-processing. Tuning performs the pre-processing once and then assumes that you are now just trying to change some of the parameter values. A new simulation will take place, but using the same network topology and list of measurements. Only the small changes regarding the new parameter values are needed. You can tune a large number of components, including those that are processed by a measurement equation component, such as VSWR.

Before you get started tuning, you can optionally set up your tuning preferences by choosing the **Options > Preferences** menu item and then selecting the *Tuning* tab. For more information, *Setting Tuning Options* (custom).

Tuning Benefits

The benefits of the ADS tuning feature include:

- A reduction in total simulation time (by avoiding the pre-processing step).
- The ability to view the effects of changing parameter values. Note that parameter values can easily be stored and recalled in order to compare one set of parameter values to another. A memory trace is automatically plotted in the Data Display which can be used as a reference for comparing results. This can help you understand or locate the sensitive components or parameters in your design.
- A convenient means to quickly enter new parameter values.

Basic Tuning Procedure

Before starting any tuning session, you must first meet the prerequisites listed below.

Tuning Prerequisites

1. Build your design.
2. Set up your simulation.
3. Simulate your design and verify that your simulation operates as expected.
4. Set up, display, and analyze your results in the Data Display window.

Basic Tuning Procedure

The basic tuning procedure consists of the following steps:

1. *Start Tuning* - Start the tuning application by choosing the **Simulate > Tuning** menu item or click the Tune Parameters icon. For more information, refer to *Tuning Parameters*.
2. *Select Parameters* - Click the parameter(s) you want to tune. For more information, refer to *Setting Up Parameters Inside of Tuning*.
3. *Tune Parameters* - Move the sliders or click the up or down arrow to tune a parameter. For more information, refer to *Tuning Parameters*.
4. *Use Memory Traces* - Use memory traces to store intermediate results. For more information, refer to [Managing Parameter Values and Traces](#).
5. *Update the Schematic* - Update your schematic with the new values and save your design. For more information, refer to [Updating Your Design](#).

Tunable and Non-Tunable Parameters

Most Advanced Design System parameters can be tuned; however, not all can. The following rules govern whether or not a parameter can be tuned. In order for a parameter to be tunable, *all* of the following statements must be true:

- It is a VAR or a simulator component parameter.
- It is contained in the hierarchy of the network.
- The design has been saved since you placed the component.

Note

If you attempt to set up a component parameter for tuning that has not been saved, the tuning application will not consider the component in the design hierarchy and an error message will be generated.

- It is optimizable, although specification of an optimization range is not required.
- It is real- or integer-valued (can include a scale factor with or without units); it cannot be expression-valued.
- It is a *specified* value, entered in the ADS design (parameters that default to simulator default values are not tunable).

Next are some examples of these rules.

- It is a VAR or a simulator component parameter.
Valid: VAR1.xx
Valid: C1.C
Not Valid: SMT_Pad.W (SMT_Pad is not a simulator component)
- It is optimizable, although specification of an optimization range is not required.
Valid: MLIN1.L = 50 mil
Valid: MLIN1.L = 50 mil opt{ 25 mil to 100 mil }
Not Valid: MLIN1.Subst = "Msub1" (not optimizable)
- It is real- or integer-valued (can include a scale factor with or without units); it cannot be expression-valued.
Valid: R1.R = 50
Valid: R1.R = 50 K
Valid: R1.R = 50 kOhm
Not Valid: R1.R = Rnom (expression-valued)
Not Valid: R1.R = 50 * Rscale (expression-valued)
Not Valid: R1.R = X (expression-valued)

Note

To use the *ADS Ptolemy Simulation* (ptolemy) (Signal Processing) Interactive Controls and Displays library components (such as TkPlot) with tune mode, you must dismiss the Interactive Controls and Displays component between each tune with its pop-up dialog box.

Choosing Tuning Parameters

There are different ways of selecting tune parameters depending on whether you are outside or inside of the tuning application.

- *Outside of tuning* - If you are outside of the tuning application, you can enable a parameter for tuning through the *Edit Component* dialog box or edit the value directly. For more information, refer to *Setting Up Parameters Outside of Tuning*.
- *Inside of tuning* - If you are within the tuning application, there are four different methods available for enabling tune parameters:
 - By clicking the parameter in the schematic.
 - By clicking the component in the schematic. A separate *Instance Tunable Parameters* dialog box is launched where you can choose from among all of the component's tunable parameters.
 - By clicking the **Include Opt Params** button inside the *Tune Parameters* dialog box. This automatically includes all of the parameters in the hierarchy that are enabled for optimization.
 - By clicking the **Enable/Disable** button inside the *Tune Parameters* dialog box. This launches a separate dialog box that displays all of the parameters in the hierarchy that are enabled or disabled for tuning.

For more information, refer to *Setting Up Parameters Inside of Tuning*.

Note

It is recommended that you only tune about four to five parameters during any one tuning session. While you can generally tune as many parameters as you like, selecting more than four or five may make it difficult to keep track of which changes are impacting your design.

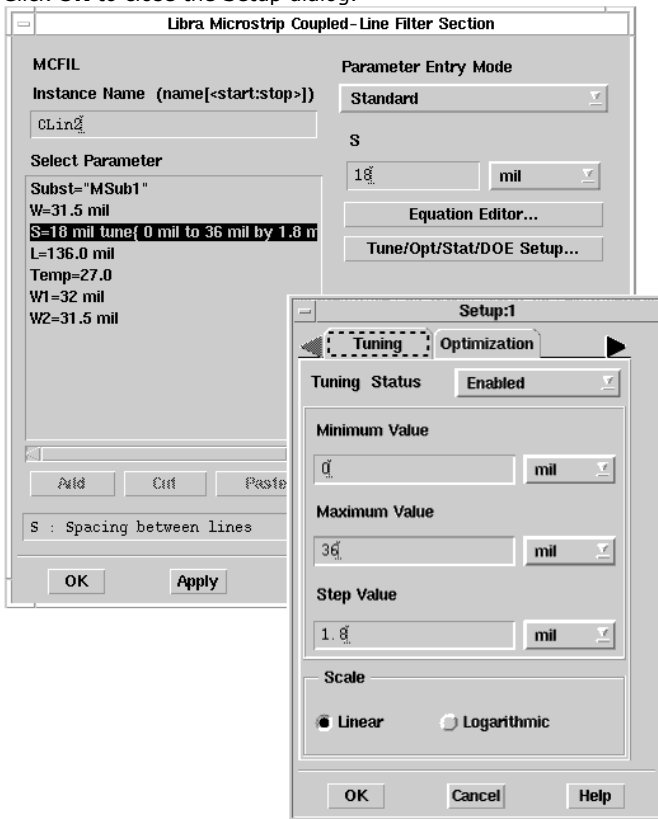
Setting Up Parameters Outside of Tuning

When choosing a parameter that you want to tune, you have the option of defining the tuning setup. To set up a tunable parameter:

1. Double-click the component that contains the parameter(s) you want to tune. The *Edit Component* dialog box appears.
2. In the *Select Parameter* field, click the parameter that you want to set up for tuning and then click the **Tune/Opt/Stat/DOE Setup** button. The *Setup* dialog box appears.

Note
Some components or component parameters are not tunable. For example, a component that has no nominal value assigned will not be tunable; the Tuning tab will be greyed out in the Setup dialog box. For more information, refer to *Tunable and Non-Tunable Parameters*.

3. Click the *Tuning* tab and then select *Enabled* from the *Tuning Status* pull-down menu item. The *Tuning Status* drop-down list provides three choices,
 - **Enabled** - When this option is selected, the parameter will be activated for tuning.
 - **Disabled** - The parameter is deactivated for tuning; however, the setup information is retained.
 - **Clear** - The parameter is deactivated for tuning and there is no setup information.
4. Setup the Minimum Value, Maximum Value, Step Value, and Scale (Linear or Logarithmic) settings for the enabled parameter. Note that you can change these values later during your tuning session as needed.
5. Click **OK** to close the Setup dialog.



Using the Tune Syntax

The tune syntax is added to tunable parameters in the schematic window when you select a parameter for tuning. The tuning syntax uses the form:

$$y = x \text{ tune}\{ \text{min to max} [\text{by step} | \text{logScale}] \}$$

where

- y* is the parameter name
- x* is the parameter value, in specified units
- min* is the minimum parameter tune value, in specified units
- max* is the maximum parameter tune value, in specified units
- by step** is a linear parameter step value, in specified units. This is an optional argument; however, if it is not selected, you must select **logScale**.
- logScale** is the logarithmic scale. This is an optional scale setting; however, if it is not selected, you must select a linear scale; that is, **by step**.

For example, a resistor with a nominal value of 50 ohms that you want to tune from 25 to 75 ohms in increments of 5 ohms would take the form:

$$R = 50 \text{ Ohms tune}\{25 \text{ Ohms to } 75 \text{ Ohms by } 5 \text{ Ohms}\}$$

The three modes used in the tune Setup dialog box include:

- **Enabled** - Tuning syntax is set up and the tuning option is currently enabled. For example,
`S = 18 mil tune{15 mil to 25 mil by 5 mil}`
or
`S = 18 mil tune{15 mil to 25 mil logScale}`
- **Disabled** - Tuning syntax is set up; however, the tuning option is currently disabled. For example,
`S = 18 mil notune{15 mil to 25 mil by 5 mil}`
or
`S = 18 mil notune{15 mil to 25 mil logScale}`
Note that the disabled tuning syntax uses *notune* as opposed to *tune*.
- **Clear** - No tuning syntax is included.
`S = 18 mil`

Note that parameter values can be edited directly on the schematic using the appropriate tuning syntax. Use the examples above as a guide to editing your tuning parameters.

When you launch tuning, parameters that use the *tune* syntax will automatically be included in your tuning session.

Note that the tune syntax is part of the parameter value itself, so when you save your ADS design, the tuning information will also be saved.

If a component you have selected does not include the **Tune/Opt/Stat/DOE Setup** button, you may need to manually include the tuning syntax for the parameter.

Abbreviating the Tune Syntax

Parameter values using the tune syntax can be set to appear abbreviated on the schematic. To change the behavior of how the tune syntax appears on the schematic,

1. From the schematic window, choose **Options > Preferences**. The Preferences for Schematic dialog box appears.
2. Select the *Component Text/Wire Label* tab. The *Format* section on the Component Text/Wire Label tab includes *Tune* format attributes which can be set to *Full*, *Short*, or *None* using the appropriate radio buttons.
 - **Full** is the syntax described in *Using the Tune Syntax*. For example, 50 Ohm
`tune{25 Ohm to 75 Ohm by 5 Ohm}`
 - **Short** is an abbreviated syntax: `{t}` for tune and `{-t}` for notune. For example,
50 Ohm `{t}`
 - **None** will only display the nominal values in the annotation. For example, 50 Ohm

Note
On-screen editing of the *Short* or *None* formatted component text will expand to the *Full* format. Also when using **Edit > Component > Edit Component Parameters** to edit the tune syntax of a component, the associated dialog box will always display component text in *Full* format. After editing using either of these methods, the component text on the schematic will continue to be in the format specified by the schematic preferences.

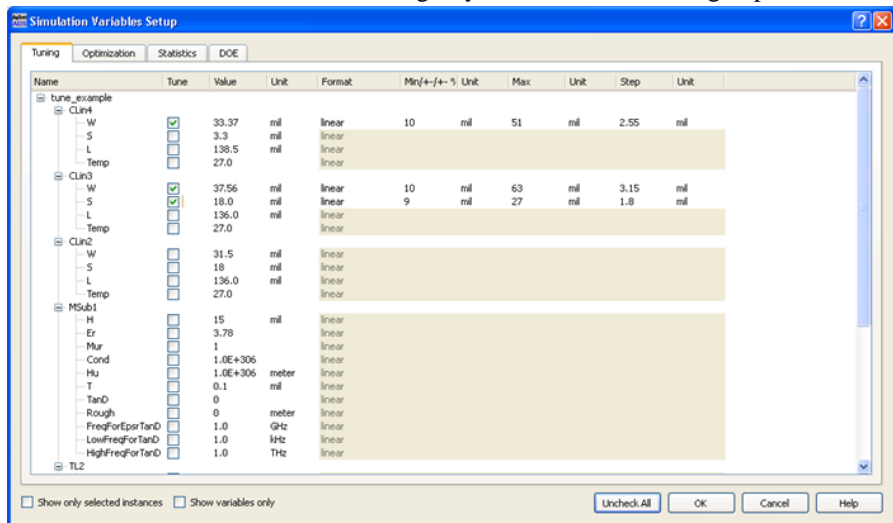
For more information, refer to *Setting Component Text/Wire Label Options (in Advance)* (custom).

Tuning Parameters Simultaneously for Multiple Components

The Simulation Variable dialog lists all the tunable parameters in a design while allowing the simultaneous adjustment of a number of parameters across multiple components on a single schematic. To view the dialog, in the Schematic window, click **Simulate > Simulation Variable Setup**.

To configure a tunable parameter:

1. Select the checkbox in the *Tune* column to enable tuning for a particular component.
2. In the Format drop-down list, select the desired format for tuning, either **linear** or **logarithmic**.
The default values for *Min*, *Max*, and *Step* are displayed as appropriate. To modify a *Min*, *Max*, and *Step* value, enter the desired value in the appropriate field. The default values for *Min*, *Max*, and *Step* are 50%, 150% and 10% of the nominal value of the parameter, respectively.
3. To disable tuning, deselect the checkbox in the *Tune* column.
4. Select the *Show only selected instances* checkbox to display only the selected components in the schematic.
5. Click **Uncheck All** to deselect all tunable parameters.
6. Click **Ok** to close the dialog.



Tuning Parameters

After fulfilling the prerequisites defined in the *Basic Tuning Procedure*, you are ready to launch a tuning session.

Setting Up Parameters Inside of Tuning

1. Click the *Tune Parameters* icon (tuning fork), or choose the **Simulate > Tuning** menu item. The *Tune Parameters* dialog box appears. The following table provides a brief description of the features.

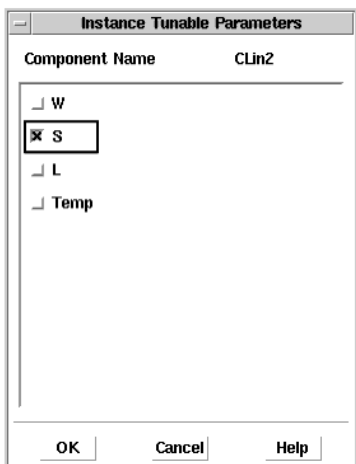
Tune Parameters Dialog Box

Section	Option	Description
Simulate	After Pressing Tune	Perform an analysis only after the Tune button is clicked. This option is designed for tuning after multiple changes, but can also be used for single changes. †
	After Each Change	Perform an analysis after each change. †
	While Slider Moves	Performs continual analyses while moving the slider. This option is similar to the "After Each Change" option, except that it is continuous. †
	Tune	Tune the design. This button is active only when the "After Pressing Tune" option is selected.
Parameters	Include Opt Params	Include parameters that are enabled for optimization. If the optimization-enabled parameter does not already have a tuning setup, the optimization setup will be used for tuning.
	Enable/Disable	Launches the Enable/Disable Parameters dialog box. This dialog box is used to enable disabled parameters and disable enabled parameters.
	Snap Slider to Step	For parameters tuned in a Linear scale, the slider moves in increments of the step when "Snap Slider to Step" is selected. Otherwise, the slider moves continuously. For parameters tuned in a Logarithmic scale, the slider moves continuously, regardless of the "Snap Slider to Step" option's setting. †
Traces and Values	Store	Stores the tuned parameter values in temporary storage and creates a memory trace for each trace in the Data Display. Note that when you close tuning, all of the stored traces and values are deleted.
	Recall	Restores the parameter values for a specified, previously-stored state. Note that if you have changed which parameters have been tuned since the state was originally stored, you may need to choose between the original values at the time the state was stored and the current values. At this point you will be asked to choose between Original or Current. †
	Trace Visibility	Lists all of the stored states and enables you to specify whether a memory trace is visible. †
	Reset Values	Resets the tuned parameters to their nominal values.
Tuned Parameters	Update Schematic	Updates the schematic with the tuned parameter values.
	Close	Closes the Tune Parameters dialog box. Note that all stored states and memory traces will be deleted.
	Help	Launches the online help.
	Value	Change the value of the parameter.
	Max	Enter the maximum value for the parameter's tuning range.
	Min	Enter the minimum value for the parameter's tuning range.
	Step	Enter a value that represents the step size. When the up/down arrow buttons are clicked, the value will increment/decrement by the step size. This value is also used to create the slider increments when the "Snap Slider to Step" option is selected.
	Scale	Select a Linear or Logarithmic slider scaling. †

† The default setting for this option is defined by the user preference settings. To set user preferences, choose Options > Preferences and select the Tuning tab.
† If there are no stored traces and values, this button will be deactivated.

1. Move your cursor over the schematic. Notice that the crosshairs appear in the window. This lets you select the tune parameters.
2. Click a parameter that you want to tune. The *Tune Parameters* dialog box is updated with a new slider for the parameter selected and the schematic is updated with the tune syntax.

The method above describes how to select individual parameters from the schematic. Alternatively, you can click any component in the schematic. A separate *Instance Tunable Parameters* dialog box is launched enabling you to choose from among all of the component's tunable parameters as shown in the dialog box below.



Note that you can modify each parameter's, Max, Min, Step, and Scale settings after initiating a tuning session. If the parameter is specified as an integer, the tuning application will constrain the parameter value to integer values. If you attempt to set the Max less than the parameter value, the value automatically changes to the Max. Similarly, if you attempt to set the Min greater than the parameter value, the value automatically changes to the Min. Step is ignored if you select a logarithmic scale for a parameter. Note that the step field is grayed out when the Scale is set to Log. The Min and the Value must be positive when using a logarithmic scale.

Tuning Hierarchical Networks

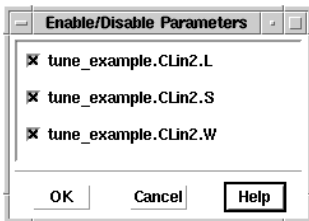
If your schematic design is hierarchical, that is, if it contains subnetworks, you can tune the components within those subnetworks without having to exit tuning.

While in the Tune Parameters mode, select **Push Into Hierarchy** from the View menu in the Schematic window or choose the *Push Into Hierarchy* icon. Click the subnetwork of interest. The Schematic window now displays the subnetwork design. At this point, you can proceed to tune parameters inside the subnetwork.

Note
Changes are made at the definition level, not the instance level. Therefore, if you update a subnetwork using tune mode, *all* instances of that subnetwork will be changed, not just the one you are pushed into.

Enabling and Disabling Parameters

Click the **Enable/Disable** button inside the *Tune Parameters* dialog box to launch the *Enable/Disable Parameters* dialog box. This dialog displays all of the parameters in the hierarchy that are enabled or disabled for tuning.



Deselecting a tunable parameter simply disables the parameter by changing the tuning syntax from *tune* to *notune*. For example, a component parameter that is set up for tuning but has been disabled will appear similar to the following,

```
S = 18 mil notune{15 mil to 25 mil by 5 mil}
```

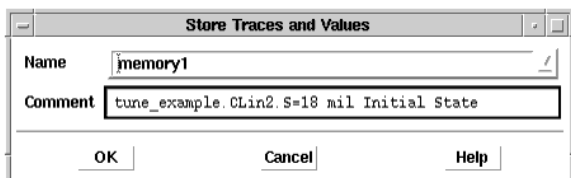
Note that the disabled tuning syntax uses *notune* as opposed to *tune*. The tuning set up is maintained.

Managing Parameter Values and Traces

Advanced Design System provides additional capability that enables you to store the tuned parameter values to memory, recall these tuned parameter values, and modify the visibility of the stored values' memory traces. Having your results stored also enables you to return to a stored state at any time during the tuning session.

Storing Values and Traces

1. Click the **Store** button to create your trace data and store parameter settings to memory. The *Store Traces and Values* dialog box appears.



A default *Name* and *Comment* will appear in the dialog box. The *Comment* defaults to the tuned parameter names and values. When using Legends in Data Display, both the *Name* and *Comment* appear in the legend for identification purposes. For more

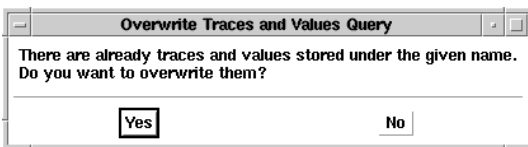
information on legends, refer to the *Data Display* (data).

		dB(S(1,1))	—
		dB(S(2,1))	—
memory1	S=18 mil	mem(memory1 dB(S(1,1)))	- - - - -
memory1	S=18 mil	mem(memory1 dB(S(2,1)))	- - - - -
memory2	S=15 mil	mem(memory2 dB(S(1,1)))	- - - - -
memory2	S=15 mil	mem(memory2 dB(S(2,1)))	- - - - -
memory3	S=25 mil	mem(memory3 dB(S(1,1)))	- - - - -
memory3	S=25 mil	mem(memory3 dB(S(2,1)))	- - - - -

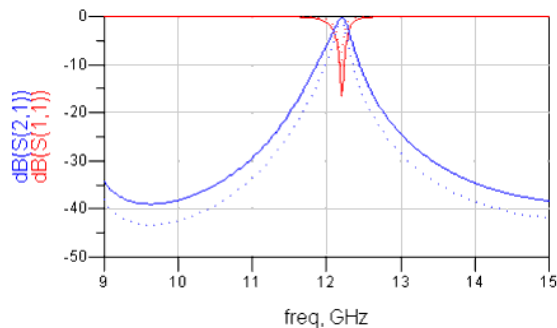
Example Plot Legend

Names are always required while Comments are optional. Comments should be kept concise in order to fit well within Data Display legends. This is important for general readability as well as documentation. Notice in the previous figure how a relatively short comment can quickly expand the width of the legend.

1. Enter a unique name and comment for your tuning state. For example, you might accept the default *memory1* and enter *good stability* as a comment. You can alternatively select the name of a previously stored state from the drop-down list in the *Name* field. This enables you to overwrite the previous state with the current values. Note that this will delete the previous stored state's memory trace and create a new one. A confirmation dialog will appear asking if you want to overwrite state.



2. Click **OK** to save your memory trace and return to your tuning session. Notice that the memory trace is now displayed with a dotted trace type (see the following figure) in your Data Display window along with your current data trace. As you continue to tune your parameters, you can compare your existing trace with the trace you have stored in memory.



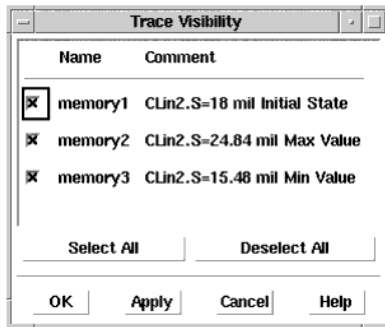
Example Memory Trace (dotted trace)

3. If after some additional tuning, you want to store another state, click the **Store** button again to store your new trace data and parameter settings to memory. The *Store Traces and Values* dialog box appears again.
4. Enter a new name and comment for your new tuning state and click **OK**. The new memory trace is also displayed with a dotted trace type in your Data Display window along with your original memory trace and your current data trace. As you continue to tune your parameters, you can compare your existing trace with the two traces you have stored in memory.

Setting Trace Visibility

The Trace Visibility button in the Tune Parameters dialog box enables you to display or hide one or more stored memory traces in the Data Display window. To display or hide a stored memory trace,

1. Click the **Trace Visibility** button. The *Trace Visibility* dialog box appears.



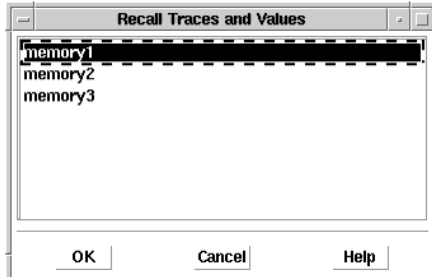
When you deselect a memory trace in the *Trace Visibility* dialog box, the visibility of the memory trace is turned off in the Data Display window after you click **Apply** or **OK**. The memory trace is still available. It is just not displayed.

2. Individually select the memory traces that you want to be visible in the Data Display window. If you want all traces visible, click **Select All**.
3. Deselect the memory traces that you want to hide in the Data Display window. If you do not want any of the traces visible, click **Deselect All**.
4. Click **Apply** to preview your selections in the Data Display.
5. If you are satisfied with your settings, click **OK**.

Recalling Values

You can recall parameter values that you have stored by clicking the *Recall* button in the Tune Parameters dialog box. To recall tuned parameter values that you have previously stored,

1. Click the **Recall** button in the *Tune Parameters* dialog box. The *Recall Traces and Values* dialog box appears containing a list of each of your stored states.



2. Click the stored state that you want to recall and then click **OK**. The parameters are recalled from memory and the Data Display window is updated. Note that if you have changed which parameters have been tuned since the state was originally stored, you may need to choose between the original values at the time the state was stored and the current values for the parameters that were not saved. At this point you will be asked to choose between Original or Current.

!optstat-2-1-10.gif!

Note
Memory traces are frozen. They are not reevaluated as tuning continues. If you change the equation that defines a memory trace, the memory trace will not be reevaluated using the modified expression. Also, if you delete a data trace in the Data Display, all of its associated memory traces will also be deleted. You can change memory trace display properties such as line type, color, thickness, etc. For more information, refer to *Editing Traces (data)*.

Updating Your Design

To update the design with the values of the tuned parameters:

1. Click the **Update Schematic** button to transfer your tuned parameter values to the schematic.
2. Click the **Close** button to close the Tune Parameters dialog box.
3. Save your schematic design. If you want to use your old and new improved designs for comparison later on, save the design with a new name.

Tuning Examples

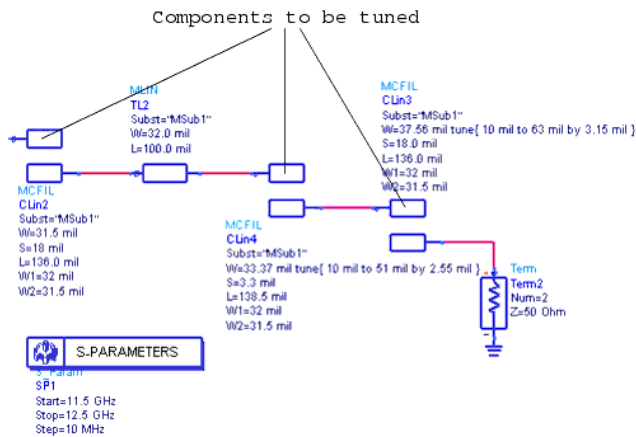
This section includes examples that are intended to help with your understanding of various tuning topics. The examples provided in this section include the following topics:

- [Analog/RF Systems Tuning Example](#)
- [Signal Processing Tuning Example](#)

Analog/RF Systems Tuning Example

This section shows a tuning example for Analog/RF Systems simulation. If you want to follow the similar steps for a Signal Processing example, skip to [Signal Processing Tuning Example](#).

The following figure shows part of the two-section microstrip filter with a 12 GHz bandpass example. Parameters of the three components shown here will be tuned in this example.



Two Section Microstrip Filter Example

The workspace containing this design can be copied from your examples directory.

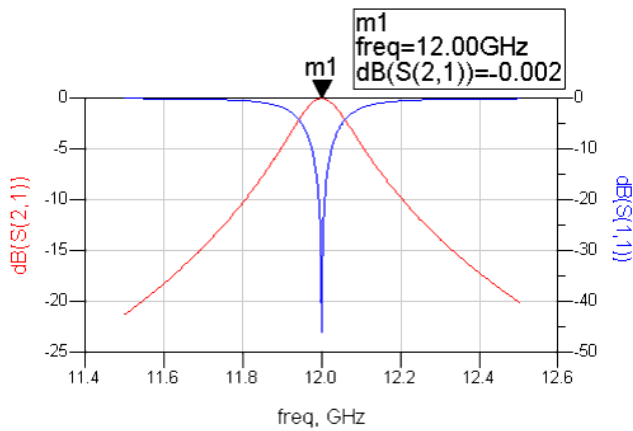
`$HPESOF_DIR/examples/Tutorial/Learn_Tune_wrk`

Using the example above, you will vary the effect of the filter by tuning the microstrip coupled-line filter components and observing plots of S_{11} and S_{21} while tuning. The MCFIL component instance CLin2's spacing parameter will be tuned after first tuning the width parameters for the CLin3 and CLin4 instances. Before you begin tuning, you will first need to copy the example workspace and simulate the initial design.

To practice tuning the example circuit shown in the previous figure, perform the following steps:

1. Copy the example workspace to a working directory where you have write permission.
2. Open *tune_example* and simulate the design.
3. Choose **Simulate** > **Simulate** or click the *Simulate* icon from the toolbar.
4. After the simulation has finished, a Data Display is automatically launched with a rectangular plot in the Data Display window. The plot shows the results for S_{21} and S_{11} in dB.
5. Place a marker on the S_{21} trace at 12 GHz to use as a reference while tuning by selecting the **Marker** > **New** menu item.


TWO SECTION MICROSTRIP FILTER



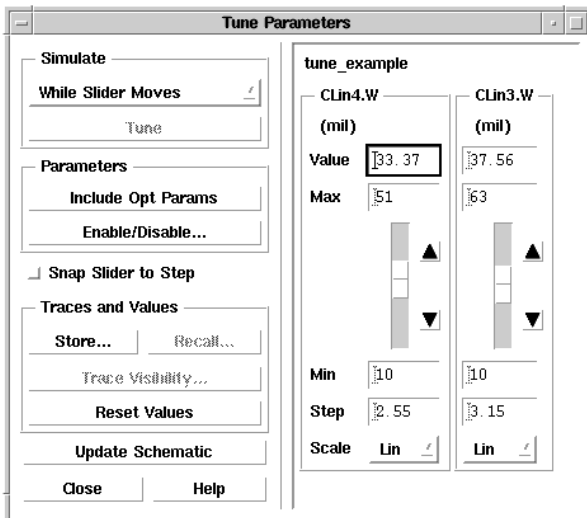
Initial Simulation Results

You can also add a legend to your plot using the **Insert > Plot Legend** menu item if desired. For more information on markers and legends, refer to the *Data Display* (data).

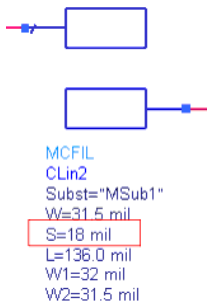
1. Choose **Simulate > Tuning** or choose the *Tune Parameters* icon (tuning fork) from

the toolbar. 

2. Wait for the initial analysis to complete. The *Tune Parameters* dialog box appears with CLin4.W and CLin3.W already enabled for tuning.

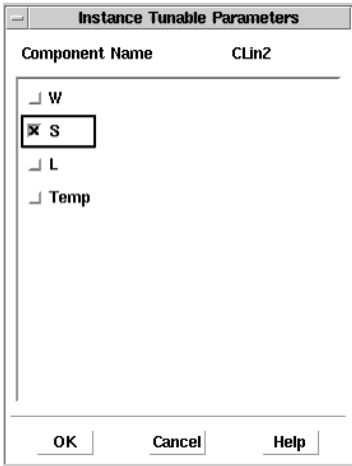


3. While observing the Data Display window, move the sliders up and down. Notice how the center frequency of the bandpass shifts up or down while tuning.
4. Now move your cursor over the schematic and notice that the crosshairs are active. Locate and click the component *MCFIL CLin2* instance on the schematic (see the following figure). anchor:1107858}

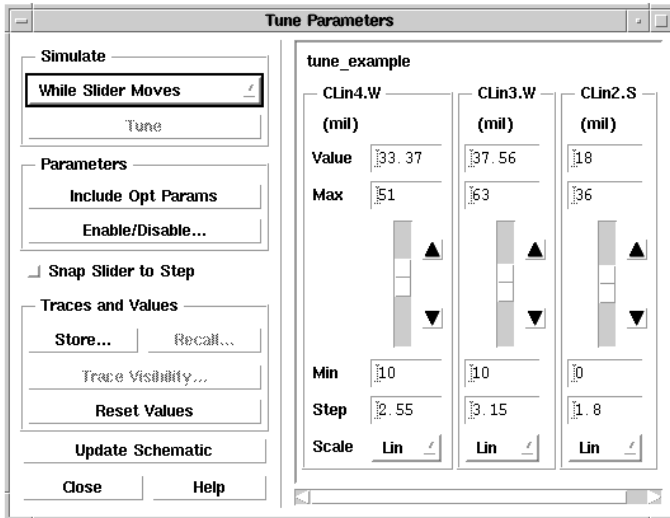


MCFIL CLin2 Instance Showing S (spacing) Parameter

After clicking the component, the *Instance Tunable Parameters* dialog box appears with a list of parameters as shown below.



1. For the *MCFIL* component *CLin2* instance (shown in the figure with *MCFIL* above), select the **S** (spacing) parameter in the *Instance Tunable Parameters* dialog box and click **OK**.
2. Notice that the *Tune Parameters* dialog box is updated with a new *CLin2.S* tuning slider as shown below.



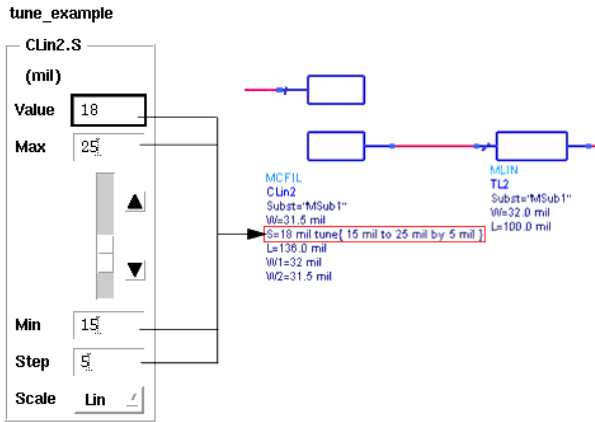
3. Select the tune analysis mode from the Simulate drop-down list in the Tune Parameters dialog box. This specifies when you want tuning to occur in the program. For this example, start with the *While Slider Moves* option. Try using each tuning analysis method (*After Pressing Tune*, *After Each Change*, *While Slider Moves*) to see which one works best for you. For more information on the Tune Parameters dialog box, refer to *Tuning Parameters*. The results of the tuning session are displayed in the same Data Display window that the initial simulation was displayed in.
4. You can change the tunable parameter by using any of the following methods:

Move the slider up or down

Click the up or down arrow

Manually enter a new value into the dialog box

Change the parameter values in the Tune Parameters dialog box for the **tune_example.CLin2.S** (spacing) parameter using a *Min* value of **15** mil (slightly below in the initial value of 18 mil), a *Max* value of **25** mil, and a *Step* value of **5** mil as shown below.



Notice that the *tune* syntax in the schematic window automatically updates with the new values you entered in the *Tune Parameters* dialog. For more information on the syntax, refer to *Using the Tune Syntax*.

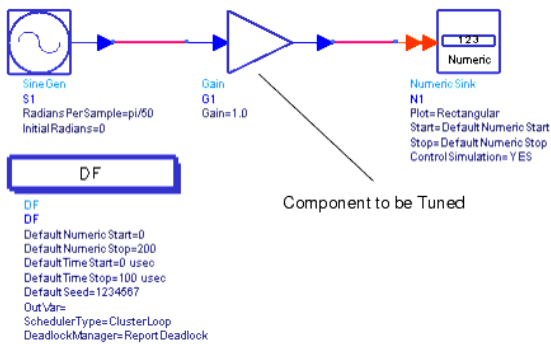
5. Move the slider up and down and observe the results in the Data Display each time you make a change.
6. You can click the **Update Schematic** button if you want the value you currently have entered in the Tune Parameters dialog box to be written to your schematic.
7. Click **Close** to exit the Tune Parameters dialog box.

For more practice tuning, try experimenting with the other tuning features using the other designs provided in the `$HPEESOF_DIR./examples/Tutorial/Learn_Tune_wrk` ADS example workspace.

Signal Processing Tuning Example

This section shows a tuning example for Signal Processing simulation. The steps are generally the same as in the [Analog/RF Systems Tuning Example](#), but this example uses a Signal Processing design.

The design in the following figure consists of a sine wave source, a gain component, and a numeric sink.



Sine Wave Source with Gain Component and Numeric Sink

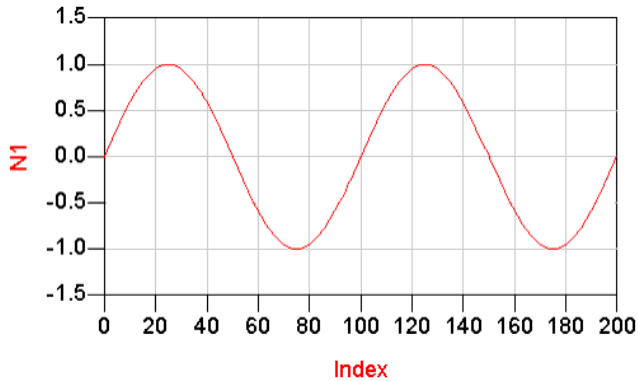
To build the example circuit shown in the previous figure,

1. Place the following components into a DSP schematic design window:
 - A *Sine Gen:Sine wave output* component from the Common Components palette
 - A *Gain:gain value* component from the Common Components palette
 - A *Numeric Sink:Numeric Data Sink* component from the Common Components palette
 - A *Data Flow Controller (DF)* from the Controllers palette.
2. Connect the sine wave generator, the gain value component, and the Numeric Sink using the Insert Wire icon.
3. Change the component settings so they match the settings in the previous figure.
4. After you have entered all of your changes, save your design.

Note If your complete design has not been saved, the tuning application will not consider the components in the design hierarchy and an error message will be reported when you attempt to set up a component parameter for tuning.

5. Simulate the design.

- Set up the Data Display window to display N1 (from the numeric sink). Note that you may need to change the Y Axis to accommodate the changes during your tuning session. Before any tuning occurs, the initial simulation results should appear as shown in the following figure.



Initial Results

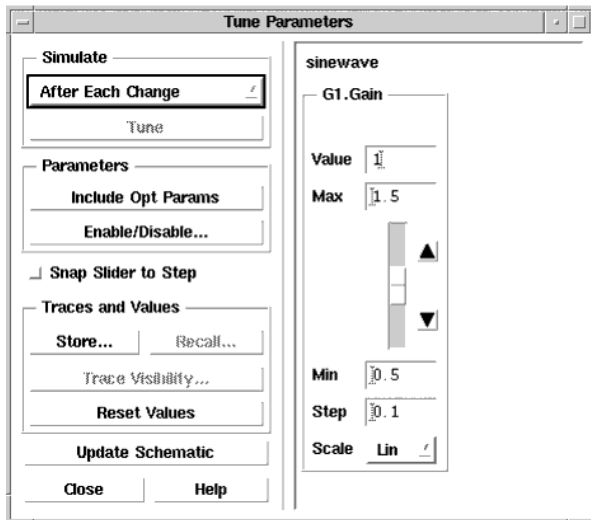
After the initial simulation, tune the circuit by following the procedure below:

- Choose **Simulate > Tuning** or choose the *Tune Parameters* icon from the toolbar.



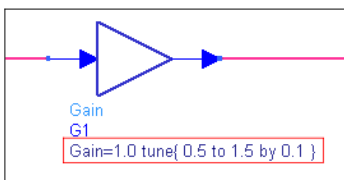
The *Tune Parameters* dialog box appears.

- Move your cursor over the schematic and notice that the crosshairs are active. Locate and click the *Gain* parameter in the Gain component on the schematic. Note that the *G1.Gain* parameter now appears with a slider in the Tune Parameters dialog box as shown in the following figure.



Tune Parameters with G1.Gain Parameter Setup

The Gain component parameter in the schematic window also displays the *tune* syntax as shown in the next figure. For more information on the syntax, refer to *Using the Tune Syntax*.



Gain Parameter with Tune Syntax

- In the Tune Parameters dialog box, select the tune analysis mode from the Simulate drop-down list. This tells the tuning application when you want tuning to occur. For this example, choose *After each change*. After you have completed this example, try going back and using the different tuning

analysis modes (*After Pressing Tune, After Each Change, While Slider Moves*) to see which one works best for you. For more information on the different tuning modes, refer to *Tuning Parameters*.

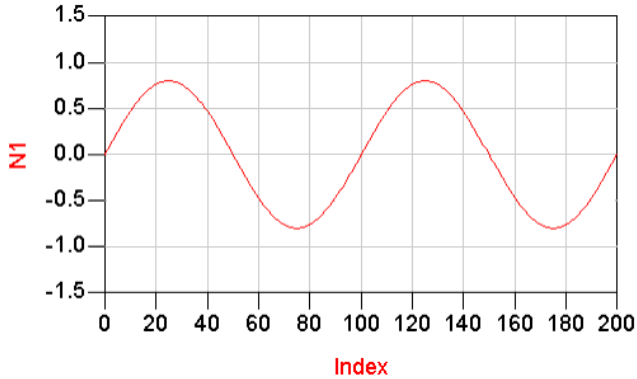
2. You can change the tunable parameter by using any of the following methods:

Move the slider

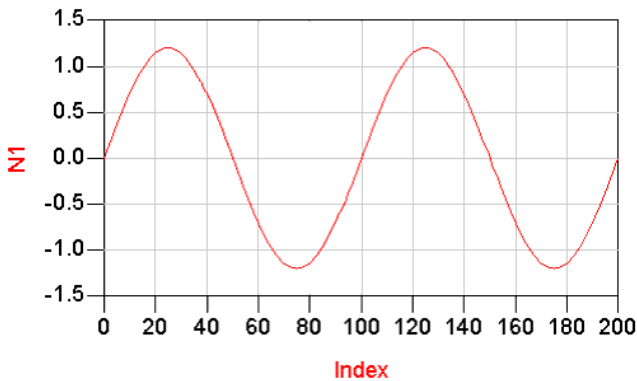
Click the up or down arrows

Enter a new value directly in the field

Change the parameter values in the Tune Parameters dialog box for the **G1.Gain** parameter using a *Min* value of **0.8**, a *Max* value of **1.2**, and a *Step* value of **0.1**. As you vary the slider, the results of your tuning session are displayed in the same Data Display window that the initial simulation was displayed in. The next two figures below show the simulation results for a minimum gain of 0.8 and a maximum gain of 1.2 respectively.



Results with G1.Gain Slider Set to 0.8



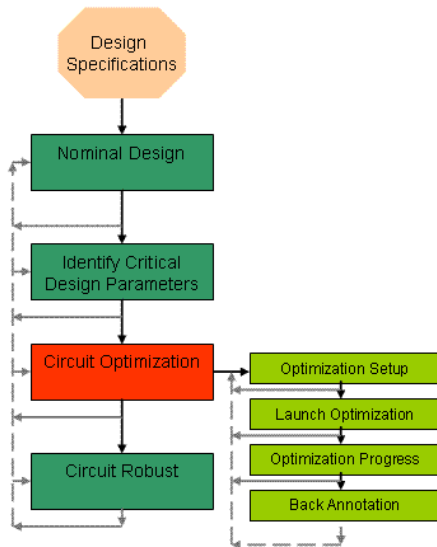
Results with G1.Gain Slider Set to 1.2

1. Once you are satisfied with your results, click the **Update Schematic** button in the Tune Parameters dialog box to update the parameter value in the schematic. If you do not want to change the values in your schematic, continue tuning or click the **Close** button to end the tuning session.

About Nominal Optimization

Introduction

Nominal Optimization, also known as Performance Optimization, is an important part of the engineering work. It helps the designers to find the optimal design to meet the design specifications.



Considering the design flow in ADS, the designer usually starts with the given design specifications. Then he creates the circuits and does the nominal simulation. He will compare the results with the design specifications on DDS. He then identifies the critical design parameters and starts the optimization. The final step is usually the circuit robust analysis.

Overall, optimization is the process of modifying a set of parameter values to satisfy predetermined performance goals. Optimizers compare computed and desired responses and modify design parameter values to bring the computed response closer to that desired. Nominal Optimization is available in the Advanced Design System simulators as follows:

- For Analog/RF systems simulation using any analysis type (such as AC, DC, S-Parameter, Harmonic Balance, Circuit Envelope, and Transient simulation types)
- For ADS Ptolemy signal processing simulation

Nominal optimization can be performed in conjunction with any frequency-domain or time-domain Analog/RF Systems simulation component as well as most Signal Processing components. For example:

- To optimize the response of a low-pass filter, you can perform an S-parameter simulation or an AC simulation to calculate the output amplitude of the filter over a frequency range, then change filter parameter values to refine filter response shape.
- To optimize the rise time of a pulse, you can perform a transient simulation to calculate the output voltage over a period of time, then change circuit parameter values to fine-tune the rise time of the pulse.
- You can optimize the gain of a carrier recovery loop to achieve a desired lock time and residual loop error.
- You can optimize a fixed-point bit-width parameter in a DSP design.

Note

Some ADS Ptolemy parameter types (Complex, Precision, Array, String, or Filename) require additional steps to complete optimization. These steps are described in *Optimizing Various Parameter Types* (ptolemy).

Examples of goals include characteristics of an output signal such as rise time, bandpass shape, or harmonic output. Minimum and/or maximum acceptable performance are used to define the limits of a goal. These limits can be a function of a fixed or a swept variable evaluated at the beginning of the optimization process.

The steps of nominal optimization include:

1. Running a simulation.
2. Comparing results with the goal.
3. Changing the circuit parameters to obtain results that are likely to be closer to the goal.
4. Running a simulation again with the new parameter values.

Features

ADS provides multiple optimizers for different design problems and different design purposes:

- *Random Optimizer* (optstat)
- *Gradient Optimizer* (optstat)
- *Random Minimax Optimizer* (optstat)
- *Gradient Minimax Optimizer* (optstat)
- *Quasi-Newton Optimizer* (optstat)
- *Least Pth Optimizer* (optstat)
- *Minimax Optimizer* (optstat)
- *Random Max Optimizer* (optstat)
- *Hybrid Optimizer* (optstat)
- *Discrete Optimizer* (optstat)
- *Genetic Optimizer* (optstat)
- *Simulated Annealing Optimizer* (optstat)

Each optimizer uses a different combination of error-function (EF) formulation and search methods to achieve the desired results. The detailed information for these optimizers is available on *Summary of Optimizers* (optstat).

Combined with these advanced optimizers, ADS also provides the following optimization methodologies:

- **Single Optimization** - Single Optimization is the basic optimization flow. The design only contains a single optimizer (or a single optimization controller), which is to find the best values of the defined design parameters to satisfy the desired performance goals.
- **Final Analysis** - Final analysis, specified in the optimization controller, is run automatically after an optimization and uses the optimal design parameters. It is useful when the analysis executed by the optimizer uses a different sweep grid than the one you want for your output.
- **Swept Optimization** - Swept optimization enables designers to optimize any circuit at any swept parameter value. For example, a circuit can be optimized at any temperature level with the temperature being the swept variable. Similarly, a digital programmable attenuator can be optimized at every level of attenuation with the attenuation voltage being the swept variable. Swept optimization is achieved by using a Parameter Sweep Controller referring to an Optimization Controller.
- **Sequential Optimization** - Sequential optimization is an extension of swept optimization. It will run different optimizers for the same design. The most common usage include: use random optimizer first to do large range search. Then changed to gradient optimizer for the fast convergence to a local minimum. Sequential optimization is achieved by using the Parameter Sweep component with more than one Optimization component.
- **Programmable Optimization** - Programmable optimization is an extension of sequential optimization. The purpose of the programmable optimization is to enable the flexible optimization order. For example, firstly optimize the design for gain and then optimize it for noise figure. Usually, the optimizer in the different step will use different goals and different optimization variables. Programmable optimization contains a Parameter Sweep Controller with one or more Optimization Controller. The Parameter Sweep Controller will enable you to program your optimization steps. The optimization controller will set the job parameters.

Note

Sensitivity is not part of Optimization. Sensitivity Controller is available in **Opt/Stat/DOE** palette. For more information on sensitivity analysis, refer to *Sensitivity Analysis* (sensana).

Minimum Requirements

An optimization process requires three parts:

- **Goals**- Goals are the specifications for the circuit performances to be met.
- **Optimization Variables/Parameters**- The optimization variables/parameters are the critical design parameters, to which the circuit performances are sensitive.
- **Optimization methodology and optimizers**- Optimization methodology and optimizers define the approach to adjust the optimization variables/parameters to meet the goals.


So, the minimum requirements for the optimization setup includes:

- At least one optimization goal component (*Goal*) placed in the Schematic window.
- At least one component variable/parameter in your design identified as an

optimization variable. You specify details in the Component Parameter dialog box by choosing the **Tune/Opt/Stat/DOE Setup** button, or in the central Simulation Variable Setup window by choosing **Simulate > Simulation Variables Setup...** menu pick from schematic window.

- One nominal optimization component (*Optim*) placed in the Schematic window. The *Goal* and *Optim* components are accessed as follows:
 - For Analog/RF Systems simulation, from the *Optim/Stat/DOE* palette or component library.
 - For Signal Processing simulation, from the *Controllers* palette or component library.
- One simulation control component (a Data Flow controller for ADS Ptolemy simulating or an AC, DC, S-Parameter, Harmonic Balance, LSSP, XDB, Circuit Envelope, Transient, ChannelSim and Budget simulation component for Analog/RF Systems simulation).

Initiation

To initiate optimization after specifying the Goals, the optimization variables/parameters and the optimization component, either choose **Simulate > Optimize...** from the menu toolbar or click **Optimize** button  from the icon toolbar. Depending on the setup of the optimization component, the optimization will execute either in **Cockpit** mode, or **Non-Cockpit** mode. The status window will also show the optimization status, including the initial and current error function (EF) values, and the current trial/iteration.

To cancel the optimization process, close **Cockpit** if it is in Cockpit mode, or choose **Simulate > Stop and Release Simulator...** to interrupt the process.

Single Optimization

Introduction

Single Optimization is the basis of all ADS optimization features. The design contains a single optimizer (or a single optimization component) to find the best value of the defined design parameters to satisfy the desired performance goals.

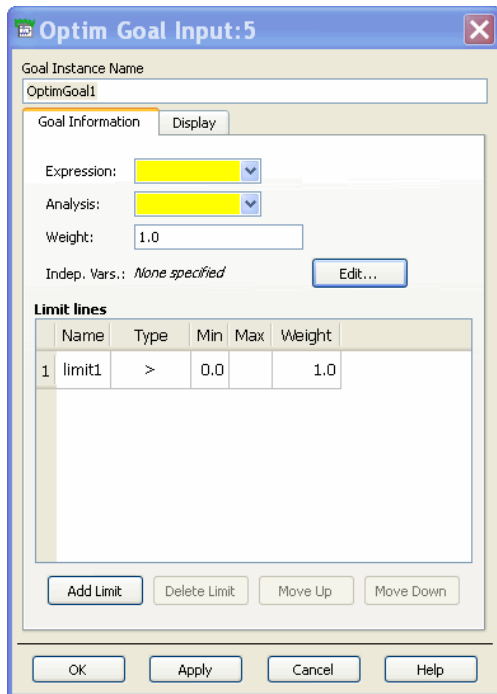
Setup

Setting Optimization Goals

Optimization goals are specified by placing a Goal component and double-clicking it to display the Goals for Nominal Type Optimization dialog box. The Goal component can be found in:

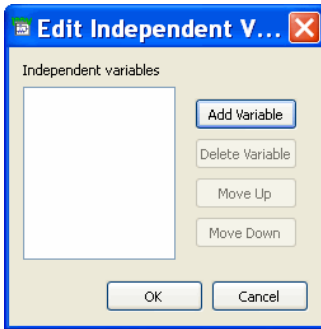
- The *Optim/Stat/DOE* palette or library, for Analog/RF Systems simulation.
- The *Controller* palette or library, for Signal Processing simulation.

You can specify and place more than one Goal if needed. The goals to be used are referenced by the Nominal Optimization component, as described in the later section, [Selecting an Optimizer and Goals](#). By default, all goals placed apply to all Nominal Optimization components in a design.



To set appropriate goal specifications in this dialog box:

1. Goal Instance Name: The name of the goal component.
2. Expression: A valid AEL expression that operates on the simulation results. All relevant measurement equations in your design are available in the list. Or you can type in an equation.
3. Analysis: The instance name of the simulation control component to which **Expression** will be applied. All relevant simulation control components in the design are available in the list.
4. Weight: This weight applies to all of the limit lines. The final weight factor for each limit line is the product of this weight factor and the limit line weight factor.
5. Indep. Vars: The independent variable(s) for **Expression**. This is used to restrict the limit line to a subset of the data produced by the simulation. For example, when the SP analysis is from 1 GHz to 2 GHz, but the pass-band limit line is only from 1.5 GHz to 1.8 GHz. To modify the independent variables, click **Edit** and the *Edit Independent Variables* dialog appears:

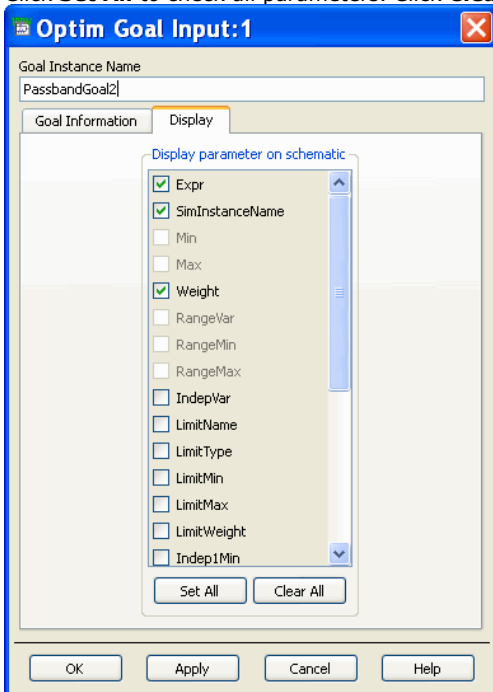


6. Limit Lines table: Contains one or more limit lines, which define the acceptable responses. For example, if the design specifications have one passband and two stopbands for dB(S21), then there will be three limit lines in the limit line table. Each limit line contains:
- Limit name: The name of the limit line.
 - Type: the relationship between the limit line Min/Max values and the responses.
 - >: specifies "Expression > Min"
 - <: specifies "Expression < Max"
 - =: specifies "Expression = Min = Max"
 - Inside: specifies "Min < Expression < Max"
 - Outside: specifies "Expression < Min OR Max < Expression"
 - Min: The minimum limit value. It is related with Type. See above.
 - Max: The maximum limit value. It is related with Type. See above.
 - Weight: The weight factor for the limit line. The actual weight factor used in the error function calculation for the limit line is the product of the **Goal Weight** factor and the **Limit Line** weight factor.
 - Indep.Var Min, Indep. Var Max: These appear only when you define the Indep. Vars. Then in the limit line table, you can specify the limit line with the indep. var range. For example, you can specify the passband limit line for the frequency range from 1.5 GHz to 1.8 GHz.
- In the limit line table, you can use "Add Limit", "Delete Limit", "Move Up", and "Move Down" buttons to manipulate the limit lines.

Displaying Goal Parameters on the schematic

The *Display* tab of the Optimization Goal dialog box is used to select the parameters that are displayed on the schematic related to goal component. Generally, it is not necessary to modify the visible parameters. Check the appropriate boxes in the *Display* tab dialog box as shown below:

- To select or deselect any parameter, click in the appropriate checkbox.
- Click **Set All** to check all parameters. Click **Clear All** to uncheck all parameters.



- After making appropriate selections, click **Apply** to continue entering data in the other tabs of this dialog box. If you are finished entering data in this dialog box, click **OK**.

Parameter	Description	Use Model
Expr	A valid AEL expression that operates on the simulation results, such as mag(S11), or the name of a MeasEqn. For more information on AEL expressions, refer to <i>AEL (ael)</i> or <i>Measurement Expressions (expmeas)</i> .	All associated expressions are displayed in drop down list. Select the one you want to optimize, or you can type it using the combo box.
SimInstanceName	Enter the instance name for the simulation control component that you placed in your design, which will generate the data used by the Expr field.	All associated analysis control components are displayed in the drop down list. Select the analysis component (simulation controller), such as S-parameter, that you want to optimize, or you can type it directly in the combo box.
Weight	Enter a weighting valued to be used in error function calculation. Default is 1. For more information on using the weighting factor to form the error function, refer to <i>Weighting Factors (optstat)</i> .	The weight factor will be applied to all of the limit lines within the goal component. If you want put more efforts on one goal, you can just increase the weight factor. If you want to disable this goal component, you can simply set the value as zero.
Indep. Vars	Independent variable name.	<i>Edit Independent Variables...</i> dialog will pop up once you click Edit button. You can "Add", "Delete", "Move Up" and "Move Down" for the independent variables in this dialog. The maximum number of independent variables are limited to 6 now.
LimitName	Limit line name.	A default name is always given, which can be changed by selecting the field and enter new name.
LimitType	Choose a limit line type.	There are 5 different limit line type. They are used to define the relationship between the expression and the LimitMin and/or LimitMax.
LimitMin	Enter a number for a minimum acceptable response value.	Entry Mode will be dependent on the Limit Line Type.
LimitMax	Enter a number for a maximum acceptable response value.	Same as above.
Indep1Min	Minimum limit of range for first independent variable of Expression.	Same as above.
Indep1Max	Maximum limit of range for first independent variable of Expression.	Same as above.
Indep2Min	Minimum limit of range for second independent variable of Expression.	Same as above.
Indep2Max	Maximum limit of range for second independent variable of Expression.	Same as above.
Indep3Min	Minimum limit of range for third independent variable of Expression.	Same as above.
Indep3Max	Maximum limit of range for third independent variable of Expression.	Same as above.
Indep4Min	Minimum limit of range for fourth independent variable of Expression.	Same as above.
Indep4Max	Maximum limit of range for fourth independent variable of Expression.	Same as above.
Indep5Min	Minimum limit of range for fifth independent variable of Expression.	Same as above.
Indep5Max	Maximum limit of range for fifth independent variable of Expression.	Same as above.
Indep6Min	Minimum limit of range for sixth independent variable of Expression.	Same as above.
Indep6Max	Maximum limit of range for sixth independent variable of Expression.	Same as above.

Note
 Min, Max, RangeVar, RangeMin and RangeMax variables are obsolete since ADS2009 Update 1. These variables are now mapped to:

- Min, Max - LimitMin, LimitMax
- RangeVar - IndepVar
- RangeMin, RangeMax - Indep{i}Min, Indep{i}Max. For e.g., Indep1Min, Indep1Max, Indep2Min, Indep2Max..... depending upon the number of IndepVars the goal has.

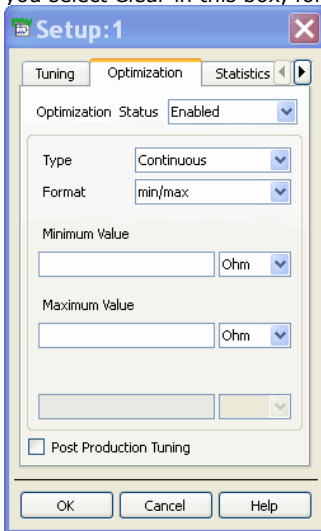
Setting Optimization Variables/Parameters

ADS provides two different flexible ways to specify the type and format for the parameters range over which optimization is to take place. You can either use the discrete model (component by component basis), or the central **Simulation Variable Setup...** model.

Setting Optimization Variables/Parameters using Discrete Model

The procedure for specifying component parameters for optimization is as follows:

1. Select and place an appropriate component from one of the component palettes or component libraries. For example, place a parallel resistor-inductor-capacitor (PRLC) from the *Lumped Components* palette in Analog/RF Systems or a *Gain component* in Signal Processing.
2. Double-click the component in the Schematic window to edit its parameters.
3. From the component dialog box, highlight the parameter that you want to optimize in the Select Parameters box (for example R for parallel resistance), then choose the **Tune/Opt/Stat/DOE Setup** button, which will appear for optimizable or statistical parameters (and when the default *Standard* Parameter Entry Mode is selected). The *Setup* dialog box appears, with the **Optimization** tab active. Note that the Tuning, Statistics, and DOE tabs are not needed for Nominal Optimization. These additional setup tabs are described in:
 - **Tune** - See *Tuning in Advanced Design System* (optstat)
 - **Stat** - See *Using Statistical Design* (optstat)
 - **DOE** - See *Using Design of Experiments (DOE)* (optstat)
4. From the Optimization Status drop-down list, select **Enabled** so you can edit the appropriate fields. *Enabled* causes the parameter to be optimized when the simulation is run. *Disabled* temporarily deactivates this parameter from being optimized, and *Clear* removes the values you previously applied to the design after you select *Clear* in this box, followed by *Apply* in the component dialog box.



5. From the Type drop-down list, select an appropriate optimization Value Type (Continuous or Discrete). For a description of Discrete optimization refer to *Discrete Optimization Example* (optstat). For descriptions of Value Types, refer to the section *Value Types for Nominal Optimization* (optstat).
6. From the Format drop-down list, select an appropriate optimization format (*min/max*, *+/- Delta %*, *+/- Delta*, or *Unconstrained*). Generally, you should pick as narrow a range as you believe will work. A large range or *Unconstrained* could use more simulation time. For descriptions of the available formats, refer to the section *Value Types for Nominal Optimization* (optstat) in *Available Value Types* (optstat).
7. If you selected a *min/max* format, you can optionally enter values for nominal, minimum, and maximum in the appropriate boxes, and select an appropriate unit assignment for each from the drop-down list next to the boxes. If you selected an *Unconstrained* format, only a nominal value and associated unit need to be specified. If you selected either of the *Delta* formats, the companion limit values must be specified.
8. From the Nominal Value field and the Units drop-down list, the value and units in your design for this component are displayed. You can change these to set your starting point for your optimization if you wish.

Note
The *Post Production Tuning* checkbox is used in Statistical Design, not in Nominal Optimization, and is described in *Using Statistical Design* (optstat).

Setting Optimization Variables/Parameters Using Central Model

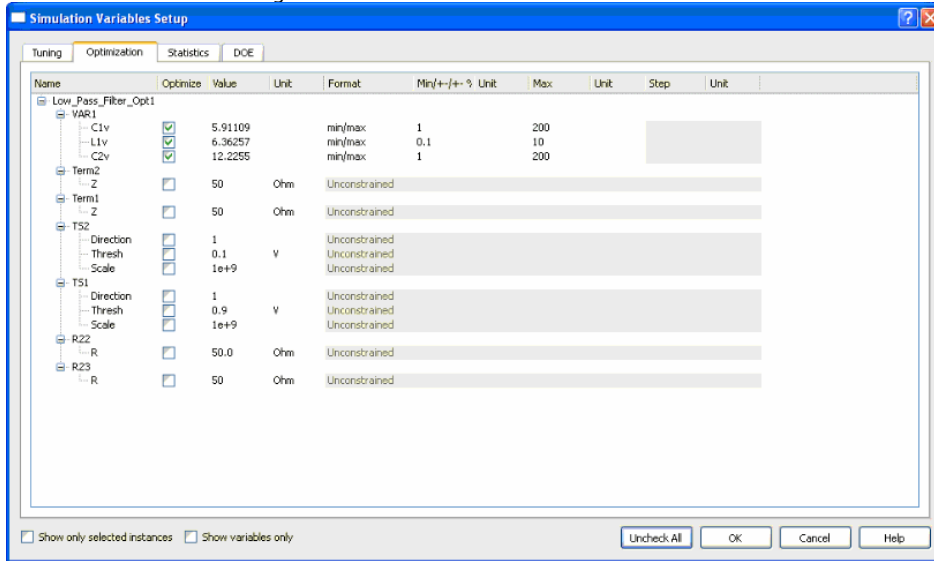
Using the *Simulation Variable* dialog, you can enable or disable the optimization status of the parameter and specify the type and format for the parameter range over which optimization is to take place.

To specify components for optimization:

1. In the Schematic window, click **Simulate > Simulation Variable Setup** to open the

Simulation Variable dialog.

2. To enable optimization, select the checkbox in *Optimize* column next to the desired component.
3. Select tuning format, **unconstrained**, **min/max**, **+/- delta**, **+/- delta %**, or **min/max/step** from the *Format* drop-down list.
The default values for *Min*, *Max*, and *Step* are displayed as appropriate. To modify a *Min*, *Max*, and *Step* value, enter the desired value in the appropriate field. The default values for *Min*, *Max*, and *Step* are 50%, 150% and 10% of the nominal value of the parameter, respectively.
4. To disable optimization, deselect the checkbox in the *Optimize* column.
5. Check the "Show only selected instances" checkbox to display only the components selected in the schematic.
6. Click **Uncheck All** to deselect all optimizable parameters.
7. Click **Ok** to close the dialog.



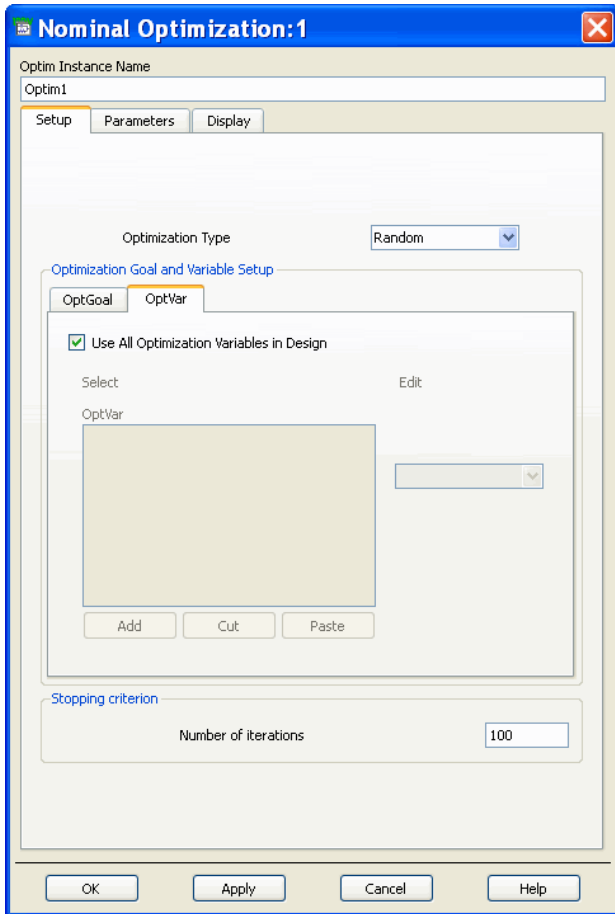
Setting Optimization Controller

To set job parameters, you need to specify data in the Nominal Optimization dialog box, as follows:

1. Place an *Optim* component in the appropriate Schematic window. It is found as follows:
 - For Analog/RF Systems simulation, from the *Optim/Stat/DOE* palette or library
 - For Signal Processing simulation, from the *Controllers* palette or library
2. Double-click the component to bring up the Nominal Optimization dialog box, which has three tabs. It is displayed with the *Setup* tab active.
3. Make specifications in each tab of the dialog box, as described below.

Selecting an Optimizer and Goals

Follow the steps below to set up an optimization in the *Setup* tab of the Nominal Optimization dialog box:

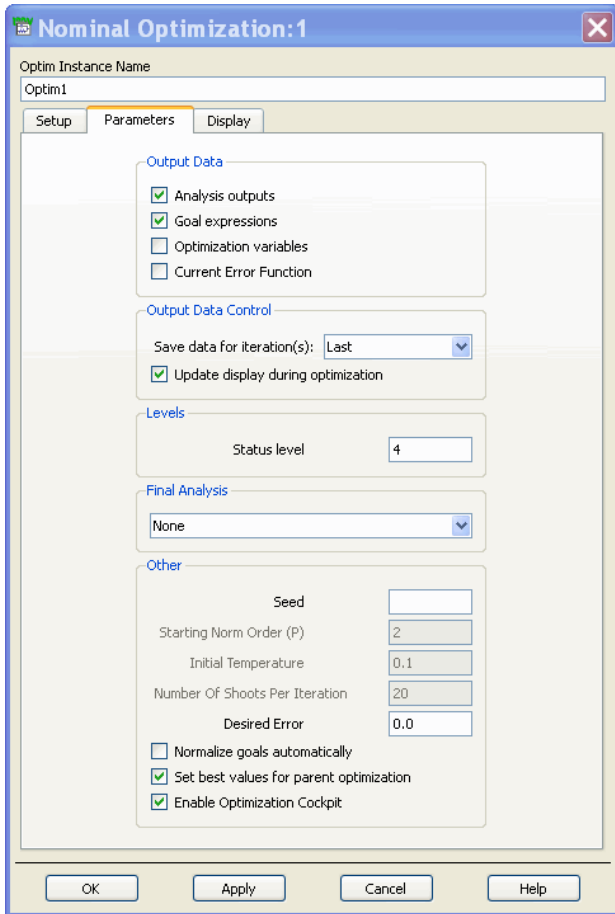


1. In the Optimization type field, select an appropriate optimizer from the drop-down list, such as Random (the default), Gradient, Least Pth, Minmax, Genetic, etc. For details on selecting an appropriate optimizer, refer to *Summary of Optimizers* (optstat).
2. Under *Stopping criterion*, specify the number of desired maximum trial/iteration to use during the optimization process. For the random optimizers (Random, Random Minimax, Random Max, and Genetic) this value represents the number of trials to attempt. Values in the range from 25-100 are recommended initially. For the iterative optimizers (Gradient, Gradient Minimax, Quasi-Newton, Least Pth, and Minimax) this value represents the number of iterations (improvements in the error function) to attempt. Less than 10 iterations are recommended initially. The default for all optimizers is 25 trials/iterations. The Discrete Optimizer will calculate the number of trials itself according to the variable setup. So this field is not applicable for the discrete optimizer.
3. In the Optimization Goal and Variable Setup box, first select the *OptGoal* tab and check the *Use All goals in Design* (default) checkbox. This is the best approach for most designs, and all goal components placed in a design will be implicitly associated with the optimization controller. To associate a *subset* of all goals with a given optimization controller, deselect the *Use All goals in Design* checkbox. Select a goal from the *Edit* drop-down list, which will include all Goal components that are currently placed in the design, as described in the preceding section, [Setting Optimization Goals](#). Click *Add* to place it in the *OptGoal* box, and repeat this step as necessary. Click the *Cut* or *Paste* buttons, if necessary to make any changes in the *OptGoal* box.
4. In the Optimization Goal and Variable Setup box, next select the *OptVar* tab and check the default *Use All Optimization Variables in Design* checkbox, unless you want to use only some optimization variables. You may have many optimization variables, specified by placing one or more VAR (variables and equations) components, in your design and want to associate only one or two of the variables, for example, with a given optimization controller. To associate a *subset* of all optimization variables with a given optimization controller, deselect the *Use All Optimization Variables in Design* checkbox. Then enter the name of an optimization variable you want to use in the *Edit* field and click the *Add* button. It is added to the *OptVar* box. Repeat this step as necessary. Click the *Cut* or *Paste* buttons, if necessary to make any changes in the *OptVar* box.
5. Click **Apply** button to retain the specifications that you have made while you enter data into the *Parameters* tab, as described below.

Section	Parameter	Description
Optimization Type	For more information, refer to <i>Summary of Optimizers</i> .	Options include Gradient, Random Minimax, Gradient Minimax, Quasi-Newton, Least Pth, Minimax, Random Max, Hybrid, Discrete, Genetic and Simulated Annealing.
OptGoal (Optimization Goal Setup) Tab	Use All goals in Design	When selected, all of the active goal components placed in a design will be used for the current optimization controller. The default is "selected". When de-selected, the OptGoal list box becomes active.
	OptGoal list box	Enables you to perform an optimization for a subset of the goal components placed in your design. Goal components can be added to the OptGoal list box using the Edit drop-down list and the Add or Paste button. Note that you can alternatively de-activate the unwanted Goal components in the design one by one. This method works if there is only one optimization controller in your design.
	Edit drop-down list	Includes all Goal components that are currently active in the design.
	Add	Adds a selected goal component from the Edit drop-down list.
	Cut	Cuts a selected goal component from the OptGoal list box.
	Paste	Pastes a selected goal component into the OptGoal list box.
OptVar (Optimization Variable Setup) Tab	Use All Optimization Variables in Design	When selected, all of the design parameters defined in your design will be used for the current optimization controller. The default is "selected". When de-selected, the OptVar list box becomes active.
	OptVar list box	Enables you to perform an optimization for a subset of the design parameters defined in your design. Design variables can be added to the OptVar list box using the Edit drop-down list and the Add or Paste button. Note that you can alternatively disable the unwanted design parameters one by one in the design. This method works if there is only one optimization controller in your design.
	Edit drop-down list	Includes all optimization design variables that are currently active in the design.
	Add	Adds a selected optimization design variable from the Edit drop-down list.
	Cut	Cuts a selected optimization design variable from the OptVar list box.
	Paste	Pastes a selected optimization design variable into the OptVar list box.
Stopping criterion	Maximum Number of trials/iterations	Specifies the number of desired trials/iterations to use during the optimization process. For the random optimizers (Random, Random Minimax, Random Max, Genetic, and Simulated Annealing) this value represents the number of trials to attempt. Values in the range from 25-100 are recommended initially. For the iterative optimizers (Gradient, Gradient Minimax, Quasi-Newton, Least Pth, and Minimax) this value represents the number of iterations (improvements in the error function) to attempt. Less than 10 iterations are recommended initially. The discrete optimizer won't use the input maximum number of iterations. The default for all optimizers is 25 trials/iterations.

Setting Parameter Information

Follow the steps below to set parameter information (type of data to save, etc) in the *Parameters* tab of the Nominal Optimization dialog box:



- In the Output Data field, specify which data you want to retain in your dataset following optimization. Check the following choices that apply.
 - Analysis outputs* send all analysis results (including measurement equations) to the dataset. This can create a substantial amount of data, especially if you choose to save all iterations.
 - Goals expressions* (default) sends the result of each active Goal's *Expr* field to the dataset.
 - Optimization variables* sends the values of all active optimization variables to the dataset for each improvement found during the optimization.
 - Current Error Function* sends the value of the current error-function (EF) formulation used to the dataset. For more information, refer to *Error-Function (EF) Formulation* (optstat).
- In the Output Data Control field, specify whether you want to:
 - Save data for iterations*. Choices are:
 Last - Only the last iteration is saved to the dataset.
 Nominal & last (default) - Only the nominal and last (best) iterations are saved to the dataset.
 All - Data for all iterations is saved. This can create a substantial amount of data and utilizes lot of memory.
 - Update display during optimization* (default). This updates the dataset on each optimization iteration so you can see the results in the Data Display window as they occur (instead of waiting till the end where all the traces are displayed at once). For faster results, turn this feature OFF.
- In the *Levels* field, enter a number for the desired annotation level. Levels are 0-4, with increasing information displayed in the Status window. (Default value is 4.)
- In the *Final Analysis* field, specify whether you want to employ an analysis run after your optimization is complete. This analysis can be of any analysis controller component that does not introduce circularity in analysis execution. The drop-down list allows you to select *None* (for no Final Analysis, the default) or any analysis controller component currently in your design (such as an S-Parameter or Data Flow controller).

Final analysis is useful when the analysis executed by the optimizer uses a different sweep grid than the one you want for your output. For example, if a coarse grid is required for optimization, but a finer grid, or a different range, is desired for output, then the analysis setup to generate this finer grid may be run after the optimization is completed, using the Final Analysis feature.

Multiple analysis are run by grouping them together in a parameter sweeper (without specifying a sweep variable), and choosing this sweeper in the Final Analysis drop-down list box. Note that for swept or nested optimization, the Final Analysis

parameter should not refer to any controller that is already executing. Data output for Final Analysis follows the information (flags) set in the companion *Optim* controller component.

5. In the *Other* field, specify a *Seed* value, *Order of optimization norm*, and *Desired Error* for use during optimization.
 - *Seed* is a value for the random number generator used to initiate an optimization. If *Seed* is not specified, the simulator chooses its own seed, which will be different each time an optimization type requiring a seed is used (Random, Random Max, Random Min, Genetic, and Discrete).
 - Possible values for *Order of optimization norm* are 2, 4, 8, or 16. (2 is the default.) For more information, refer to the section, *Error-Function (EF) Formulation* (optstat).
 - *Initial Temperature* applies only to the Simulated Annealing optimizer. The initial temperature and the number of shoots per iteration determine the annealing schedule mechanism for the system. The default value for the initial temperature is 0.1. The recommended range for it is 0.001 - 1000. For more information, see *Summary of Optimizers* (optstat).
 - *Number of Shoots Per Iteration* applies only to the Simulated Annealing optimizer. It sets the maximum number of iterations the downhill simplex method uses for each temperature level, or for one iteration of simulated annealing method. It decides the thermodynamics state of the system in one iteration. The default value for it is 20. The recommended range is 10 - 5000. For more information, see *Summary of Optimizers* (optstat).
 - The *Desired Error* field represents the value of the error function that is acceptable to terminate the analysis. If you want all goals to be met, accept the default of 0.
 - It is strongly suggested that you select the *Normalize goals automatically* checkbox when the optimization goals use a default *weighting factor*. The effect of this option is to produce an internal *weighting factor* for each goal. This prevents any one goal from dominating the error function. For more information, refer to *Error-Function (EF) Formulation* (optstat).
 - If you want to *Set the best values for the parent optimization*, leave the checkbox in its default setting (checked). With this box checked, the optimal values are saved internally so that they can be either user-updated or utilized by a subsequent analysis. To disable this setting, click the checkbox.
 - By default, *Enable Optimization Cockpit* is selected and the cockpit will pop up when you click **Optimize** button. You can use the optimization cockpit to monitor and control the optimization process. If you do not want to run optimization in the cockpit mode, you need to disable this setting by clicking the checkbox.
6. Choose **Apply** to retain the specifications that you have made while you enter data into the *Display* tab, as described in the next section.

Nominal Optimization Parameters Tab

Section	Parameter	Description
Output Data This field is used to specify which data you want to retain in your dataset following an optimization. Check all choices that apply.	Analysis outputs	When activated, all of the outputs including the measurements from the analyses called from the optimization controller are sent to the dataset. This can create a substantial amount of data. Default is "deselected".
	Goal expressions	When activated, it sends the results of the Goal's Expr field to the dataset for each Goal component used in the optimization controller. Default is "selected".
	Optimization variables	When activated, it sends the values of the design parameters associated with the optimization controller to the dataset. Default is "deselected".
	Current Error Function	When activated, it sends the value of the current error-function (EF) formulation used to the dataset. For more information, refer to <i>Error-Function (EF) Formulation</i> in <i>Summary of Optimizers</i> . Default is "deselected".
Output Data Control	Save data for iteration(s)	Last - Only the output data from the last (best) iteration is saved to the dataset. Nominal & Last - This is the default. Only the output data from the nominal and last (best) iterations are saved to the dataset. All - The output data for all iterations is saved. This can create a substantial amount of data.
	Update display during optimization	This is a built-in snapshot feature for optimization. It enables the real-time updates of the dataset and the real-time snapshot for each optimization iteration in the Data Display window. Note 1): the unattached measurement equations will not be in the snapshotted dataset. These equations are only available for the final dataset. So if the DDS has any plots for these measurement equations, these plots will only have the valid data at the end of the optimization. Note 2): This feature will slow down your simulation speed. For faster results, turn this feature off. Then the dataset and Data Display window will be updated only once at the end of the simulation. Default is "selected".
Levels	Status level	Enter a number for the desired annotation level. Levels are 0-4, with increasing information displayed in the Status window. Default is 4.

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Final Analysis		Used to specify an analysis run after your optimization is complete using the optimal design parameters. None - No Final Analysis. This is the default option. Any other analysis simulation control component that currently exists in your design such as Sweep1 or DC1. For more information, refer to Final Analysis.
Other	Seed	Seed is a value for the random number generator used to initiate an optimization. It only affects random optimizers including Random, Random Max, Random Min, Genetic and Discrete. For such optimizers, the results are reproducible with a fixed Seed. If Seed is not specified, the simulator chooses its own seed, which is different for each run and the results are un-reproducible.
	Starting Norm Order (P)	This parameter is only used for the Least Pth optimizer to define its starting norm order. Possible values for Starting Norm Order (P) are 2, 4, 8, or 16. Default is "2". For more information, see _Summary of Optimizers_.
	Initial Temperature	Applies only to the Simulated Annealing optimizer. The initial temperature and the number of shoots per iteration determine the annealing schedule mechanism for the system. The default value for the initial temperature is 0.1. The recommended range for it is 0.001 - 1000. For more information, see _Summary of Optimizers_.
	Number of Shoots Per Iteration	Applies only to the Simulated Annealing optimizer. It sets the maximum number of iterations the downhill simplex method uses for each temperature level, or for one iteration of simulated annealing method. It decides the thermodynamics state of the system in one iteration. The default value for it is 20. The recommended range is 10 - 5000. For more information, see _Summary of Optimizers_.
	Desired Error	Provides an additional stopping criteria for optimization. Represents the value of the error function that is acceptable to terminate the optimization. If you want all performance requirements to be met, accept the default of 0.0. Otherwise, specify an alternate value to cause the optimization to terminate sooner.
	Normalize goals automatically	Provides a built-in method to make the contributions from all associated goals equal to the error function. Useful when there is more than one Goal component associated with the optimization controller. The desired performances can be different in orders, which can result in the error function being biased to the responses with larger values. To prevent any one goal from dominating the error function, use either of the following methods: - Manually set up the weight factor correctly for each Goal component according to the performance values. - Let weight factor for each Goal component set automatically to the appropriate value for the performance level. For more information, refer to <i>The Weighting Factors</i> .
	Set best values for parent optimization	Used for the advanced optimization features including swept optimization and programmable optimization.
	Enable Optimization Cockpit	Used to decide the optimization running mode: optimization cockpit mode (realtime communication mode), or batch mode.

Displaying Analysis Data on the Schematic

The Display tab of the Nominal Optimization dialog box is used to select the parameters that will be displayed on your schematic related to nominal optimization. The same Display tab and procedure is used for yield analysis and yield optimization described in *Using Statistical Design (optstat)*. Generally, it is not necessary to modify the visible parameters. Check the appropriate boxes in the Display tab dialog box (the default is all boxes are checked).

Make specifications as follows:

- To select or deselect any parameter, click in the appropriate checkbox.
- Click **Set All** to check all parameters. Click **Clear All** to uncheck all parameters.
- After all selections have been correctly made, click **Apply** if you want to continue entering data in the other tabs of this dialog box. If you are finished entering data in this dialog box, click **OK**. Above is a nominal optimization dialog box example.

Note
MaxTrials, *Enable* and *RestoreNom* are reserved parameters and are not currently available. Selecting these parameters in the *Display parameter on schematic* section will display the parameters on the schematic; however, changes to the parameter values will not be recognized.

Display parameter	Description
OptimType	Optimization Type
ErrorForm	Error Function Formulation
MaxIters	Maximum number of Iterations
P	Starting Norm Order (P)
DesiredError	Desired Error
StatusLevel	Status Level
FinalAnalysis	Final Analysis
NormalizedGoals	Normalized Goals
SetBestValues	Set Best Values
Seed	Seed
SaveSolns	Save Solutions
SaveGoals	Save Goals
SaveOptimVars	Save Optimization Variables
UpdateDataset	Update Dataset
SaveNominal	Save Nominal values
SaveAllIterations	Save All Iterations
UseAllOptVars	Use All Optimization Variables
OptVar	Optimization Variables
UseAllGoals	Use All Goals
GoalName	Goal Name
SaveCurrentEF	Save Current Error Function
InitialTemp	Initial Temperature
NumShootsPerIter	Number of Shoots Per Iteration
EnableCockpit	Enable Optimization Cockpit Running Mode

Running Optimization

The optimization procedure starts on clicking *Optimize* button. According to the setup of the optimization controller, there are two different running modes:

1. Cockpit Mode: Optimization cockpit allows you to monitor and control the optimization. It not only allows you to monitor the optimization history, but also adjust the optimization settings. It allows you to adjust optimization inputs (optimization variables, optimization goals, optimizers), store/recall states, update designs, perform tuning, etc. For more information, see *Optimization Cockpit* (optstat).
2. Non-cockpit Mode: Non-cockpit mode is a batch mode. Once the optimization starts, you cannot do any adjustment for it. The status window will report the optimization progress.

Updating Design on the Design

There are two different approaches to update design functionality:

1. Updating Design Within Cockpit
While the cockpit is up, you can update the current view state to the design. The state concept includes:
 - Optimization Goals
 - Optimization Variables/Parameters, including both the nominal values and the range setups
 - Optimization Controller
If any field is an expression, then that field won't be updated in this mechanism with a warning information.
2. Updating Design Outside Cockpit
This mechanism will only update the nominal value of the optimization variables. It won't update any optimization variable/parameter range setup, any goals and optimization controllers. It is only available after the dataset for the optimization exists.
Select **Simulate > Update Optimization Values** if you want to update your Schematic window with the new parameter values resulting from a successful optimization. If you want to save the design at this point, select **File > Save** or **File > Save As** and assign an appropriate name.

Advanced Optimization

Using a *ParamSweep* component can form more flexible applications. Nominal optimization, yield analysis, yield optimization, and design of experiments (DOE) can all be swept as any other ADS analysis. When these analysis controllers are referenced by a parameter sweep controller, the nominal optimization, yield analysis, yield optimization, or DOE is performed for each value of the sweep variable, and/or is executed in the order defined in the parameter sweep controller. Any level of sweep nesting can be used and multiple optimization, yield, yield optimization, or DOE controllers may be referenced at any level.

Swept Optimization Methodology

Swept optimization enables designers to optimize any circuit at any swept parameter value. For example, a circuit can be optimized at any temperature level with the temperature being the swept variable. Similarly, a digital programmable attenuator can be optimized at every level of attenuation with the attenuation voltage being the swept variable. Swept optimization is achieved by using a Parameter Sweep Controller referring to an Optimization Controller, and that Parameter Sweep Controller goes to sweep some variable/parameter.

One of the key concerns for the swept optimization is the starting values of the design parameters for the optimization for each value of the sweep variable. It is controlled by the *Set best values for parent optimization* checkbox on the Parameters tab of the Optim components Nominal Optimization dialog box (see *Setting Parameter Information* (optstat)). For each value of the sweep variable, the optimization can either start from the nominal values of the design parameters (if *Set best values for parent optimization* is not activated), or it can start from the optimal values of the previous optimization results (if *Set best values for parent optimization* is activated). When *Set best values for parent optimization* is activated, the optimal values are saved internally so that they can be either user-updated or utilized by a subsequent analysis.

See also *Swept Optimization Example* (optstat).

Final Analysis

After the optimal design is obtained, several analysis are usually applied to check its time-domain and/or frequency domain performance. They are called Final Analysis. Final analysis is useful when the analysis executed by the optimizer uses a different sweep grid than the one you want for your output. For example, if a coarse grid is required for optimization, but a finer grid, or a different range, is desired for output, then the analysis setup to generate this finer grid may be run after the optimization is completed using the Final Analysis feature.

Either of the following approaches works for this final analysis purpose:

- After the optimization process finishes, *Update Optimization Values* to get the optimal design. Then deactivate the optimization controllers. Place the additional analysis controllers on the schematic and repeat the simulation.
- Use a *parameter sweep* analysis to sequence the optimization with additional analysis.
- Use the *Final Analysis* feature with nominal optimization

Built-in Final Analysis

The *Final Analysis* drop-down list (see *Setting Parameter Information* (optstat)) is used to specify an analysis run after your optimization is complete while using the optimal design parameters. This analysis can be of any analysis controller component that does not introduce circularity in analysis execution. The drop-down list enables you to select *None* (for no Final Analysis, the default) or any analysis simulation control component currently in your design.

Multiple analysis can run as final analysis by grouping them together in a parameter sweep (without specifying a sweep variable), and choosing this sweep in the Final Analysis drop-down list. Note that for swept or nested optimization, the Final Analysis parameter should not refer to any controller that is already executing. Data output for Final Analysis follows the information (flags) set in the companion *Optim* component.

See also *Final Analysis Example* (optstat).

Final Analysis using Parameter Sweep

Using a ParamSweep component to group an optimization controller and some additional final analysis. The ParamSweep component defines the analysis order: first the

optimization, then the first final analysis, then the next final analysis etc. For example, an optimization can reference an S-parameter analysis with fewer frequency points. This makes the optimization more efficient. Then, a follow-on *final analysis* can be specified that uses additional frequency points to display the results of the optimization. Note that you must check the control parameter, *Set best values for parent optimization*, in the Optim component so that the final analysis always uses the optimal results of the design parameters.

Sequential Optimization Methodology

Sequential optimization is an extension of swept optimization. Using a ParamSweep component to group several optimization controllers, each of which using a different optimizer but associated with the same optimization variables and goals. For example, you might sequence random optimization with gradient optimization in an effort to attain a more robust and fast solution. In this case, you use the random optimizer to locate the results around local/global minimum. Then the gradient optimizer reaches the minimum results fast. Note that you must check the control parameter, *Set best values for parent optimization*, in the Optim component so that the following optimizer starts from the optimal results of the previous optimizer.

Programmable Optimization Methodology

Programmable optimization is an extension of sequential optimization. Using a ParamSweep component to group several optimization controllers, each of which is associated with different optimization variables and goals. The ability to designate a particular subset of both optimization variables and goals is accomplished by using the *OptVar* and *OptGoal* parameter tabs (Optimization Variables and Optimization Goals) in the *Optim* component (see *Selecting an Optimizer and Goals* (optstat)). One example for this application is to simulate a full-blown tuning/test procedure. For this example, the control parameter, *Set best values for parent optimization*, in the Optim component must be checked so that the following optimizers start from the optimal results of the previous optimizer.

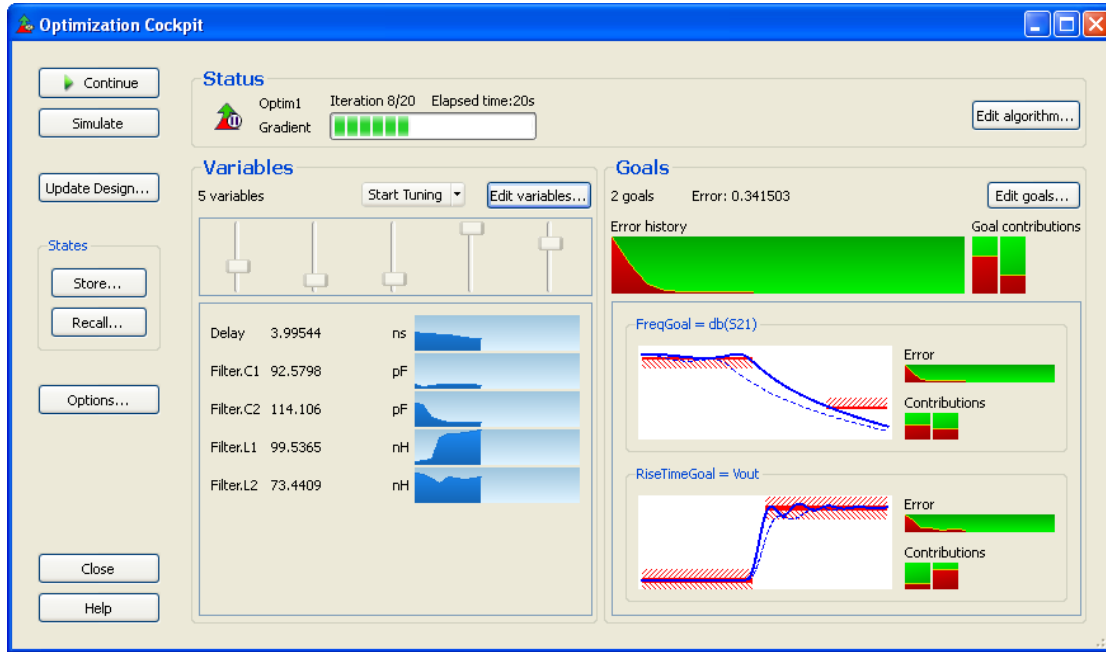
See also *Programmable Optimization Example* (optstat).

Optimization Cockpit

The Optimization Cockpit is a live, graphical view of an optimization job. You see the cockpit data (error graphs, goal plots, and variable values) change in real time as the optimization progresses. In addition, you can use the cockpit to control the optimization while the optimization is running. For example, during the course of an optimization, you can increase the range of an optimization variable, modify the limit line of a goal, tune the optimization variables, and change the algorithm from Random to Gradient.

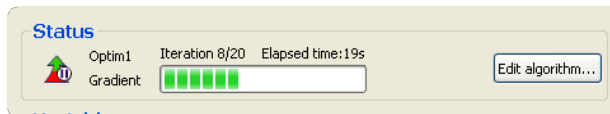
Cockpit Panels

The cockpit has three main panels: Status, Variables, and Goals. It also has a control panel on the left-hand side.



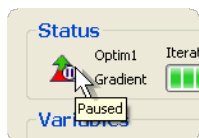
Status panel

The Status panel displays the optimizer's status, type, elapsed time, and progress. It also has a button for changing the optimization algorithm settings.



Activities

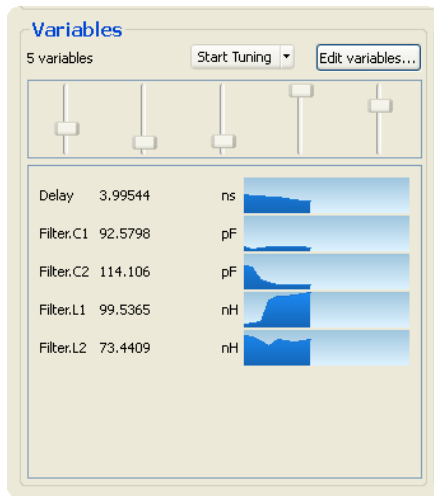
- Roll the pointer over the status icon to know the state of the optimizer



- While the optimizer is paused or towards the end of an optimization, click **Edit Algorithm** to modify the optimization algorithm. See [Modifying the algorithm](#) for more information.

Variables panel

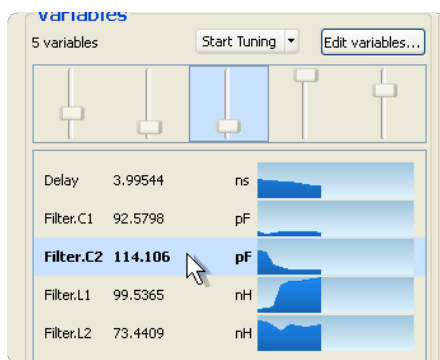
The Variables panel displays the optimization variables as a row of sliders and as data in tabular format. It also has buttons for tuning and for editing the variable definitions.



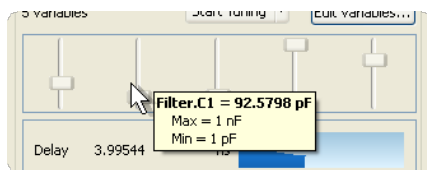
- The sliders represent the current value of each optimization variable relative to its minimum and maximum values.
- Each row of the table shows important information for each optimization variable:
 - Name
 - Current value
 - Units (if any)
 - A history graph showing the values of the variable over the course of the optimization.

Activities

- Click a row in the table to highlight its corresponding slider or vice-versa



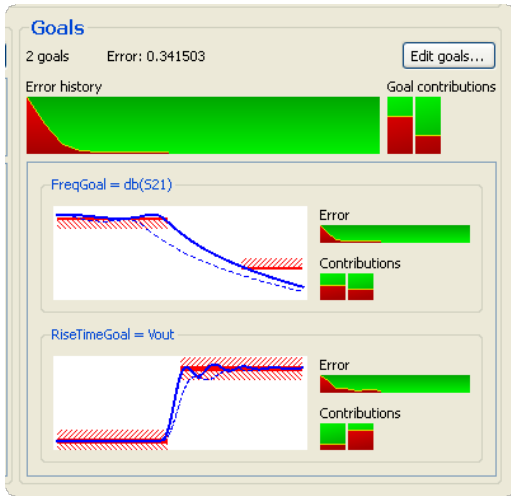
- Move the pointer over a slider to see information about that variable



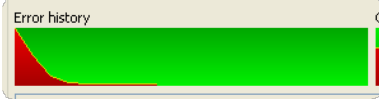
- Click **Start Tuning** to switch to tune mode. See [Tuning](#) for more information.
- While the optimizer is paused or towards the end of an optimization, click **Edit Variables** to modify the variable settings. See [Modifying variables](#) for more information.

Goals panel

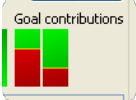
The Goals panel displays the current error, the error history graph, the goal contribution histogram, and the goals table. It also has a button for editing the goal definitions.



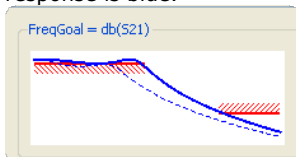
- The *Error history* graph shows the overall error over the span of optimization.



- The *Goal contributions* histogram is displayed when there are two or more goals. It represents each goal's contribution to the overall error.



- The table contains a row for each goal. Each row represents the following information:
 - A plot of the goal's response and limit lines. The limit lines are red and the response is blue.



The solid blue trace is the response for the current values of the optimization variables. The dashed blue trace is the response at the beginning of the optimization.

- A history plot of the goal's error over the span of optimization.



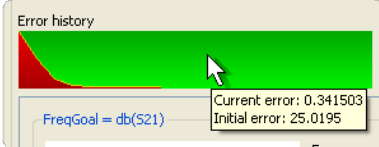
ⓘ Sometimes you will notice the error for a goal increases. The optimizer is driving the overall error to zero, so the error for a particular goal can increase as long as the overall error decreases.

- A Contributions histogram is displayed only when the goal has more than one limit line. The histogram shows each limit line's contribution to the goal's error.

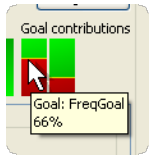


Activities

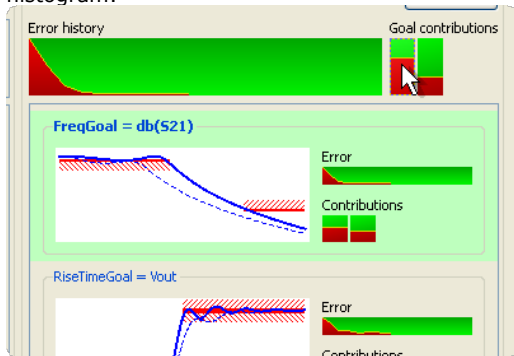
- Move the pointer over the *Error history* graph to see information about the error:



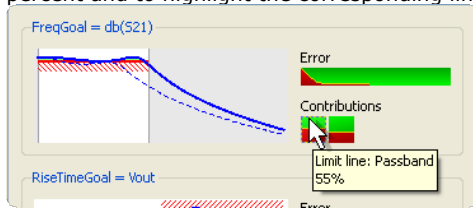
- Move the pointer over a bar in the *Goal contributions* histogram to see information about that goal's contribution to the error:



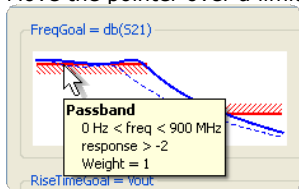
- Click a bar in the Goal Contributions histogram to highlight the corresponding goal in the table or click a goal in the table to highlight the corresponding bar in the histogram:



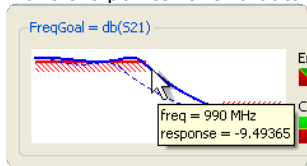
- Move the pointer over a bar in the *Contributions* histogram to display the contribution percent and to highlight the corresponding limit line on the plot:



- Move the pointer over a limit line to see information about that limit line:



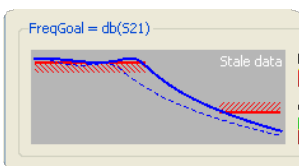
- Move the pointer over a data point in the trace to see the coordinates of the point:



- While the optimizer is paused or towards the end of an optimization, click **Edit Goals** to modify the goal settings. See [Modifying goals](#) for more information.

Stale plots

Goal plots go stale when the algorithm, goal, or variable settings change, but no simulation is performed.



This means that the simulation results (error value, the contribution graphs, and the goal plots) displayed in the Cockpit do not accurately reflect the current algorithm, goal, and variable settings. For example, click **Edit Goals** and modify one of the limit lines you will see the goals plot will go stale. Click **Simulate** to refresh the goals plot.

Control panel


This panel has several buttons on the left-hand side of the Cockpit that helps you to control the optimization while it is running.

Pause

Click **Pause** to pause a simulation while it is running. Once an optimization is paused, you can interact with the optimizer. It can take time for the optimizer to pause when the low-level simulation (for example, transient or harmonic balance) takes a while to run. The **Pause** button changes to **Continue** when an optimization is paused or is completed.

Continue

Click **Continue** to continue the optimization. This button is available when the optimizer is paused or has finished.

 If the optimizer has reached the maximum number of iterations and you click **Continue**, you can increase the maximum number of iterations.

Simulate

Click **Simulate** to re-evaluate the goals. This is useful after you have changed the settings (for example, modified a limit line on a goal) and you want to update the goals plots or see the new error value. It is also useful in **Simulate after pressing Simulate** tuning mode. See [Tuning modes](#) for more information.

Update Design

Click **Update Design** to transfer variable values to the design. This command can also transfer algorithm and goal data to the design. This is useful when you have used **Edit Algorithm** or **Edit Goal** to modify the algorithm or goal definitions. For more information on modifying the goal or algorithm settings, see [Controlling the Optimization](#).

Note: The options you specify while updating the design do not change from one optimization to other.

If the sweep panel is displayed (see [Sweep Panel](#)) when you click **Update Design**, the design is updated using the optimization specified by *View*.

Store

This option allows you to store the current optimization state. For more information, see [Using Optimization states](#).

Recall

This option helps you to recall a state. For more information, see [Using Optimization states](#).

Options

It allows you to view the Cockpit options. For more information, see [Scaling the plots](#).

Close

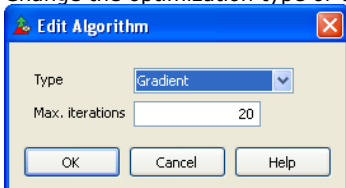
Click **Close** to close the Optimization Cockpit. If the optimization is still running, you have the option to stop it or to update the design.

Controlling the Optimization

Modifying the algorithm

Follow the steps below to modify the algorithm:

1. Click **Edit Algorithm**.
2. Change the optimization type or the maximum number of iterations.

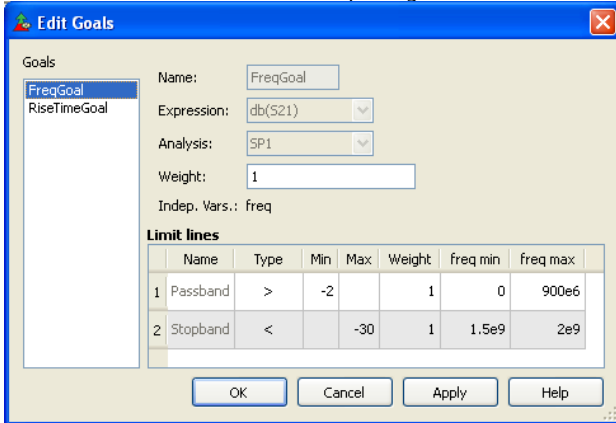


- Click **OK**.
Note: When you click **OK**, the Optimization component on the schematic is not updated.
- Click **Update Design** to transfer the new settings to the schematic.

Modifying goals

Follow the steps below to modify the goals definition:

- Click **Edit Goals** to view or modify the goal definitions.



- Goals are listed in the *Goals* list. Click an entry from the list to view its properties. You can change the goal's weight and also the definition of each limit line. From the cockpit, you cannot change a goal's name, expression, or analysis. Also, it doesn't allow you to add a new goal, delete an existing goal, or change the name of a limit line.

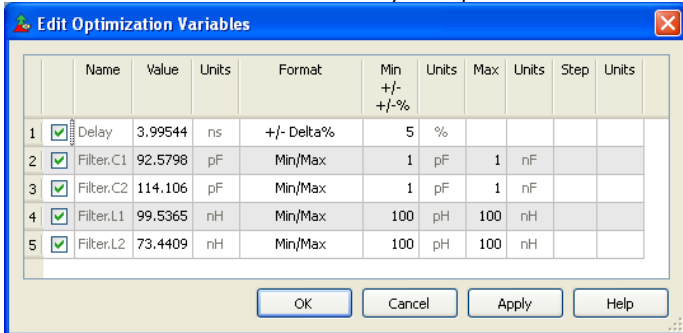
Set *Weight* = 0 to disable a goal or a limit line.

- After changing the goals definition, Click **OK**.
Note: When you click **OK**, the Goal components on the schematic are not changed.
- Click **Update Design** to transfer the new settings to the schematic.

Modifying variables

Follow the steps below to modify the variable settings:

- Click **Edit Variables** to view or modify the optimization variable settings.



- Select the check-box to enable that variable.
The edit variable dialog box allows you to change the nominal value, format, and format's min and max values. It doesn't allow you to change a variable's name, add a new variable, or delete an existing variable.
- After modifying the variable settings, Click **OK**.
Note: When you click **OK**, the variables on the schematic are not changed.
- Click **Update Design** to transfer the new settings to the schematic.

Note: Variables of **+/- Delta** or **+/- Delta%** *Format* are handled differently. For these variables, the minimum and maximum values for the variable are calculated as offsets from the nominal value. This calculation is performed at the beginning of the optimization and the minimum and maximum values do not change as the optimization progresses. In the above example, it seems as if **Delay** is defined as 3.99544 ns +/- 5%. This is not the case. At the beginning of the optimization, the nominal value for **Delay** was 4 ns, so the definition is actually 4 ns +/- 5%. When you edit any field for this variable and click **OK** or **Apply**, the variable's minimum and maximum are recalculated based on the value in the

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design table; otherwise, the variable's minimum and maximum are not recalculated.

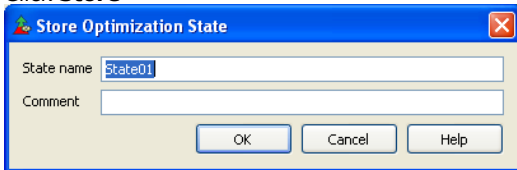
Using Optimization states

The cockpit has the ability to store and recall optimization states. This is useful when you explore the behavior of your design using the Optimization Cockpit. For example, before modifying some variable definitions or goal definitions, you can store the current optimization state and recall the saved state when required. The optimization state contains the algorithm, goal, and variable settings.


Store

Follow the steps below to store an optimization state:

1. Click **Store**



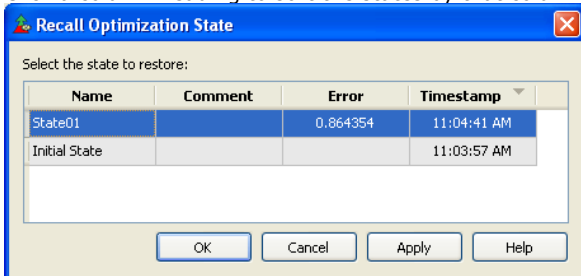
2. Specify the name of a state in *State name* field
3. Add a comment in the *Comment* field (optional)
4. Click **OK** to store the current optimization state

 The Optimization Cockpit automatically stores the initial state.

Recall

Follow the steps below to recall an optimization state:


1. Click **Recall**
2. Click a column heading to sort the states by that column



3. Select a state
4. Click **OK** or **Apply** to restore the selected state

When you recall a state, the cockpit's algorithm, goal, and variable settings are replaced with those of the recalled state. It is similar to manually reverting back to previous settings using Edit Algorithm, Edit Goals, and Edit Variables to restore the state. This means that the history graphs and goal graphs do not change to what they were when you saved the state.

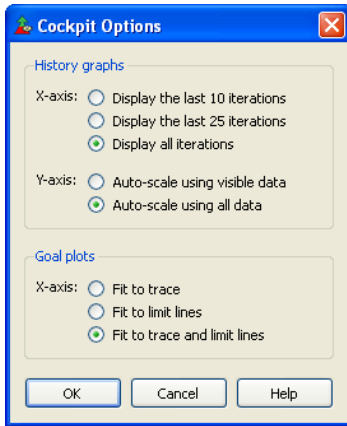
When you recall a state the goal plots will go stale (see [Stale plots](#)). Click **Simulate** to recalculate the error and update the goal graphs.

 The automatically stored initial state has no Error value associated with it because the state is stored before any simulation is performed.

Note: Optimization states are discarded when the Optimization Cockpit is closed. If you want to use an optimization state in your next optimization, click **Recall** to recall the state and then click **Update Design** to update the schematic.

Scaling the plots

Click **Options** to view or modify the Cockpit options.



- **History graph X-axis options** These options modify the X-axis scaling of all history graphs (both error and variable history graphs).
 - **Display the last 10 iterations** The graph displays only the 10 most recent iterations.
 - **Display the last 25 iterations** The graph displays only the 25 most recent iterations.
 - **Display all iterations** Scales the X-axis from 0 to Max. iterations.
- **History graph Y-axis options** These options modify the Y-axis scaling of the error history graphs.
 - **Auto-scale using visible data** Consider only the iterations that are visible in the graph when scaling the Y-axis.
 - **Auto-scale using all data** If there are iterations that are not visible on the graph (for instance, the graph shows the last 10 iterations, but the current iteration is greater than 10) then consider the invisible iterations when scaling the Y-axis.
- **Goal plots** These options modify the X-axis scaling of all the goal plots.
 - **Fit to trace** Set the X-axis scaling to show the entire trace.
 - **Fit to limit lines** Set the X-axis scaling to show all the limit lines.
 - **Fit to trace and limit lines** Set the X-axis scaling to show all the limit lines and the entire trace.

Note: All of these options are remembered from one optimization to the next.

Tuning

Click **Start Tuning** to start tuning. If the optimization is running, it will pause.

While tuning, the **Start Tuning** button is renamed as **Stop Tuning**.

To disable the tune mode:

- Click **Stop Tuning**, or
- Click **Continue** to resume the optimization.

You can modify the optimization variable values using the sliders or by entering values directly in the variables table.

Note: The minimum and maximum values of the variables are determined by the optimization variable definitions. The settings used by the standard tune mode are not used. To change the minimum and maximum values for a variable, click **Edit Variables**.

DDS is also integrated with the tuning process. Once **Start Tuning** button is activated, the plots of the results before tuning in Cockpit and DDS will not refresh. The tuning results will then be added to those plots to compare.

Figure: Tuning Variables

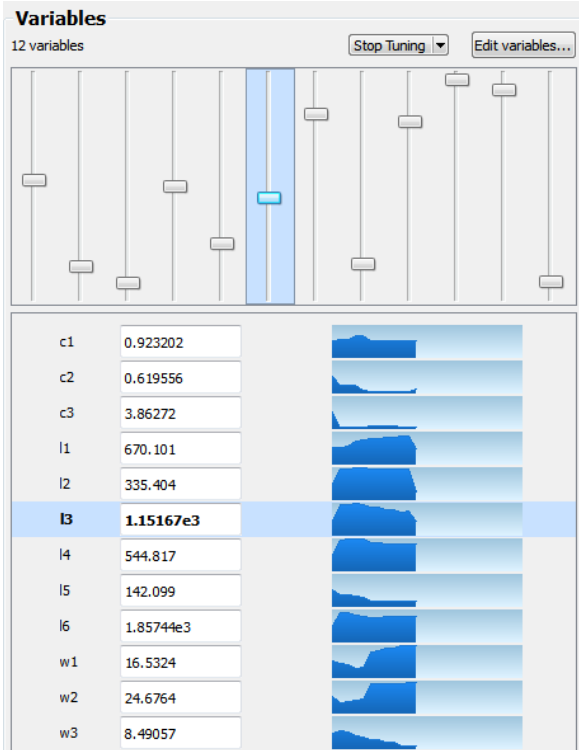


Figure: Goals

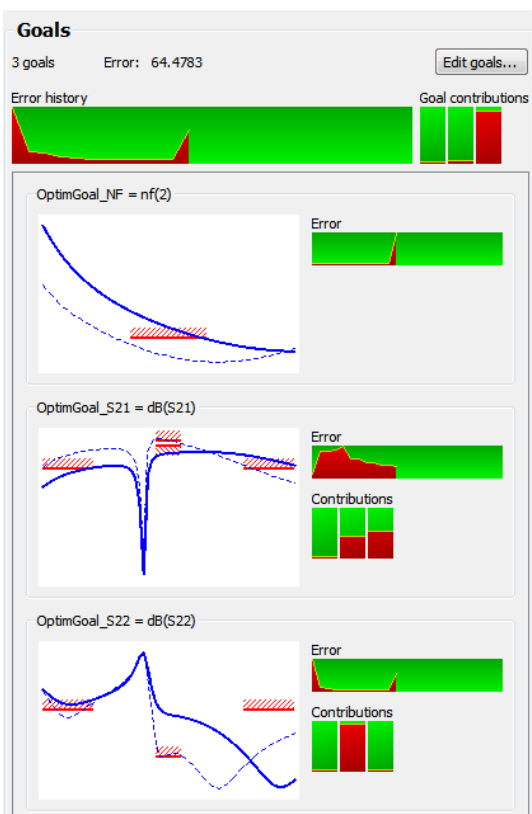
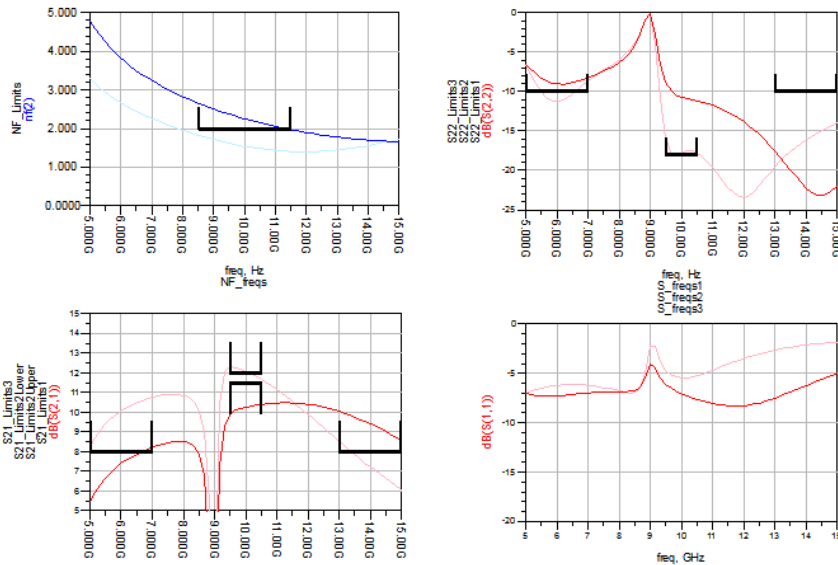


Figure: DDS plots

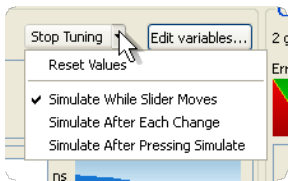


Activities

- Click a row in the variables table to highlight the corresponding slider. Click a slider to highlight the corresponding row.
- On the keyboard, press *Tab* (or *Enter*) to move down in the variable table. Press *Shift+Tab* (or *Shift+Enter*) to move up.

Tuning menu

The **Start Tuning** (or **Stop Tuning**) button has a menu attached to it. Click the arrow to the right of the button to access the menu.



Reset values

Choose **Reset Values** to reset the optimization variable values to the values they had when you started tuning.

Use the **Store** and **Recall** buttons to store and recall intermediate tuning states. For more information, see [Using Optimization states](#).

Tuning modes

Tuning has three different modes. Use the Tuning menu to change modes.

- **Simulate while slider moves** - In this mode, the simulator performs simulations while you are moving the slider. This is useful for faster simulations (less than 0.5s).
- **Simulate after each change** - In this mode, the simulator waits to perform a simulation until you release the slider. This is useful when the simulations are not as quick (0.5s to 5s).
- **Simulate after pressing Simulate** - In this mode, no simulations are performed until you click **Simulate**. This mode is useful when the simulation takes longer (greater than 5s) or you know in advance that you want to make several changes before simulating.

Note: This setting is remembered from one optimization to the next.

View-only mode

The Cockpit switches to view-only mode when either of the following is true:

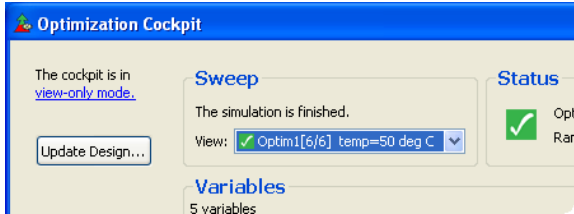
- The top-level simulation controller is not an optimization. For example, there is a **ParamSweep** sweeping an optimization.
- There are two or more top-level simulation controllers. For example, there is a top-

level harmonic balance controller and a top-level optimization for different simulation, but not for HB analysis.

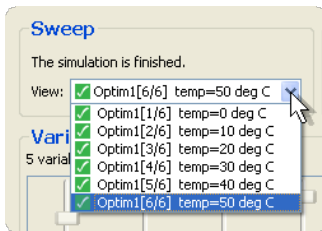
In view-only mode, the cockpit still displays the real-time optimization data, but none of the interactive features is available.

Sweep Panel

The sweep panel appears when there is more than one optimization in the simulation job. This happens when an optimization is swept (using a **ParamSweep** component) or when there is more than one optimization in the simulation job.



Use the *View* drop-down list to select an optimization to be displayed by the cockpit.



Changing the view has no effect on the simulation, so you can view the results of the first point in the sweep while the rest of the sweep is running. When you click **Update Design**, the design is updated using the optimization specified by *View*.

A red icon in the *View* list indicates that the optimization was terminated because of a simulation error. A green icon indicates that no simulation error occurred.

i The green icon does not indicate that the error value for that optimization is zero. The error will be non-zero if the optimization was terminated for other reasons (for example, maximum iteration limit reached, gradient is zero).

Turning off the Optimization Cockpit

You can turn off the Optimization Cockpit by clearing the **Enable Optimization Cockpit** checkbox on the *Parameters* tab of the Optimization component.

Summary of Optimizers

This topic provides details on the various optimizers available for performing nominal optimization. For details on setting up a nominal optimization, refer to *About Nominal Optimization* (optstat).

The optimizers are differentiated by their *error function formulations* and *search methods*.

- The *error function formulation* measures the difference between the computed and desired responses. The smaller the value of the error function, the more closely the responses coincide. For more information, refer to [Error-Function \(EF\) Formulation](#).
- The *search method* determines how the optimizer arrives at new parameter values. When optimizers execute their search method, they substitute new parameter values to effect a reduction in the error function value. For more information, refer to [Search Methods](#).

For more information on the individual optimizers, refer to [Available Optimizers](#).

Obtaining Global Optimal Results

Most of the optimizers provided are local optimizers. [Genetic Optimizer](#) and [Simulated Annealing Optimizer](#) are global optimizers and [Random Optimizer](#) has the capability to find the global optimal solutions. If you really want a global solution, try Simulated Annealing optimizer first, then Genetic optimizer. You can also try to use Random optimizers (without fixed Seed) multiple times to select the *global* solution.

Available Optimizers

When you select an optimizer, the system automatically sets both the *error-function (EF) formulation* and the *search method* to be used. The available optimizers and their associated error-function formulations and search methods are shown below.

While there are a number of optimizers available, the *Random*, *Gradient*, and *Simulated Annealing* optimizers tend to be the most frequently used because they work for most cases.

Available Optimizers and Associated EF & Search Method

Optimizer	Error-Function (EF) Formulation	Search Method
Random Optimizer	Least-Squares EF	Random Search
Gradient Optimizer	Least-Squares EF	Gradient Search
Random Minimax Optimizer	Minimax L1 EF	Random Search
Gradient Minimax Optimizer	Minimax L1 EF	Gradient Search
Quasi-Newton Optimizer	Least-Squares EF	Quasi-Newton Search
Least Pth Optimizer	Least Pth EF	Quasi-Newton Search
Minimax Optimizer	Minimax EF	Gauss-Newton/Quasi-Newton (minimax) Search
Random Max Optimizer	Negated Least-Squares EF	Random Search
Hybrid Optimizer	Least-Squares EF	Random Search and Quasi-Newton Search
Discrete Optimizer	Least-Squares EF	Exhaustive Search
Genetic Optimizer	Least-Squares EF	Genetic Algorithm Search
Simulated Annealing Optimizer	Least-Squares EF	Simulated Annealing Algorithm Search
Sensitivity Analysis†		

† Note that Sensitivity Analysis appears in the list of available optimizers. Sensitivity Analysis is not actually an optimization process. It is a fundamental element of gradient optimization.

Available Types for Optimization-Variables

Some optimizers may work with discrete valued parts, while others must work with continuous valued parts. It is important to take care when selecting an optimizer because different optimization problems require different optimizers for better performance and accuracy.

Optimizers and Optimization-Variable Types

Optimizer	Operates with Discrete Variables	Operates with Continuous Variables
Random	Yes	Yes
Gradient	No	Yes
Random Minimax	Yes	Yes
Gradient Minimax	No	Yes
Quasi-Newton	No	Yes
Least Pth	No	Yes
Minimax	No	Yes
Random Max	Yes	Yes
Hybrid	No	Yes
Discrete	Yes	No
Genetic	Yes	Yes
Simulated Annealing	No	Yes

The difference between continuous and discrete variables is relatively simple.

- A *continuous* type variable can take any real value between a specified range.
- A *discrete* type variable is only allowed to take a specific list of values between a specified range.

For more information, refer to *Value Types for Nominal Optimization* (optstat).

Random Optimizer

The Random optimizer uses the *Random search method* to arrive at new parameter values by using a random-number generator, that is, by picking a number at random within a range. Starting from an initial set of parameter values for which the error function is known, a new set of values is obtained by perturbing each of the initial values, and the error function is re-evaluated. Then the error function is re-evaluated by reversing the algebraic sign of each parameter value perturbation. These two values, corresponding to positive and negative perturbations, are compared to the value at the initial point. If either value is less than the initial value, then the set of parameter values for which the error function has its least value becomes the initial point for the next iteration. If neither value is less than the initial value, then the initial point remains the same for the next iteration. Since random search uses a pseudo-random generator, the results can be different for two optimization procedures.

The Random optimizer uses the *Least-Squares error function* to minimize the average weighted violation for the desired responses. So the value for the error function represents the average weighted violation for the desired responses and the value of zero indicates that all of the intended performance goals have been reached. The Random optimizer guarantees to find at least one local minimum result. It also has the probability to find the global minimum result. The Random optimizer is probably the best optimizer for the following cases when considering the average violation for the performance goals:

- There are continuous and discrete optimization variables in the design
- The number of optimization variables is large (the effort for each iteration is almost independent of the number of optimization variables)

See Also

- [Least-Squares EF](#)
- [Random Search](#)

Gradient Optimizer

The Gradient optimizer uses the *Gradient search method* to arrive at new parameter values using the gradient information of the network's error function. The gradient of the error function indicates the direction to move a set of parameter values in order to reduce the error function. For each iteration, the error function and its gradient is evaluated at the initial point. Then the set of parameter values is moved in that direction until the error function is minimized. A single iteration usually includes many function evaluations; therefore, an iteration in the gradient search method takes much longer than the random search method.

The Gradient optimizer uses the *Least-Squares error function* to minimize the average weighted violation for the desired responses. So the value for the error function represents the average weighted violation for the desired responses and the value of zero indicates that all of the intended performance goals have been reached. The Gradient optimizer guarantees to find a local minimum result. A design that is optimized by the gradient optimizer has the least sensitivity (more stable) to slight variations in its

parameter values.

The Gradient optimizer is the best optimizer to use for simple circuits with straightforward requirements; that is, the larger number of function evaluations will not slow the optimization appreciably, but the optimizer will converge on a solution quickly. The Gradient optimizer is also quite good at following contours.

See Also

- [Least-Squares EF](#)
- [Gradient Search](#)

Random Minimax Optimizer

The Random Minimax optimizer uses the *Random search method* to arrive at new parameter values. The procedure is the same as that described for the Random optimizer.

The Random Minimax optimizer uses the *Minimax L1 error function* to minimize the point that constitutes the greatest violation for the desired responses. So the value for the error function reveals the greatest violation for the weighted desired responses and the value of zero indicates that all of the intended performance goals are satisfied. Just like the Random Optimizer, the Random Minimax optimizer guarantees to find at least one local minimum result and has the probability to find the global minimum result.

The Random Minimax optimizer is probably the best optimizer for the following cases when considering the greatest violation for the performance goals:

- There are continuous and discrete optimization variables in the design
- The number of optimization variables is large (the effort for each iteration is almost independent of the number of optimization variables)

See Also

- [Minimax L1 EF](#)
- [Random Search](#)

Gradient Minimax Optimizer

The Gradient Minimax optimizer uses the *Gradient search method* to arrive at new parameter values. The procedure is the same as that described for the Gradient optimizer.

The Gradient Minimax optimizer uses the *Minimax L1 error function* to minimize the point that constitutes the greatest violation for the desired responses. So the value for the error function reveals the greatest violation for the weighted desired responses and the value of zero indicates that all of the intended performance goals are satisfied. Just like the Gradient Optimizer, the Gradient Minimax optimizer guarantees to find a local minimum result. A design that is optimized by a gradient minimax optimizer has the least sensitivity (more stable) to slight variations in its parameter values.

The Gradient Minimax optimizer is the best optimizer to use for simple circuits with straightforward requirements; that is, the larger number of function evaluations will not slow the optimization appreciably, but the optimizer will converge on a solution quickly. The Gradient Minimax optimizer is also quite good at following contours.

See Also

- [Minimax L1 EF](#)
- [Gradient Search](#)

Quasi-Newton Optimizer

The Quasi-Newton optimizer uses the *Quasi-Newton search method* to arrive at new parameter values. The Quasi-Newton search method uses the second-order derivatives of the error function and the gradient to find a descending direction. The second-order derivatives are estimated by the Davidson-Fletcher-Powell (DFP) formula or its

complement. Appropriately combined with the gradient, this information is used to find a direction and an inexact line-search is conducted. Much like the optimizers using the gradient search method, an iteration in the optimizers using the Quasi-Newton search method consists of many function evaluations. Therefore, a single iteration using the Quasi-Newton search method takes longer than an iteration in the optimizers using the Random search method.

The Quasi-Newton optimizer uses the *Least-Squares error function* to minimize the average weighted violation for the desired responses. So the value for the error function represents the average weighted violation for the desired responses and the value of zero indicates that all of the intended performance goals have been reached. The Quasi-Newton optimizer guarantees to find a local minimum result.

See Also

- [Least-Squares EF](#)
- [Quasi-Newton Search](#)

Least Pth Optimizer

The Least P^{th} optimizer uses the *Quasi-Newton search method* to arrive at new parameter values.

The Least P^{th} optimizer uses the *Least Pth error function* formulation, in which the error function is equal to the P^{th} power of the difference between the simulated responses, where $p = 2, 4, 8, \text{ or } 16$. The optimizer automatically increases p in this sequence. This emphasizes the errors that have high values much more strongly than those that have small values. As p increases, the Least P^{th} error function approaches the minimax error function.

The Least P^{th} optimizer allows the error function to become negative when you specify a performance window and the response moves inside of that window. For example, there may be a minimum and maximum gain specification on an amplifier and the Least P^{th} optimizer can go beyond the specification and place the gain halfway between the two limits. The Least P^{th} optimization routine is the exponential sum of the error function, where the exponent p is not necessarily equal to 2. It can be a positive number, usually an integer. The Least P^{th} formulation is used as an indirect method to achieve a minimax design. Minimax error function can contain edges or discontinuities in their derivatives. These occur at points where the error contributions due to different goals intersect in the parameter space. The Least P^{th} error functions avoid this problem.

For a large value of p , the errors having the maximum value are more strongly emphasized over the other errors, that is, they are given higher priority in optimization. As p increases to infinity, the Least P^{th} formulation leads to a minimax error function. The problem is solved through a sequence of Least P^{th} optimizations with p being gradually increased. The sequential Least P^{th} optimization used in the program uses $p = 2\ 4\ 8\ 16$. This strategy often provides a smooth path towards a minimax solution.

See Also

- [Least Pth EF](#)
- [Quasi-Newton Search](#)

Minimax Optimizer

The minimax optimizer consists of two stages. In the first stage of the algorithm, the *Gauss-Newton search method* solves a minimax problem using a linear programming technique. In doing so, the status and potential of each individual error function component are analyzed. Its contribution to the minimax problem is mathematically assessed and taken into account during optimization. In the second stage, the optimizer works with a *Quasi-Newton search method* using approximate second-order derivatives. Such extra effort becomes necessary for an accurate and efficient solution to certain ill-conditioned problems (i.e., singular problems).

Using the *minimax error function formulation*, the Minimax optimizer calculates the difference between the desired response and the actual response over the entire measurement parameter range of optimization. Then the optimizer tries to minimize the

point that constitutes the greatest difference between actual response and desired response. The minimax optimizer terminates when responses become optimally equal-ripple or the relative change in the variables is less than 0.05 percent. It also stops when the number of iterations is reached. Note that the bounds imposed on the variables are formulated and treated directly as linear constraints without having to resort to variable transformation; therefore, a source of nonlinearity is eliminated.

See Also

- [Minimax EF](#)
- [Gauss-Newton/Quasi-Newton \(minimax\) Search](#)

Random Max Optimizer

The Random Max optimizer uses the *Random search method*. The Random Max optimizer uses the same general process as the Random optimizer.

The Random Max optimizer uses the *Negated Least-Squares error function* which provides for a worst-case analysis. As the name implies, the optimizer internally negates the least-squares error function so that the effect is a maximization of the error function.

See Also

- [Negated Least-Squares EF](#)
- [Random Search](#)

Hybrid Optimizer

The Hybrid optimizer combines the *Random* and *Quasi-Newton search methods*. It offers a compromise between the ability to find a minimum quickly, using the fewest possible circuit analyses (this is the strength of Quasi-Newton optimization), and the possibility to find the global cost minimum in the presence of many local minima (this is the strength of Random optimization).

When hybrid optimization is chosen, the system starts out using the *Quasi-Newton search method*, and quickly finds the nearest local minimum. When the gradient approaches zero, it is near a minimum and can do little to decrease the cost function. The system then uses the *Random search method* to generate a new initial guess. The *Quasi-Newton search method* is then performed, starting at this initial guess. This process continues until the optimizer can no longer improve the performance of the circuit, or has reached the maximum number of iterations that has been specified. For this reason, hybrid optimization is an excellent choice if time is available for very long analyses.

Hybrid optimizer has two useful properties:

- It uses gradient information during the optimization.
- It is useful for finding a local minimum, and it has some probability to find the global minimum.

The Hybrid optimizer also uses the *Least-Squares error function*.

See Also

- [Least-Squares EF](#)
- [Random Search](#)
- [Quasi-Newton Search](#)

Discrete Optimizer

The Discrete optimizer uses the *exhaustive search method* exclusively. This optimizer only affects parameter values specified as discrete-valued optimization variables. The exhaustive search method involves a comprehensive search for the combination of discrete values that results in the best design performance. Starting from an initial set of parameter values for which the error function is known, an update in the parameter values occurs upon an improvement in the error function. Because the parameter values that may change are not continuous variables, this search method is more a series of

trials than iterations. Moreover, the number of trials required to attempt all combinations may often be prohibitive. To reduce optimization time, keep the number of discrete variables to a minimum and reduce the number of values the discrete variables may take on.

The Discrete optimizer also uses the *Least-Squares error function*.

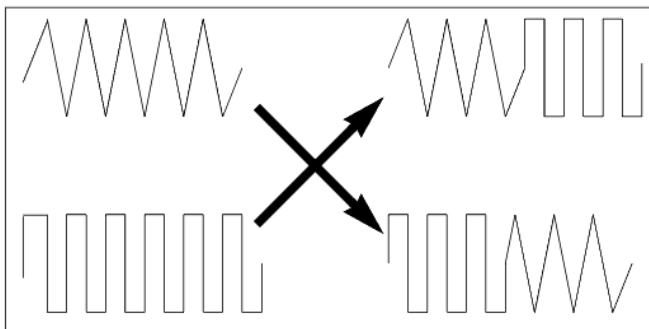
See Also

- [Least-Squares EF](#)
- [Exhaustive Search](#)

Genetic Optimizer

Genetic algorithms (GA's) provide another direct search optimization method. The basis of the procedure is a set of trial parameter sets, sometimes called chromosomes, which are allowed to evolve towards a set that gives progressively better performance. The key to the genetic optimization is the strategy of change, sometimes likened to survival of the fittest. The idea is that with each change in the parameter population, that is, each generation of parameters, the performance given by the parameter population improves. This whole process is achieved using a five-step process called (1) Representation, (2) Evaluation, (3) Reproduction, (4) Breeding and Crossover, and (5) Mutation as described below.

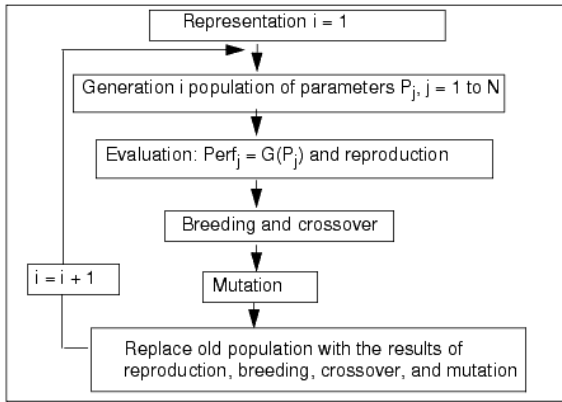
1. *Representation* : Genetic algorithms require the input parameter set to be represented as a string of digits. It is straightforward to map each parameter onto the interval 0 to 1, for instance, and then have each of the n parameters occupy a position in the string of n bounded numbers. The algorithm then manipulates and optimizes this string of numbers as a whole. An individual string of parameters is called an element within the population of parameter strings.
2. *Evaluation* : Each generation of parameters begins with a performance evaluation of each string in the population. Usually this involves determining the performance $G(P)$ for each representation of P in the population. Each element is then graded as to how well it performed, often using an error function, known as the *fitness function* .
3. *Reproduction* : Some of the members of the population for this generation are copied, that is, reproduced, and added to the next generation population. The number of copies depends on the performance evaluation. The elements that perform well are copied several times, and those with poor performance are not copied at all. The copies, or offspring, then make up the next generation. Elements that are not copied are not represented in the next generation. Note that the number of elements in each generation is constant. There are several suggested methods of ranking and reproduction, including ratioing, where the number of copies is directly related to the element's performance, and ranking where the performances are ranked, with the top performers being copied more times than the lower ranked performers.
4. *Breeding and crossover* : The previous step, Reproduction, produced a population of strings where each evaluated well. Breeding then combines parts of two strings to form two different and new strings. In this way good representations are mixed with poorer representations, with the result eventually being evaluated in the next generation of the algorithm. There are many methods for breeding; the most common is crossover. Crossover typically takes two elements, splits them at a random location in the string, and swaps the two parts to create two new strings (see the following figure). This provides a controlled statistical exploration of the performance space.



Breeding and Crossover in the Simple Genetic Algorithm

5. *Mutation* : The last step in creating a new generation of elements is the random changing of parameters in some of the surviving strings. This comprises a completely random search of the performance space, and can be viewed as the injection of

information into the surviving population. The following figure presents a completed flow diagram of the genetic algorithm. The application of these techniques requires many tuning parameters that are not available with yield optimization. Genetic optimization techniques will prove useful for many complex optimization problems, including discrete value and tolerance optimization.



Random-Search Optimization Using a Genetic Algorithm

See Also

- [Least-Squares EF](#)
- [Genetic Algorithm Search](#)

Simulated Annealing Optimizer

Simulated Annealing is a technique that has attracted significant attention as a suitable choice for optimization problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema. At the heart of the method of simulated annealing is an analogy with thermodynamics, specifically with the way that liquids freeze and crystallize, or metals cool and anneal. The essence of the process is slow cooling, allowing ample time for redistribution of atoms as they lose mobility to ensure that a low energy state will be achieved.

The idea behind the Simulated Annealing method is to sample the Boltzmann distribution, which describes the distribution of energy of a mechanical system in a heat bath. The so-called Boltzmann probability distribution, $\text{Prob}(E) \sim \exp(-E/kT)$, expresses the idea that a system in thermal equilibrium at temperature T has its energy probabilistically distributed among all different energy states E . As a result, the system sometimes goes uphill as well as downhill. But the lower the temperature, the less likely is any significant uphill excursion.

The Simulated Annealing Optimizer in ADS is combined with a modification of the downhill simplex method. It can locate a good approximation to the global optimum of a given problem. The objective function of the optimization problem is regarded as the energy function of such a system. This amounts to replacing the single point x as a description of the system state by a simplex of $N+1$ points. The *moves* are the same as the reflections, expansions, and contractions of the simplex method. The procedure for the random changes in the configuration is: add a positive, logarithmically distributed random variable, proportional to the temperature T , to the stored function value associated with every vertex of the simplex, and subtract a similar random variable from the function value of every new point that is tried as a replacement point. This procedure always accepts a true downhill step, but sometimes accepts an uphill one. In the limit $T \rightarrow 0$, this algorithm reduces exactly to the downhill simplex method and converges to a local minimum. At a finite value of T , the simplex expands to a scale that approximates the size of the region that can be reached at this temperature, and then executes a stochastic, tumbling Brownian motion within that region, sampling new, approximately random points as it does so. The efficiency with which a region is explored is independent of its narrowness and orientation. If the temperature is reduced sufficiently slowly, it becomes highly likely that the simplex will shrink into that region containing the lowest relative minimum encountered.

Success or failure is quite often determined by the choice of *annealing schedule* to reduce the temperature *sufficiently slowly*. The choice of annealing schedule depends on the problem being solved. In the algorithm used here, the annealing schedule is chosen to decrease T to β times after m shoots for one iteration, and the process continues for n iterations. In other words, the customer can control the annealing schedule by changing the following control parameters:

- The number of Iterations which is the maximum number of lots. The temperature cools from iteration to iteration. Each iteration represents a procedure to achieve the thermal equilibrium for the current temperature. At each iteration, the temperature first decreases β times. Then the modification downhill simplex method is applied to m times (the number of shoots per iteration) to achieve the thermal equilibrium for this new temperature state. The total function evaluation (or the performance) is decided by the number of iteration and the number of shoots per iteration. The default value for the number of iteration is 25. The recommended range is 10-100.
- The initial temperature decides the system initial energy level. If the initial temperature is too low, the acceptance of the uphill is low too. As a result, the algorithm is hard to get rid of local minimum range. If the initial temperature is too high, some large increases in the objective function are accepted and some areas from the optimum are explored. As a result, the system requires more time to cool down (slower performance will be observed). The default value for the initial temperature is 0.1. The recommended range is 0.001-1000.
- The number of shoots per iteration which is the maximum number of shoots inside a lot. It decides the thermodynamics status of current temperature (or current iteration). If the number is too low, then the system can't reach the thermal equilibrium state before the system cools down to next temperature. If the number is too high, then the optimization becomes too slow. The default value for the number of shoots per iteration is 20. The recommended range is 10-5000.

In summary, simulated annealing can deal with highly nonlinear models, chaotic and noise data, and many constraints. It is a robust and general technique. Its main advantages over other local search methods are its flexibility and its ability to approach global optimality. However, since simulated annealing is a metaheuristic, a lot of choices are required to tune it for an actual problem. There is a clear trade-off between the quality of the solutions and the time required to compute them. The tailoring work required to account for different classes of constraints and to fine-tune the algorithm's parameters can be rather delicate.

Here are some suggestions on how to use the optimizer's control parameters:

- Increasing the number of shoots and the number of iterations will slow down the optimization. Adjusting the initial temperature only will not affect the performance.
- Use the default settings for most of the applications first.
- If the results are not satisfying, adjust the number of shoots per iteration first and then initial temperature, or adjust initial temperature first:
 - Increase the initial temperature if:
 - The result is one of the local optima.
 - The EF values for iterations don't change a lot, especially for several early iterations.
 - Decrease the initial temperature if:
 - You cannot find better results than the nominal case (iteration 0).
 - Increase the number of shoots if:
 - The EF values for iterations do not change a lot, especially for several early iterations.
 - Finally, adjust the number of iterations.

See Also

- [Least-Squares EF](#)
- [Simulated Annealing Algorithm Search](#)

Error-Function (EF) Formulation

The error function (EF) is the method used to evaluate how far away from the target the current iteration is. In its most general definition, the error function calculates the difference between the simulation and the specifications defined by the optimization goals.

One possible formulation of the error function is shown below in general terms.

$$EF = \sum_{all\ Goals} W_i \times |simulation_i - goal_i|^P$$

Here the error function (EF) first determines the difference between the simulation ($simulation_i$) and the goals ($goal_i$) for all of the goals ($allGoals$) that have been defined. This difference is usually called a *residual* . Each residual is then raised to a power, P , and the result is then multiplied by a weighting factor, W . The error function value is determined as the sum of all these terms. The weighting factors may have different values from one goal to another, and they are used to emphasize some optimization goals versus others by making their contribution to the error function more significant.

The example above is just one possible formulation of the error function. Numerous other possibilities exist and are explained in other sections of this document.

Each optimizer uses a different method for error function formulation. The types of error function formulations are shown in the table below.

Use of Error Function Formulation

Optimizers	Error Function Formulation
Random Optimizer, Gradient Optimizer, Quasi-Newton Optimizer, Hybrid Optimizer, Discrete Optimizer, Genetic Optimizer, Simulated Annealing Optimizer	Least-Squares EF
Minimax Optimizer	Minimax EF
Random Minimax Optimizer, Gradient Minimax Optimizer	Minimax L1 EF
Least Pth Optimizer	Least Pth EF
Random Max Optimizer	Negated Least-Squares EF

Least-Squares EF

The least-squares error-function is very popular. The residuals are squared, hence the name of this error function formulation. The generalized form is as follows:

$$EF = \sum_{all\ Goals} W_i \times |simulation_i - goal_i|^2$$

The least-squares error function is calculated by evaluating the error for each specified goal at each frequency/power point individually. The magnitudes of these errors are then squared, and the squared magnitudes are then averaged over frequency and/or power.

To help you understand the error function calculation in more generality for a measurement as a function of frequency, consider the following variable definitions.

d_i	the "ith" goal normalizing factor. Its inverse will be the "ith" goal internal weighting factor.
F_j	the set of frequency values specified by the "jth" frequency range
$R_{ij}(f)$	the "ith" frequency dependent response that is being optimized over the "jth" frequency range
g_{ij}	the "ith" goal value within the "jth" frequency range that is the optimization criterion corresponding to the R_{ij} response
W_{ij}	the "ith" goal weighting factor, within the "jth" frequency range, associated with the R_{ij} response and g_{ij} goal
$e_{ij}(f)$	the frequency dependent error contribution due to differences between R_{ij} and g_{ij} , evaluated at frequency "f."

Where

$d_i = \text{abs}(\text{Min})$	if only Min value is specified in the "ith" goal component.
$\text{abs}(\text{Max})$	if only Max value is specified in the "ith" goal component.
$(\text{abs}(\text{Max}) + \text{abs}(\text{Min}))/2.0$	if both Min and Max are specified in the "ith" goal component.
$d_i = 1.0$	if the value calculated for d_i from above equation is zero.

The error contribution $e_{ij}(f)$ are dependent on the optimization goal specification, which involves a specified relational operator (RelationOp), and the error contribution formation. The following table summarizes these conditions.

Relational operator	Goal satisfaction condition
Equal to (=)	$R_{ij}(f) = g_{ij}$
Less than (<)	$R_{ij}(f) \leq g_{ij}$
Greater than (>)	$R_{ij}(f) \geq g_{ij}$

The error contribution $e_{ij}(f)$ has different formulation for different error function formation. For the least-squared method,
 $e_{ij}(f) = (|R_{ij}(f) - g_{ij}| / d_i)^2$
 The contribution to the total error function from response $R_{ij}(f)$ over the set of frequencies in the j^{th} range is then given by:

$$E_{ij} = W_{ij} \cdot \sum_{f \in F_j} e_{ij}(f)$$

The next step is to sum the contributions from all responses within the frequency range

and divide by the number of frequencies. One way to express this mathematically is with the equation:

$$E_j = \frac{\sum_i E_{ij}}{N_j} = \frac{\sum_i E_{ij}}{\sum_i N_{ij}} = \frac{\sum_i \left[W_{ij} \cdot \sum_{f \in F_j} e_{ij}(f) \right]}{\sum_i \sum_{f \in F_j} N_{ij}(f)}$$

in which N_j is the number of frequencies in frequency range F_j .

The final step in the error function calculation is to sum over all frequency ranges.

$$E = \sum_j E_j$$

The weighting factors in different frequency ranges can be used to emphasize one of the E_j 's with respect to others, just as weights within a frequency range can be used to attach greater or lesser importance to a given response relative to other responses within that frequency range.

In summary, the error function calculation can be represented by the following triple summation:

$$E = \sum_j \frac{\left\{ \sum_f \left[\sum_i W_{ij} \cdot e_{ij}(f) \right] \right\}}{N_j}$$

Where

- The inner summation index i runs over all responses R_{ij} in frequency range F_j .
- The second or middle summation is over all frequencies in frequency range F_j .
- The outer summation index j runs over all frequency ranges.

For optimization with an additional 2nd swept variable (VAR) parameter, there is an additional summation over the parameter range. This error function formulation is represented in the following equation:

$$= \sum_j \left(\sum_f \frac{\sum_P \left[\sum_i W_{ij} \cdot e_{ij}(fP) \right]}{N_j \cdot P_i} \right)$$

Where the additional summation index p is over the swept variable levels in the swept variable range for the i th response, P_i represents the total number of swept variable levels in the i th responses power range. Thus, each response can have a unique swept variable range associated with it.

The above procedure for error function formulation is based upon the error contribution from all of the responses for j th frequency range, E_j . The total error function with respect to the error contributions from each response, E_i , can also be constructed, which includes all of the frequency range. So,

$$E_i = \frac{\sum_j E_{ij}}{N_i} = \frac{\sum_j E_{ij}}{\sum_j N_{ij}} = \frac{\sum_j \left[W_{ij} \cdot \sum_{f \in F_j} e_{ij}(f) \right]}{\sum_j \sum_{f \in F_j} N_{ij}(f)}$$

and

$$E = \sum_i E_i$$

This second approach for error function formulation is applied in this documentation. Remember that a response is any individual measurement on any network. And, that the weights W_{ij} have the value 1 unless some other value is given with the g_{ij} goal

specification.

For more information on weight, refer to [Weighting Factors](#).

Minimax EF

The Minimax optimizer calculates the difference between the desired response and the actual response over the entire measurement parameter range of optimization. Then the optimizer tries to minimize the point that constitutes the greatest difference between actual response and desired response.

Basically, minimax means *minimizing the maximum (of a set of functions generally denoted as errors)*. The error function is defined as the maximum among all error contributions, regardless of their signs or, expressed mathematically:

$$E = \max\{e_{ij}(f) * W_{ij}\}, \text{ among all of the } f, j \text{ and } i$$

where $e_{ij}(f)$ is equal to $(R_{ij}(f) - g_{ij})/d_i$ for less than goal relationship, and is equal to $(g_{ij} - R_{ij}(f))/d_i$ for greater than relationship.

The minimax objective function always represents the worst case, where the specifications are either most severely violated (in which case $E > 0$) or, are satisfied with the worst error (in which case $E < 0$). The minimax optimizer will spend all its effort trying to minimize these. A minimax solution is one such that the goal specifications are met in an optimal, typically equal-ripple manner.

Minimax L1 EF

Random Minimax and Gradient Minimax optimizers use the Minimax L1 error function:

$$E = \max\{0, e_{ij}(f) * W_{ij}\}, \text{ among all of the } f, j \text{ and } i$$

These optimizers calculate the difference between the desired response and the actual response over the entire measurement parameter range of optimization. Then the optimizers try to minimize the point that constitutes the greatest violation for the desired response. Compared with minimax error function, minimax L1 error cannot be less than zero. So that it only accounts for the most severely violated case.

Least Pth EF

The Least P^{th} optimizer uses an error function formulation similar in makeup to the least squares method found in the random, gradient, and the quasi-Newton optimizers. But, instead of squaring the magnitudes of the individual errors at each frequency, it raises them to the P^{th} power, where $p = 2, 4, 8, \text{ or } 16$. The optimizer automatically increases p in that sequence. This emphasizes the errors that have high values much more strongly than those that have small values. As p increases, the Least P^{th} error function approaches the minimax error function.

Least P^{th} allows the error function to become negative. That happens when you specify a performance window and the response moves inside that window.

For example, there may be a minimum and maximum gain specification on an amplifier and the Least P^{th} optimizer can go beyond the specification and place the gain halfway between the two limits.

The Least P^{th} optimization routine is the exponential sum of the error function, where the exponent p is not necessarily equal to 2. It can be a positive number, usually an integer.

First of all, the maximum error is found as:

$$E_{\max} = \max_i (E_i)$$

According to the sign of E_{\max} , the Least P^{th} error function is defined as follows:

$$E = \left(\sum_{i, E_i > 0} (E_i)^P \right)^{1/P}$$

if $E_{\max} > 0$

$$E = 0$$

if $E_{\max} = 0$

$$E = \left(\sum_i (-E_i)^{-P} \right)^{-1/P}$$

if $E_{\max} < 0$

The cases $E_{\max} > 0$ and $E_{\max} < 0$ are clearly distinguished. The first case, where $E_{\max} > 0$, only positive errors enter the final error function. For the second case, where $E_{\max} < 0$, all errors become part of the final error function. In this case, all errors are negative, i.e., all the specifications are satisfied.

The Least P^{th} final error function is well-defined whether or not there are positive errors. It has the same sign as the maximum error E_{\max} and becomes negative when all the specifications are met. Therefore, the optimizer continues to improve the performance of the design, even after there are no more violations of the specifications.

The Least P^{th} formulation is used as an indirect method to achieve a minimax design. Minimax error function can contain *edges* or discontinuities in their derivatives. These occur at points where the error contributions due to different goals intersect in the parameter space. The Least P^{th} error functions avoid this problem.

For a large value of p , the errors having the maximum value ($E_i = E_{\max}$) are more strongly emphasized over the other errors, i.e., they are given higher priority in optimization. As p increases to infinity, the Least P^{th} formulation leads to a minimax error function. The problem is solved through a sequence of Least P^{th} optimizations with p being gradually increased. The sequential Least P^{th} optimization used in the program uses $p = 2 \ 4 \ 8 \ 16$. This strategy often provides a smooth path towards a minimax solution.

Negated Least-Squares EF

The optimizer that provides worst-case analysis is called *Random Max*. The random maximizer internally negates (changes the sign) the least-squares error function so that the effect is a *maximization* of the error function.

Note

The effect of this error function is to drive the values to the extreme of their ranges. To prevent destroying the originally desired response, you may want to save a copy of your optimized design, then change your optimization constraints to be equal to the tolerances on these variables.

For more information on the least-squares error function, refer back to [Least-Squares EF](#), keeping in mind the effect of the negation.

Weighting Factors

All of the error-function (EF) formulations include weighting factors, W_i . Weighting factors are a measure of the importance of a given goal relative to other goals. The default value for weighting factors is 1.0.

Setting the Appropriate Weighting Factors Manually

Assuming the internal weighting factors are 1.0 (default value), there can be two cases for setting up appropriate weighting factors related to different requirements:

- Unbiased Error Function
- Indented Emphasis

Unbiased Error Function

An unbiased error function means the partial errors for each of the goals have almost the same contributions to the error function. When the partial errors have values of the same order, there is no problem with the default weighting factors. However, when the partial errors have values of different orders, you must adjust the weighting factors to make the

weighted partial errors in the same order. For example, assume there are three requirements for one amplifier design:

$$|S_{22}| \leq 0.30$$

$$\text{Gain ripple} \leq 1 \text{ dB}$$

$$|NF_{ACT} - NF_{Min}| \leq 0.1 \text{ dB}$$

If the least-squares error function is applied to the optimization procedure, the orders for the partial errors for the three goals are 0.09, 1, and 0.01 respectively, and the order for the total error is:

$$EF = W_1 * 0.09 + W_2 * 1 + W_3 * 0.01$$

If W_1 , W_2 , and W_3 use the default value of 1.0, it is clear that the error function will be biased toward the second goal, while the third goal requirement will be almost ignored. In order to use the three goal requirements equally, the appropriate weighting factors can be used in the following order:

$$W_1 = 1.0/0.09$$

$$W_2 = 1.0/1$$

$$W_3 = 1.0/0.01$$

In other words, the appropriate weighting factor depends on your specific performance requirement and you must set them correctly to obtain valid results.

Intended Emphasis

Sometimes, it is desirable to allow the error function emphasis on some of the goals. In this case, larger weighting factors for these goals means a higher emphasis.

Setting the appropriate weighting factors based on the goal requirements as described above can become tedious. One method is provided with the effect of having the appropriate internal weighting factors defined for each of the performances automatically as shown in the following table.

Internal Weighting Factors Based on Performance Specifications

IW_i =	abs(Min)	if only Min value is specified for <i>ith</i> goal (in the <i>ith</i> goal component)
	abs(Max)	if only Max value is specified for the <i>ith</i> goal (in the <i>ith</i> goal component)
	(abs(Max) + abs(Min))/2.0	if both Min and Max are specified for the <i>ith</i> goal (in the <i>ith</i> goal component)
IW_i =	1.0	if the value calculated for IW_i from the above equation is zero

The procedure used to set up the appropriate weighting factors using internal weighting factors from the automatic scaling mode is:

- For the *Unbiased Error Function*, use the default values (1.0) for all of the weighting factors. If all of the min and/or max specifications for the goals are zero, which means all of the internal weighting factors are 1.0, the weighting factors are needed to set up appropriate values following the manual mode procedure above.
- For the *Biased Error Function*, adjust the weighting factors if you want to emphasize any one of the goals.

Search Methods

These search methods are used to arrive at new parameter values:

- [Random Search](#)
- [Gradient Search](#)
- [Quasi-Newton Search](#)
- [Gauss-Newton/Quasi-Newton \(minimax\) Search](#)
- [Exhaustive Search](#)
- [Genetic Algorithm Search](#)
- [Simulated Annealing Algorithm Search](#)

Optimization with Random Search is typically used initially. Optimization with Gradient search is generally used in later stages of optimization. Discrete optimization only affects discrete-valued variables. The genetic algorithm search is well suited to the discrete and mixed (continuous and discrete) problems.

Random Search

The optimizers using random search method (Random, Random Minimax, and Random Max optimizers) arrive at new parameter values by using a random-number generator, that is, by picking a number at random within a range, which is sometimes a slower process compared to the optimizers using gradient search methods.

Optimization with random search method is a trial and error process. Starting from an initial set of parameter values for which the error function is known, a new set of values is obtained by perturbing each of the initial values, and the error function is re-evaluated.

For optimization with random search method, a trial consists of two error function evaluations. A trial performed by optimization with random search method is completed by reversing the algebraic sign of each parameter value perturbation and re-evaluating the error function. These two values, corresponding to positive and negative perturbations, are compared to the value at the initial point.

If either value is less than the initial value, then the set of parameter values for which the error function has its least value becomes the initial point for the next trial. If neither value is less than the initial value, then the initial point remains the same for the next trial.

Gradient Search

The optimizers using gradient search method (Gradient and Gradient Minimax optimizers) find the gradient of the network's error function. These optimizers usually progress more quickly to a point where the error function is minimized, though it is possible for them to terminate in a local minimum.

The optimizers find the gradient of the error function (i.e., the direction to move a set of parameter values in order to reduce the error function). Once the direction is determined, the set of parameter values is moved in that direction until the error function is minimized. Then the gradient is re-evaluated. This cycle is equal to one iteration of the gradient optimizers.

A design that is optimized by a gradient optimizer has the least sensitivity (more stable) to slight variations in its parameter values.

A single iteration usually includes many function evaluations ; therefore, an iteration in optimization using gradient search method takes much longer than a trial in optimization using random search method.

Quasi-Newton Search

The optimizers using Quasi-Newton search method (Quasi-Newton and Least Pth optimizers) use second-order derivatives of the error function and the gradient to find a descending direction.

The optimization routine using Quasi-Newton search method estimates the second-order derivatives using the Davidson-Fletcher-Powell (DFP) formula or its complement. Appropriately combined with the gradient, this information is used to find a direction and an inexact line-search is conducted.

The optimization terminates when the gradient vanishes or the change ration in the variables is small (less than $1.0e-5$). It also stops when the number of iterations, that you have specified, is reached. The bounds imposed on the optimization variables are handled using a transformation of variables.

Like the optimizers using gradient search method, an iteration in the optimizers using Quasi-Newton search methods consists of many function evaluations, and takes longer than a trial in the optimizers using random search method.

Gauss-Newton/Quasi-Newton (minimax) Search

The minimax optimizer consists of two stages. In the first stage of the algorithm, the optimizer solves a minimax problem using a linear programming technique. In doing so, the status and potential of each individual error function component are analyzed. Its contribution to the minimax problem is mathematically assessed and taken into account during optimization.

In the second stage, the optimizer works with a Quasi-Newton method using approximate second-order derivatives. Such extra effort becomes necessary for an accurate and

efficient solution to certain ill-conditioned problems (i.e., singular problems).

The minimax optimizer terminates when responses become optimally equal-ripple or the relative change in the variables is less than 0.05 percent. It also stops when the number of iterations is reached.

Note that the bounds imposed on the variables are formulated and treated directly as linear constraints without having to resort to variable transformation; therefore, a source of nonlinearity is eliminated.

Exhaustive Search

The exhaustive search method is used exclusively by the discrete optimizer. The discrete optimizer only affects parameter values specified as discrete-valued optimization variables.

The exhaustive search method involves a comprehensive search for the combination of discrete values that results in the best design performance. Starting from an initial set of parameter values for which the error function is known, an update in the parameter values occurs upon an improvement in the error function.

Because the parameter values that may change are not continuous variables, this search method is more a series of trials than iterations. Moreover, the number of trials required to attempt all combinations may often be prohibitive.

To reduce optimization time, keep the number of discrete variables to a minimum and reduce the number of values the discrete variables may take on.

Genetic Algorithm Search

The Genetic optimizer is the only optimizer that uses the Genetic Algorithm search method.

For more information, refer to the [Genetic Optimizer](#).

Simulated Annealing Algorithm Search

The Simulated Annealing optimizer is the only optimizer that uses the Simulated Annealing Search method. Basically, it is a modified downhill simplex search mechanism. For more information, see [Simulated Annealing Optimizer](#).

Sensitivity Analysis

Sensitivity analysis is listed as a type of optimization. This feature comprises a single-point or infinitesimal sensitivity analysis of a design variable. Sensitivity analysis for circuit design involves taking partial derivatives of the response with respect to a design variable of interest. It is thought that these numbers can help pinpoint variables that contribute disproportionately to performance variance.

The method used to compute sensitivities is based on finite difference approximation requiring N+1 full circuit simulations, where N is the number of design variables. Sensitivity analysis results are also unconditionally sent to the dataset, and this data can be examined in the Data Display window. Sensitivities are approximated as follows:

$$S_{P_i} = \frac{\partial R(P)}{\partial P_i} \approx \frac{R(P^0) - R(P_i^+)}{P_i^0 - P_i^+}$$

where $R(P^0)$ is the response evaluated at the nominal point and $R(P_i^+)$ is the response due to a perturbation in the i^{th} parameter. The perturbation ratio is about $1.0\text{e-}06$ (e.g., $P_i^+ = (1+1.0\text{e-}06)*P_i^0$). The response R is actually the expression found in the goal(s) given in the Optimization window and performing the sensitivity analysis.

The sensitivity analysis also outputs normalized sensitivities. Normalized sensitivities use the approximate gradient (single-point sensitivity) to predict the percentage change in the response due to a 1% change in the design variable. Normalized sensitivity is defined as:

$$SN_{P_i} = S_{P_i} \frac{|P_i^0|}{|R(P^0)|} = \frac{R(P^0) - R(P_i^+)}{\frac{P_i^0 - P_i^+}{|P_i^0|}}$$

The sensitivity information is saved in the dataset using the Goal name. Normalized sensitivities are save with the name: norm_ <goal_name> .

Cascaded Optimization Strategies

There is no unique *winning* strategy for optimization due to the complexity and variation of circuit design. However, the starting point is very important in finding a good solution to your design problems. It is recommended that you start from a manual/empirical estimation, based on a simplified design technique. Starting with some empirical estimation or *educated guess* is always better than simply picking a random starting point. Once you start using the optimization process, you may develop your own strategies based on different applications. Some possible strategies are,

- gradient -> random -> gradient
- random -> gradient
- random -> gradient -> discrete
- random -> gradient -> random with discrete variables

For example, you can use the random optimizer first, then fine-tune the solution using the gradient optimizer. Using this strategy, the min-max ranges for a discrete optimization can be significantly narrowed. The discrete optimizer can then be used if necessary.

In addition to reducing the number of possible discrete values for each discrete variable, you should also consider minimizing the number of discrete variables to be used. The intent here is to minimize the total number of permutations which, in turn, will minimize the optimization time and the computer resources required for this process.

For more information on discrete variables, refer to *Value Types for Nominal Optimization* (optstat).

Optimization Examples

This section includes multiple examples that are intended to help with your understanding of various optimization topics. The examples provided in this section include the following topics:

- [Single Optimization Example](#)
- [Final Analysis Example](#)
- [Swept Optimization Example](#)
- [Programmable Optimization Example](#)
- [Single Optimization Cockpit Example](#)
- [Swept Optimization Cockpit Example](#)

See also *Locating Optimization Examples* (optstat).

Single Optimization Example

Example 1: Continuous Optimization

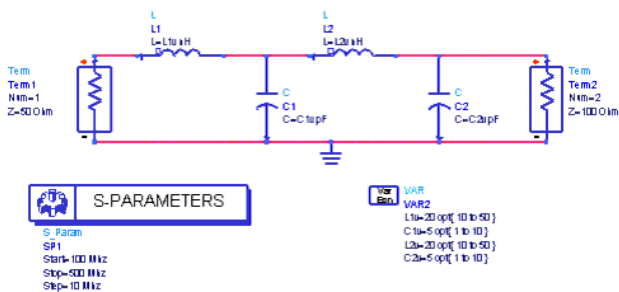
The example used in this topic is from Analog/RF Systems simulation. However, the Nominal Optimization procedure is the same for either Signal Processing or Analog/RF Systems type of simulation. For a bit-width example used in ADS Ptolemy simulation, refer to *Using Nominal Optimization* (ptolemy).

This example is called *optex1_wrk*, and it is located in the directory `$HPEESOF_DIR/examples/Tutorial`. To access this example workspace and enable simulation, open the example by choosing **File > Open > Example** from the ADS Main window.

The circuit topology for the following simple example represents a 2-to-1 impedance matching transformer with a passband of one octave. The example circuit is intended to match the 100-ohm load resistance of R1 to 50 ohms over the range of 200 to 400 MHz. The circuit is swept from 100 to 500 MHz to view the out-of-band response as well as the passband response.

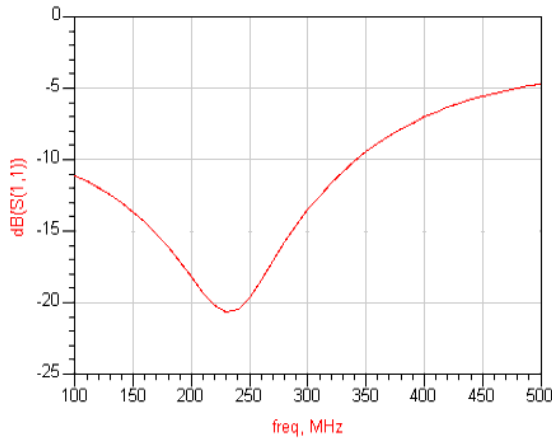
Of the next two following figures, the first one shows the beginning design with the initial component values used for this example. The display shown in the second figure demonstrates that the initial response of the circuit is far from optimum.

Design Used for Optimization Example

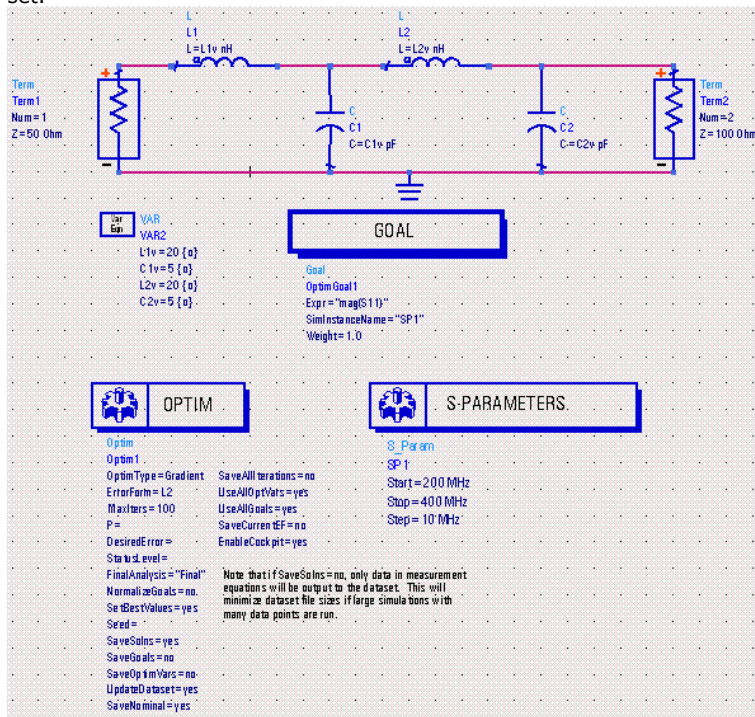


Initial Response of Example Circuit

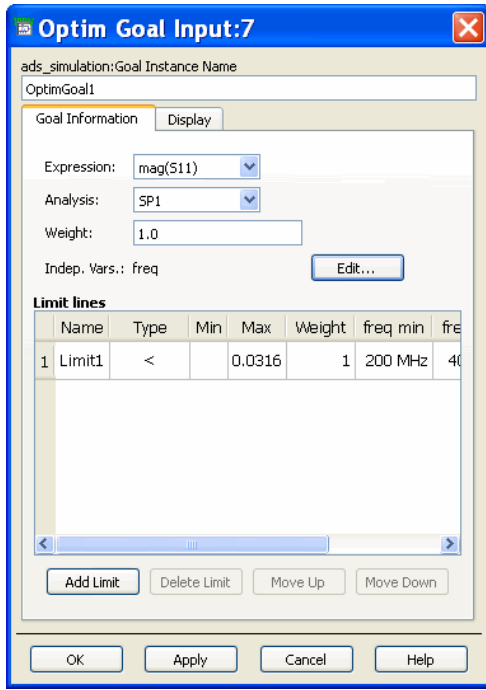
This initial simulation shows that the input reflection is too high over the 200-400 MHz range.



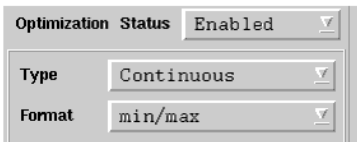
1. Set up Optimization Goals. Since the intent of this design is to match the 100-ohm load resistance to 50 ohms, and we are measuring the input reflection coefficient S11, the goal is to make S11 as small as possible. A Nominal Optimization component and a Goal component are placed from the *Optim/Stat/DOE* palette into the design, as shown in the following figure, and the appropriate specifications are set.



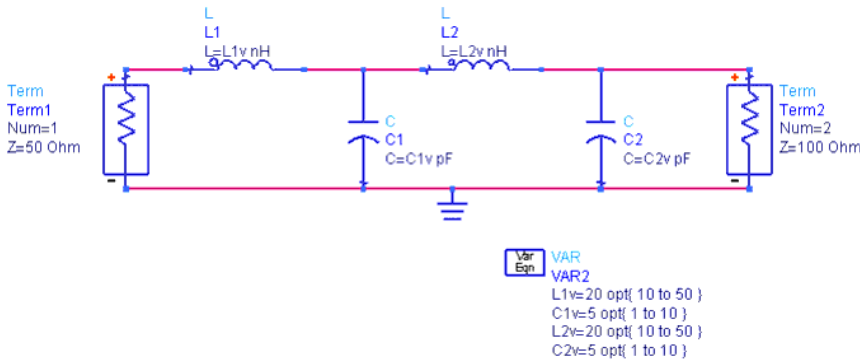
Notice that in the Goal component, there is only one limit line. The limit line is specified as "mag(S11) < 0", as shown in the following figure. This dialog is accessed by double-clicking on the placed Goal component as shown in the figure above.



2. Set up Optimization Variables/Parameters. The next step is to specify which circuit parameters are to be adjusted by the optimizer. For this example, all four parameters are optimized. These include the inductors L1 and L2, and the capacitors C1 and C2. Suppose the smallest inductance we can achieve in practice is 10 nH and the largest is 50 nH. Suppose the smallest capacitance we can achieve is 1 pF, and the largest is 10 pF. In the Setup dialog box for each of these four components, we specify a constrained optimization Value Type (specifically *min/max*), as shown in the following figure. Refer to the section *Specifying Component Parameters (optstat)* for more details.

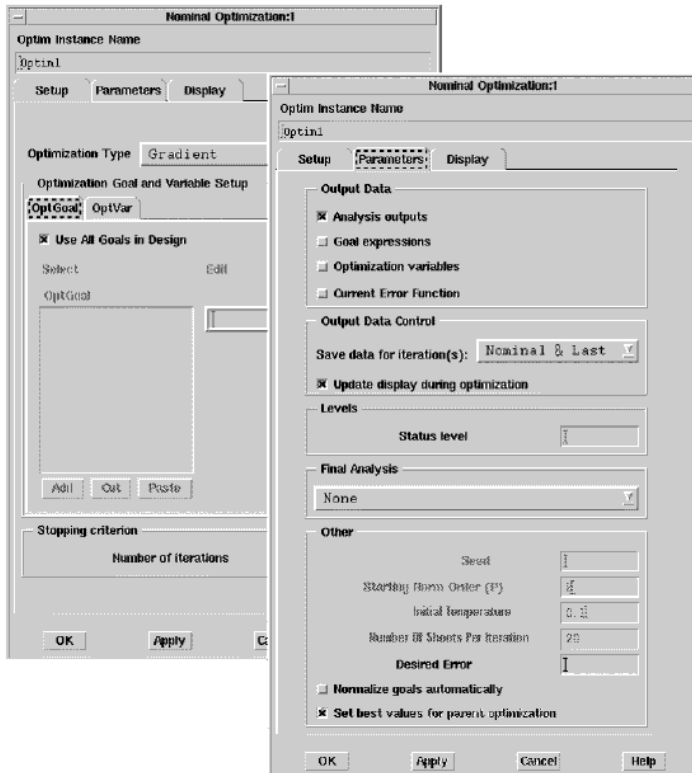


This allows the circuit parameter to be adjusted within the constraints as shown in the following figure.



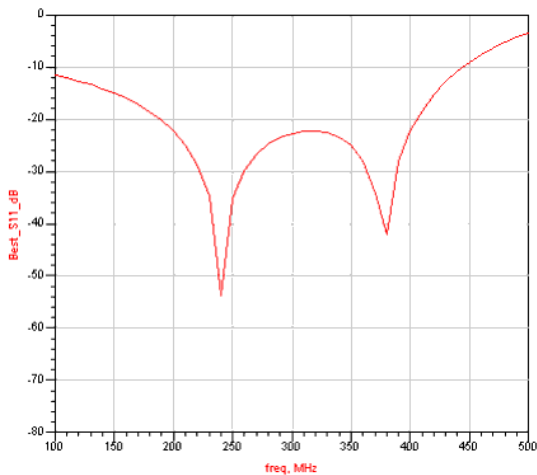
Note
 ADS support different formats for specifying optimization variables. The most common format for continuous optimization variables are Min/Max format. For more information on the optimization variable format, refer to *Understanding Optimization Variable Types (optstat)*.

3. Set up Optimization Methodology. Now the circuit can be optimized. It turns out that the least-squares error function (explained in the section, *Error-Function (EF) Formulation (optstat)*) is relatively smooth, so the Gradient optimizer is a good initial choice. In the optimization controller, the *Gradient* optimizer is selected, *Use All Goals in Design, Use All Optimization Variables in Design*, only output *Analysis outputs for Nominal & Last iteration(s)*, and *Update display during optimization*, as shown in the following figure.



4. Launch Optimization. To optimize this setup, choose **Simulate > Optimize**, or the Optimize icon on the toolbar.
5. The following figure shows the results of selecting 100 iterations of the Gradient optimizer on this circuit. The optimizer terminated due to a zero gradient, or a local minimum, in the error function after 13 iterations.

Results After 13 Iterations Using Gradient Optimizer

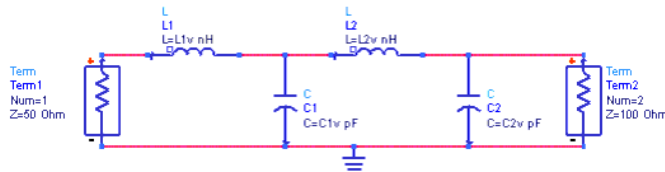


Notice that the response, although much improved over the initial response, is still not quite optimal. The return loss (S_{11} in dB) is less than 20 dB at 200 MHz (the low-frequency end of the passband), about 23 dB at 400 MHz (the high-frequency end of the passband), and about 26 dB near the band center. An optimal design would have equal values of return loss at these three frequencies.

Note
 ADS names optimization variables in a specific way to avoid confusion in the dataset or Status window, as follows:
`<enclosing_definition_name> . <instance_name> . variable_name.`
 For more information on this naming convention, refer to *Understanding How ADS Names Optimization Variables* (optstat).

6. The next step is to choose **Simulate > Update Optimization Values** to see the values that the Gradient optimizer has arrived at, as shown in the following figure.

Schematic Showing Optimized Parameter Values



S-PARAMETERS

S Param
SP1
Start=100 Mhz
Stop=500 Mhz
Step=10 Mhz

VAR

L1v=2.270670e+01 opt{ 10 to 50 }
C1v=8.712450e+00 opt{ 1 to 10 }
L2v=4.356221e+01 opt{ 10 to 50 }
C2v=4.541352e+00 opt{ 1 to 10 }

OPTIM

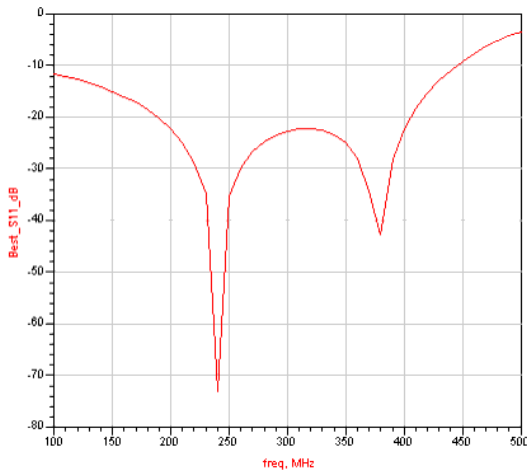
Optim
Optim1
OptimType=Minimax
ErrorForm=MM
MaxIter=100
P=.
DesiredError=.
StatusLevel=.
FinalAnalysis="None"
NormalizeGoals=no
SetBestValues=yes
Seeds=.
SaveSols=yes
SaveGoals=no
SaveOptimVars=no
UpdateDataset=yes
SaveNominals=yes

GOAL

Goal
OptimGoal1
Expr="mag(S11)"
SimInstanceName="SP1"
Mns=.
Max=0
Wlght=1
RangeVar[1]="freq"
RangeMin[1]=200 Mhz
RangeMax[1]=400 Mhz

7. Although the design shown in the previous figure is not optimal, it does provide a good starting point for further optimization. An optimal response would have equal values of return loss at the band edges and at the band center (an equal-ripple response). The Minimax optimizer, using a method described in the section *Error-Function (EF) Formulation* (optstat) is designed to achieve such a response. The following figure shows the results of selecting 100 iterations of the Minimax optimizer on this circuit. The optimizer terminated after 8 iterations.

Results Using Minimax Optimizer



8. Notice in the previous figure that the values of the return loss at the band edges and at the band center are now equal: about 22.5 dB. This is the equal-ripple response, which is optimal for the design. Selecting **Simulate > Update Optimization Values** causes the design to be updated with the optimal design values in the Var/Eqn component, as shown in the following figure.

Var Eqn

VAR VAR2

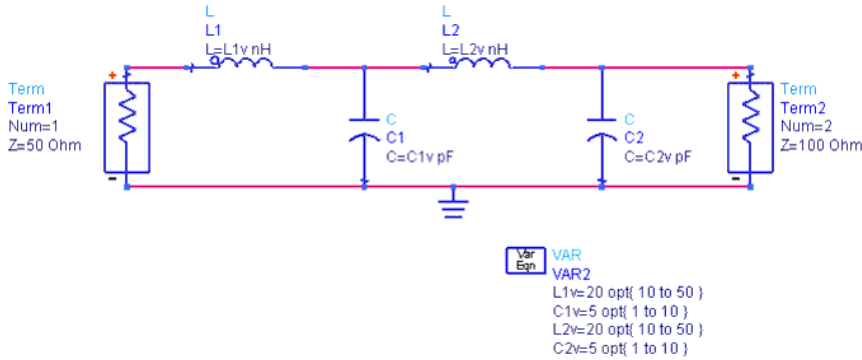
L1v=2.270670e+01 opt{ 10 to 50 }
C1v=8.712450e+00 opt{ 1 to 10 }
L2v=4.356221e+01 opt{ 10 to 50 }
C2v=4.541352e+00 opt{ 1 to 10 }

Example 2: Discrete Optimization

The previous nominal optimization example was based on continuous-value optimization. Advanced Design System also allows you to perform nominal optimization in which the results are limited to real-world, discrete-value parts. Discrete optimization variables can be defined using Vendor Component Libraries (for example, the RF Passive SMT Library), reading from files or the Min/Max/Step format. Discrete optimization can be compatible with the Random, Random Minimax, Random Max, Discrete, and Genetic optimizers.

This example uses the same workspace, *optex1_wrk*, as the previous example; however, the example is modified for discrete optimization. Most of the discrete optimization procedure is the same as other types of nominal optimization, so we will only describe the differences here.

Impedance Matching Transformer Before Discrete Optimization



There are several methods to set up discrete optimization. The *optex1_wrk* example is changed here to learn how to do two of them by:

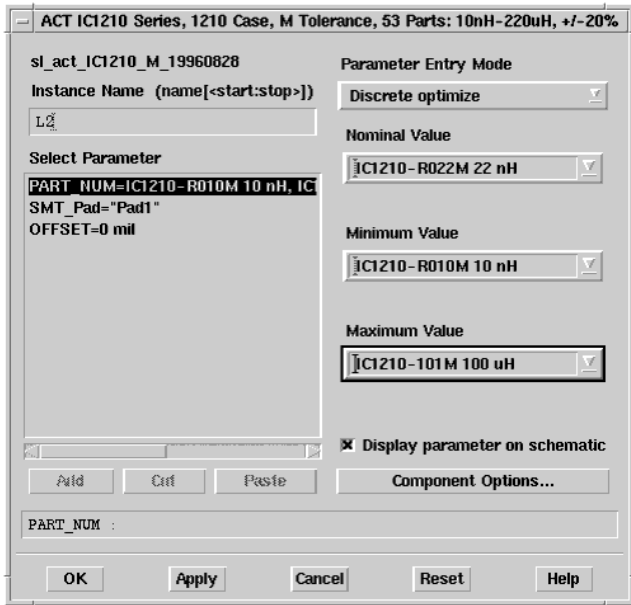
- Replacing inductor L2 with an SMT inductor from the RF Passive SMT Library and setting it up for discrete optimization. This step illustrates the use of Vendor Component Libraries with discrete optimization. Here values are based on the manufacturer's standard values.
- Changing capacitor C2v in the VAR component to be used for discrete optimization. This step illustrates the use of a component referenced in a VAR with discrete optimization. Here you specify the range and step size.
- Changing the Nominal Optimization component from *Gradient* to *Random*. Discrete optimization is compatible *only* with the Random, Random Minimax, Random Max, Discrete, and Genetic optimization types.

A third method uses a user-defined data file to set up discrete optimization and is described at the end of this section. Refer to [Performing Discrete Optimization Using a Data File](#) for details.

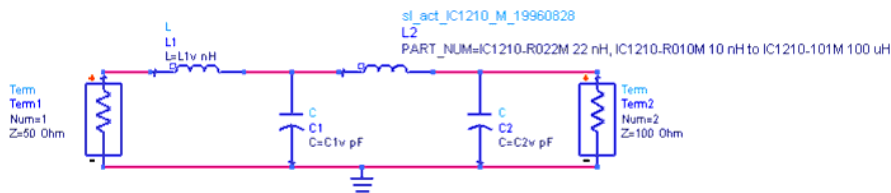
Setting Up Inductor L2 for Discrete Optimization

To set up an SMT inductor for discrete optimization:

1. Delete inductor L2.
2. Choose **Insert > Component > Component Library**. The Component Library window is displayed.
3. From the Libraries list, select the **RF Passive SMT Library**. Then select **SMT Inductor** (a sub-type under RF Passive SMT Library).
4. From the Components list on the right, select part **sl_act_IC 1210_M_19960828**.
5. Place the new part and wire it in to the design.
6. Double-click the inductor to bring up the Component Parameter dialog box.
7. Choose the **Parameter Entry Mode** drop-down list button and select **Discrete optimize** from the list.
8. Three fields appear under Discrete optimize for Nominal, Minimum, and Maximum values. Each field has a drop-down list with the inductor series associated with the inductor family we placed. Select:
 - The **22 nH** part for the Nominal Value
 - The **10 nH** part for the Minimum Value
 - The **100 uH** part for the Maximum Value
9. After selecting these values, the dialog box appears as shown below. Click **OK**.



At this point, the design appears as below. Note that the annotation has changed for the inductor, to reflect the parts to be used for discrete optimization.



Setting Up Capacitor C2v for Discrete Optimization

To set up capacitor C2v in the VAR component for discrete optimization:

1. Double-click the **VAR** component to bring up the Component Parameter dialog box.
2. Select **C2v** from the Select Parameter list.
3. Click the **Tune/Opt/Stat/DOE Setup** button. A dialog box appears.
4. Under the Optimization tab, change **Continuous** to **Discrete** in the Type field.
5. You then enter the Nominal, Minimum, Maximum, and Step values you want for discrete optimization. For this example, leave the defaults for the first three and enter **1** in the Step Value field.
6. Click **OK** to return to the previous dialog box. Click **OK** again.

The VAR component is updated to show the discrete optimization parameters, as shown below.

```

Var  VAR
Eqn  VAR2
     L1v=20 opt{ 10 to 50 }
     C1v=5 opt{ 1 to 10 }
     L2v=20 opt{ 10 to 50 }
     C2v=5 opt{ discrete 1 to 10 by 1 }
    
```

Setting Up the Nominal Optimization Component

To set up the Nominal Optimization component for discrete optimization:

1. Double-click the *Optim* component to bring up the Nominal Optimization dialog box.
2. In the *Optimization Type* drop-down list, change **Gradient** to **Random**.
3. Click **OK**.

This example is now ready for discrete optimization.

Optimization Results

After the design is optimized, the inductor will be updated for a value to 39 nH, based on the goal criteria that was specified. The schematic will be updated only if you select *Simulate > Update Optimization Values* from the Schematic window.

Performing Discrete Optimization Using a Data File

A third method to set up discrete optimization employs a user-defined data file that contains a list of the discrete values. This file is referenced by placing a *Data Access Component (DAC)* on your schematic. This method is desirable when you have a long list of values for a part that is not found in the Advanced Design System Vendor Component Libraries.

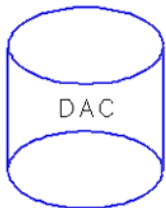
To specify parameters for discrete optimization using a data file:

1. Create your data file with an editor or program suited for this task.
 2. Place your data file in the data subdirectory of the desired workspace. (Other directories are allowed but their paths must be specified.)
- It is important for you to understand the DAC data file format when building discrete data files. The DAC data file consists of a matrix of data arranged in rows and columns. Each row represents a different possible component value or part number. Each column represents a different parameter or characteristic of your component. At the top of each column is a name used to identify that column in the file. These column names may be any alphanumeric string defined by the user. In this example, the file is a simple file-based list of discrete-value parts for inductors and capacitors as shown in the following table.

List of Discrete-Value Parts

```
BEGIN DSCRDATA
% rownumber Inductance Capacitance
0          1.9          0.5
1          2.7          0.75
2          3.3          1.0
3          4.7          1.2
4          5.6          1.5
5          6.8          2.2
6          7.5          3.3
7          8.2          3.9
8          9.5          4.7
9          10           5.6
10         12           6.8
11         15           7.0
12         19           7.5
13         22           8.2
14         24           9.1
15         27           10
16         30           12
17         33           15
18         36           18
19         39           20
20         40           22
21         43           27
22         47           33
23         51           39
END
```

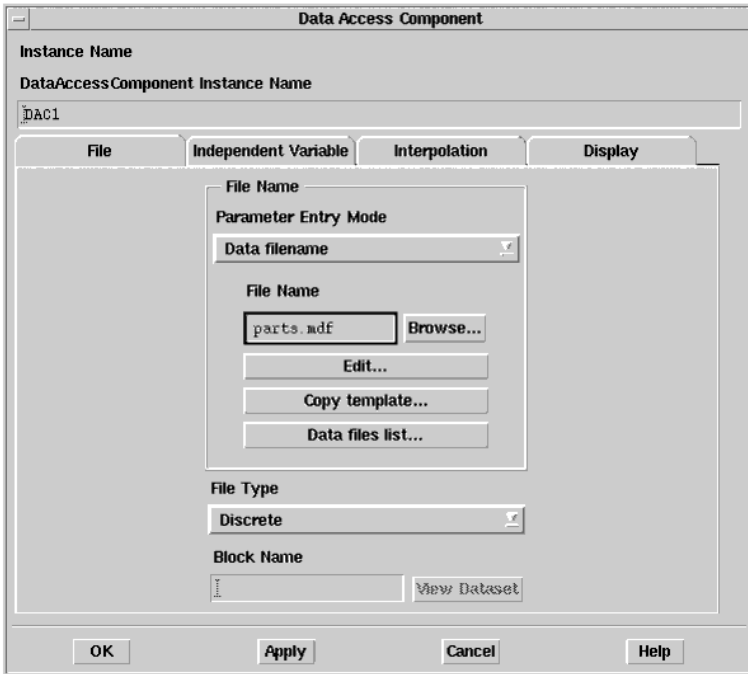
Place a DAC component in the Schematic window by choosing **Insert > Component > Component Library > Data Items > DataAccessComponent** (or choose a DAC from the *Data Items* palette) and placing it in your design. An example is shown below.



```
DataAccessComponent
DAC1
File="stdLvals.mdf"
iVar1=1
iVal1=Index
```


Double-click the component to bring up the Component Parameter dialog box.

In the Select Parameter list on the left, set the **File** parameter to refer to your data file. In this example, the file is called *parts.mdf*, as shown below.



There are a number of other parameters in the Data Access Component dialog box. Only a few are needed for this example.

Under the *File* tab, accept the default for the File Type parameter, which is *Discrete*.

Under the *Interpolation* tab, accept the defaults for Interpolation Method (Index Lookup) and Interpolation Domain (Rectangular).

Under the *Display* tab, there are pairs of parameters, called *iVar1* and *iVal1*, *iVar2* and *iVal2*, *iVar3* and *iVal3*, etc. These indicate independent variable 1 and independent value 1, independent variable 2 and independent value 2, etc. Each pair tells the software to refer to a row number in your data file for each variable. In this example, the only variable is inductance, so only *iVar1* and *iVal1* need to be specified.

Under the *Independent Variable* tab, set *iVar1* to **1** (for the first independent variable).

Then set *iVal1* to **L1_index** (for inductance index; this is a user-defined label).

Click **OK** to dismiss the dialog box.

In the Schematic window, place or edit your discrete-valued component. For this example, inductor L1 from the previous example is used. Also, as in the previous example, the VAR component is referenced to specify the parameter, as shown below.



Double-click the VAR component to edit its parameters.

For parameter L1v:

- Choose **File Based** in the Variable or Equation Entry Mode list box.
- Enter **L** in the Name field.
- Accept **DAC1** for the Data Access Component Id.
- Enter *Inductance* in the Dependent Parameter Name field.

For parameter L1_index:

- This is the parameter (inductance) that will be set up for discrete optimization.
- Choose **Standard** in the Variable or Equation Entry Mode list box.
- Enter **L1_index** in the Name field.
- The Variable Value field will be used for the nominal value set in the next step. (It can be ignored for now.)

Now click the **Tune/Opt/Stat/DOE Setup** button. The Setup dialog box appears.

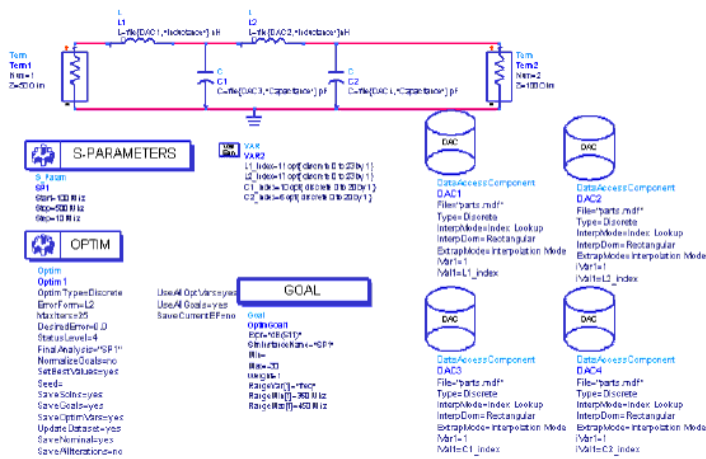
This dialog box is set up like any other for discrete optimization.

- The Optimization Status should be **Enabled**.
- The Type field should be **Discrete**.
- For this example, set the Nominal Value to **0**, the Minimum Value to **0**, the Maximum Value to **23**, and the Step Value to **1**.

- What this means for a file-based discrete optimization is that inductor values from row 0 to row 23 will be used and the step will be one row. (If the Step Value were set to 2, the simulator would skip every other row.)
- Click **OK** to dismiss the Setup dialog box.

Click **OK** again to complete the setup of the Variables and Equations parameter dialog box for L1. Follow the same procedure for L2, C1, and C2. At this point the Schematic window will appear as shown in the following figure.

Discrete Optimization using Data Files



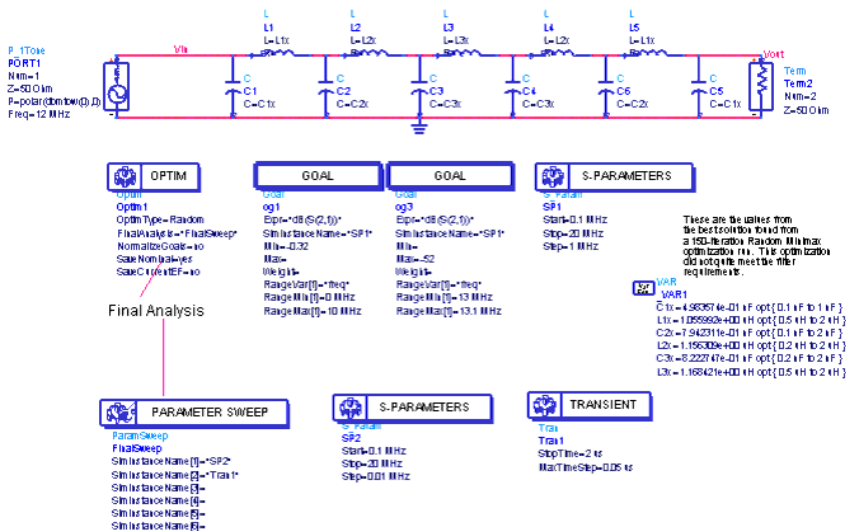
In this example, the discrete optimizer was selected. The discrete optimizer will search all of the possible combinations of the discrete design parameters. Therefore, the Number of Iterations does not apply to discrete optimizers and it takes a great deal of time if the possible combinations are large. At this point, you can start to run the discrete optimization and find its optimal design as L1_index = 11, L2_index = 17, C1_index = 10 and C2_index = 6.

Final Analysis Example

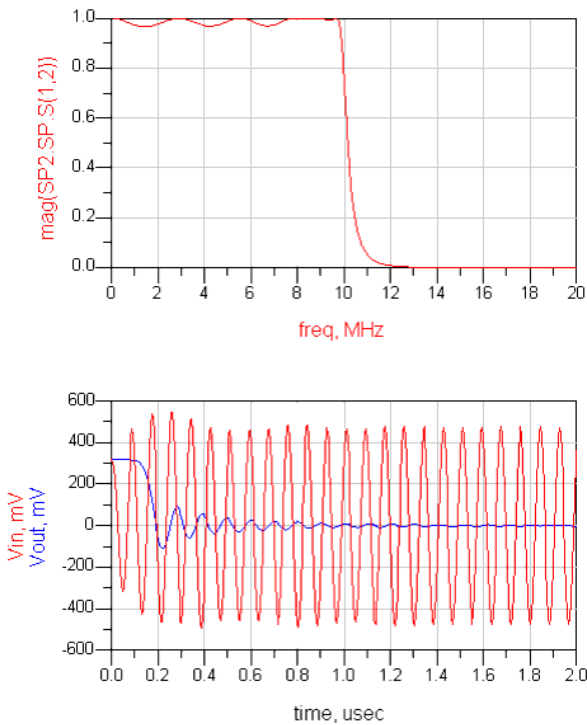
The example, \$HPEESOF_DIR/examples/Tutorial/FinalAnalysis_wrk, illustrates how to use this feature. In the following figure, the final analysis includes a finer SP analysis and a transient analysis, whose results are automatically output to the dataset and shown in the next figure.

Final Analysis Example

10 MHz Low pass filter optimization with Multiple Final Analysis.



Final Analysis Results



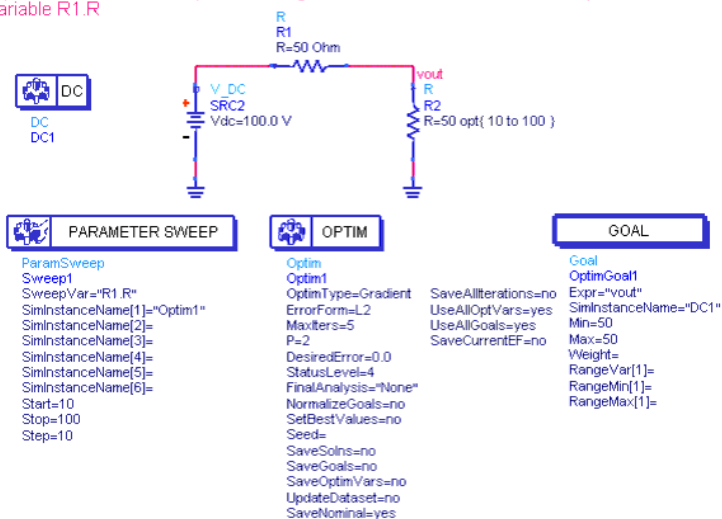
Swept Optimization Example

The following simple example is from `$HPPEESOF_DIR/examples/Tutorial/sweptOptTest_wrk` and it will help illustrate swept optimization.

The goal of this design is to optimize R2.R to form a perfect voltage divider for each value of the swept variable R1.R as shown in the following figure.

Swept Optimization Example

Optimize R2.R to form a perfect voltage divider for each value of the swept variable R1.R



In this simulation, note that the `SetBestValues` parameter of the Optim component is set

to *no*. With this setup, the optimization at each sweep point resets the value of the optimization variables (R2.R) to the initial nominal value. So after simulation, the results appear in the Data Display window as shown below:

opttiter		OPTIM.R2.R
R1.R=10.000	0	50.000
	1	10.000
R1.R=20.000	0	50.000
	1	20.000
R1.R=30.000	0	50.000
	1	30.000
R1.R=40.000	0	50.000
	1	40.000
R1.R=50.000	0	50.000
	1	50.000
R1.R=60.000	0	50.000
	1	60.000
R1.R=70.000	0	50.000
	1	70.000
R1.R=80.000	0	50.000
	1	80.000

Now set the control parameter *SetBestValues* of the Optim component to *yes*. To do this you can either edit the schematic directly or,

1. Double click the Optim controller in the schematic window. The Nominal Optimization dialog box appears.
2. Select the Parameters Tab.
3. Activate the *Set best values for parent optimization* check box.
4. Click **OK** in the Nominal Optimization dialog box to clear the dialog.

With this setup, the optimization at each sweep point will start with the optimal values of the optimization variables (R2.R) from previous optimization. So after simulation, the results appear as follows:

opttiter		OPTIM.R2.R
R1.R=10.000	0	50.000
	1	10.000
R1.R=20.000	0	10.000
	1	20.000
R1.R=30.000	0	20.000
	1	30.000
R1.R=40.000	0	30.000
	1	40.000
R1.R=50.000	0	40.000
	1	50.000
R1.R=60.000	0	50.000
	1	60.000
R1.R=70.000	0	60.000
	1	70.000
R1.R=80.000	0	70.000
	1	80.000

Notice above how the optimization variable R2.R tracks the swept variable R1.R to form a voltage divider for each sweep point for both cases.

Programmable Optimization Example

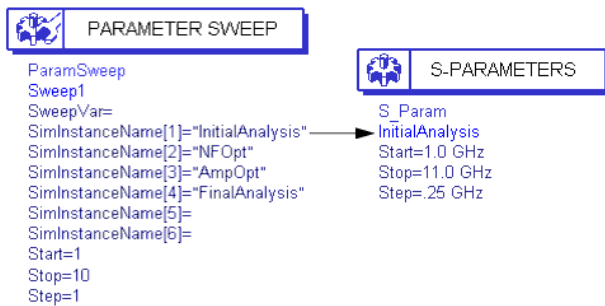
A practical application of programmable optimization for an MMIC design can be found in:

\$HPEESOF_DIR/examples/MW_Ckts/Design_Manufacturing_MMIC_wrk

Save a copy of the workspace and open the *A3_X-band_LNA_prog_optimized* design. In this example, the programmable optimization performs an optimization on a low noise amplifier (LNA) starting on the input network with its input parameters. The first optimization is followed by a new optimization on the output network with its output parameters, and finally followed by an overall final optimization of the whole LNA. The following figure shows the initial programmable optimization setup details.

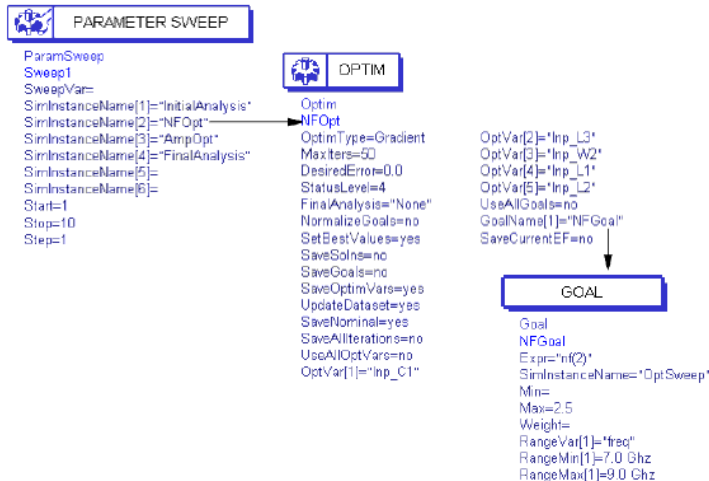
1. In the initial optimization, the *ParamSweep* component is setup as shown below in the following figure.

Parameter Sweep Initial Analysis Setup



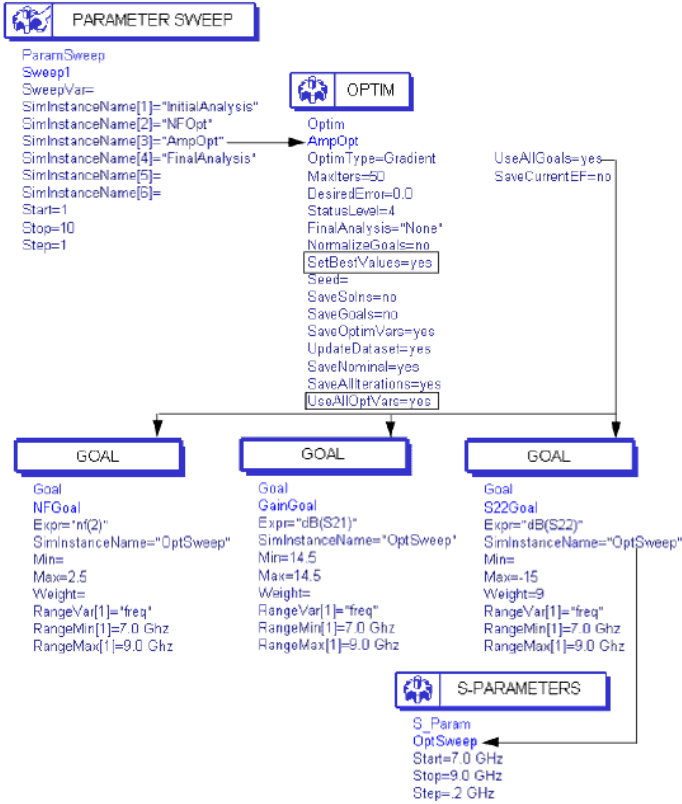
2. Using the Input Matching Network components, the circuit is first setup to be optimized for Noise Figure as shown below in the next figure.

Noise Figure Optimization Setup



3. In the second optimization, the design is optimized for Gain and Return Loss. The optimization starts by using the best values obtained from the first optimization (see parameter *SetBestValue=yes* in the Optim component). The setup is shown below in the following figure.

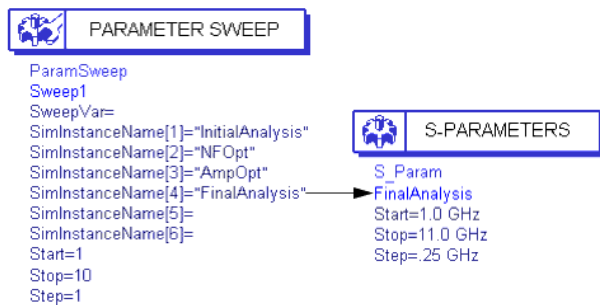
Second Optimization Setup for Gain and Return Loss



Note that the second optimization shown in this figure is set up to optimize all of the amplifier's optimizable parameters for Noise Figure, Gain, and S_{22} specifications (see parameter *UseAllOptVars=yes* in the Optim component).

- The last optimization sequence performs the Final Analysis. For more information, refer to *Final Analysis* (optstat).

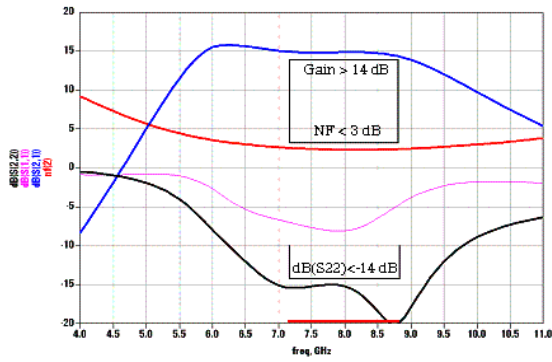
Final Analysis Setup



The results from programmable optimization can indicate the performance of the optimal design for the entire circuit. The following figure shows the results for this example.

Optimal Performance after Programmable Optimization

X-band LNA results after Programmable Optimization



Single Optimization Cockpit Example

This example is called *OptimCockpit_wrk*, and it is located in the directory `$HPPEESOF_DIR/examples/Tutorial`. To access this example workspace and enable simulation, open the example by choosing **File > Open > Example** from the ADS Main window.

The README gives the instructions for this example. The design specifications are given as:

Noise figure: $NF < 2.0$ [8.5 GHz, 11.5 GHz]

$dB(S22)$:

$dB(S22) < -10$ dB [5 GHz, 7 GHz]

$dB(S22) < -18$ dB [9.5 GHz, 11.5 GHz]

$dB(S22) < -10$ dB [13 GHz, 15 GHz]

$dB(S21)$:

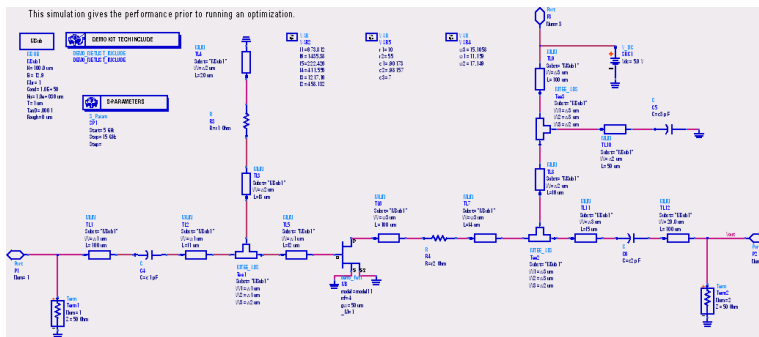
$dB(S21) < 8$ dB [5 GHz, 7 GHz]

11.5 dB $< dB(S21) < 12$ dB [9.5 GHz, 11.5 GHz]

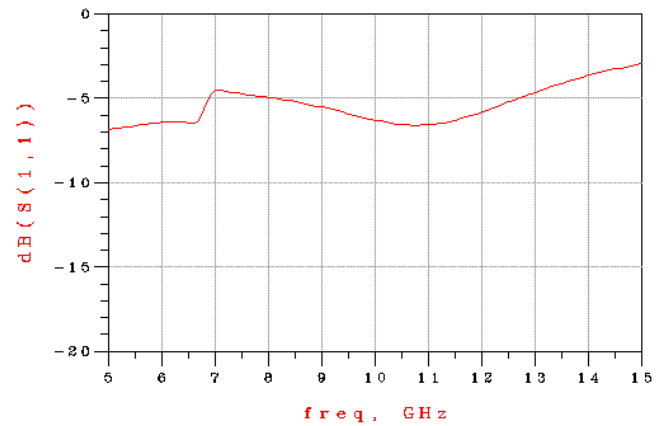
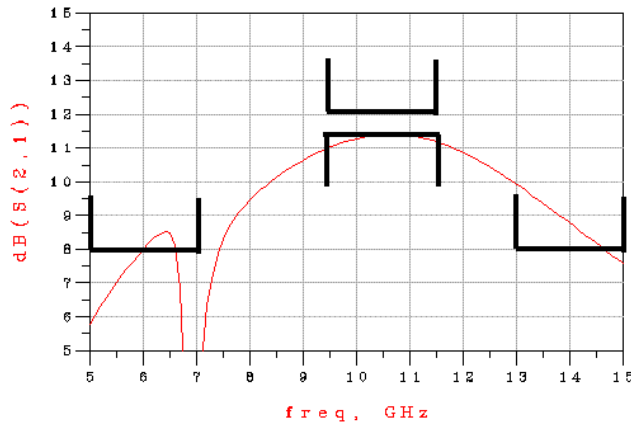
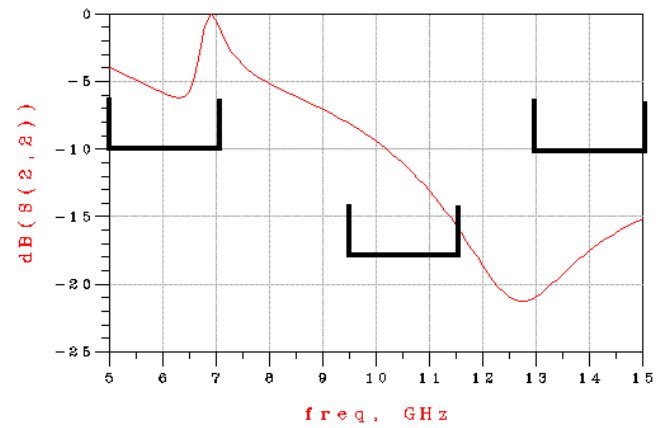
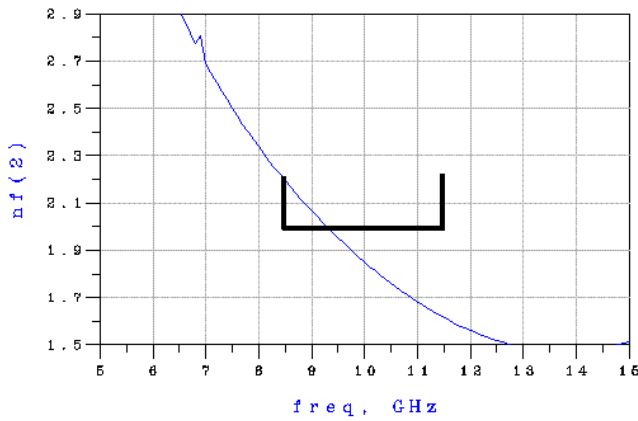
$dB(S21) < 8$ dB [13 GHz, 15 GHz]

1. The initial design is called *LNA_Nominal*. The simulation will run once you click **Simulate** and the results with the drawn limit lines are shown in the *LNA_Nominal.dds*.

Nominal Circuit Design for Low Noise Amplifier



Performance with Design Specifications



- The next design step is to identify the critical design parameters. In this example, you choose them based on the design experience. Other approaches include Tuning and Sensitivity Analysis.
- Set up optimization goals. Based on the design specifications, you will have three goals. Place Goal component from the *Optim/Stat/DOE* palette into the design, and fill in the contents based on the design specifications, as shown in the following figure. Their schematic view is also listed below.

Setup Goals

Optim Goal Input:4

Goal Instance Name: OptimGoal_S23

Expression: $dB(S21)$

Analysis: SP1

Weight: 1

Indep. Vars.: freq

Name	Type	Min	Max	Weight	freq min	freq max
1 S21_Limit1	<	8		1	5 GHz	7 GHz
2 S21_Limit2	Inside	11.5	12	1	9.5 GHz	10.5 GHz
3 S21_Limit3	<		8	1	13 GHz	15 GHz

Optim Goal Input:4

Goal Instance Name: OptimGoal_S22

Expression: $dB(S22)$

Analysis: SP1

Weight: 1

Indep. Vars.: freq

Name	Type	Min	Max	Weight	freq min	freq max
1 S22_Limit1	<	-10		1	5 GHz	7 GHz
2 S22_Limit2	<		-18	1	9.5 GHz	10.5 GHz
3 S22_Limit3	<		-10	1	13 GHz	15 GHz

Optim Goal Input:4

Goal Instance Name: OptimGoal_NF

Expression: $nf(2)$

Analysis: SP1

Weight: 1

Indep. Vars.: freq

Name	Type	Min	Max	Weight	freq min	freq max
1 NF_Limit1	<			2	8.5 GHz	11.5 GHz

Schematic View for Goals

GOAL	GOAL	GOAL
Goal OptimGoal_S23 Expr = "dB(S21)" SimInstanceName = "SP1" Weight = 1.0	Goal OptimGoal_S22 Expr = "dB(S22)" SimInstanceName = "SP1" Weight = 1.0	Goal OptimGoal_NF Expr = "nf(2)" SimInstanceName = "SP1" Weight = 1.0

- Set up optimization variables. Select **Simulate > Simulation Variable Setup** to set up the optimization variables as Min/Max format.

Set up Optimization Variables

Name	Optimized	Value	Unit	Format	Min/+-/+ Unit	Max	Unit	Step	Unit
TL1									
L	<input type="checkbox"/>	100	um	min/max					
Wall1	<input type="checkbox"/>	1.0E+30	um	min/max					
Wall2	<input type="checkbox"/>	1.0E+30	um	min/max					
TL9									
L	<input type="checkbox"/>	100	um	min/max					
Wall1	<input type="checkbox"/>	1.0E+30	um	min/max					
Wall2	<input type="checkbox"/>	1.0E+30	um	min/max					
VAR2									
l1	<input checked="" type="checkbox"/>	673.812		min/max	337	1.01e+03			
l6	<input checked="" type="checkbox"/>	1435.38		min/max	718	2.15e+03			
l5	<input checked="" type="checkbox"/>	222.426		min/max	111	334			
l4	<input checked="" type="checkbox"/>	411.556		min/max	206	617			
l3	<input checked="" type="checkbox"/>	1217.16		min/max	609	1.83e+03			
l2	<input checked="" type="checkbox"/>	458.132		min/max	229	687			
VAR5									
r1	<input type="checkbox"/>	10		min/max					
r2	<input type="checkbox"/>	55		min/max					
c1	<input checked="" type="checkbox"/>	.90173		min/max	0.451	1.35			
c2	<input checked="" type="checkbox"/>	.98157		min/max	0.491	1.47			
c3	<input checked="" type="checkbox"/>	7		min/max	3.5	10.5			
VAR4									
w3	<input checked="" type="checkbox"/>	15.1058		min/max	7.55	22.7			
w1	<input checked="" type="checkbox"/>	11.159		min/max	5.58	16.7			
w2	<input checked="" type="checkbox"/>	17.149		min/max	8.57	25.7			

- Set up methodology. Place an Optimization Controller from *Optim/Stat/Doe* palette into the design.
- Launch Optimization by clicking **Optimize**. The status window will show the simulation header. Then the optimization cockpit will pop-up as shown in the following figure.

Optimization Cockpit Running Mode

Optimization Cockpit

Status
 Continue | Simulate | Update Design...
 Iteration: 25/25 | Elapsed time: 1:15 | Stopping reason: Iteration limit reached

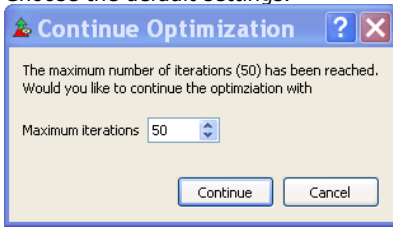
Variables
 12 variables | Start Tuning | Edit variables...
 States: Store... | Recall... | Options...

Goals
 3 goals | Error: 34.6824 | Edit goals...
 Error history | Goal contributions

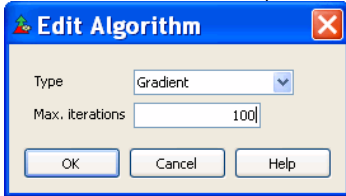
OptimGoal_NF = nf(2)
 OptimGoal_S22 = dB(S22)
 OptimGoal_S23 = dB(S21)

Current values:
 c1: 0.781298, c2: 0.948368, c3: 4.7112, l1: 862.731, l2: 410.857, l3: 1.7638e3, l4: 595.504, l5: 123.605, l6: 1.7217e3, w1: 10.5566, w2: 16.8349, w3: 12.545

- The optimization finishes since the maximum iterations is reached. The error is still very high (34.6824). Increasing the number of iterations help in achieving better results. Click **Continue**. A dialog will pop-up to increase the number of iterations. Choose the default settings.

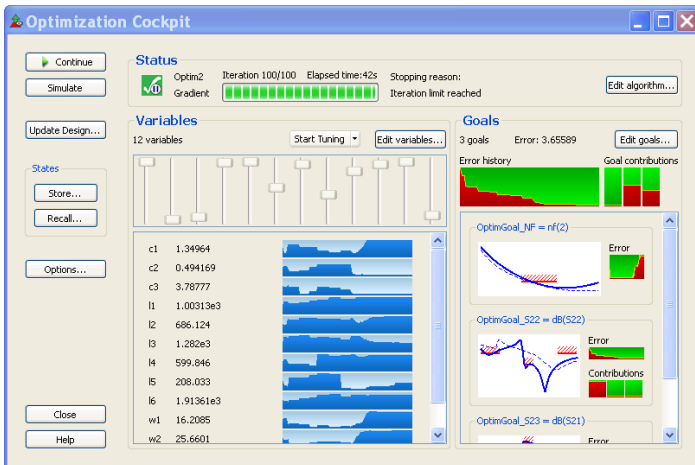


- The optimization will continue to run up to new maximum iterations. The amount of error goes down more. Now change the algorithm to gradient for the faster convergence to the local minimum. Click **Edit Algorithm** to change the algorithm from *Random* to *Gradient*, and choose the maximum iterations to 100.

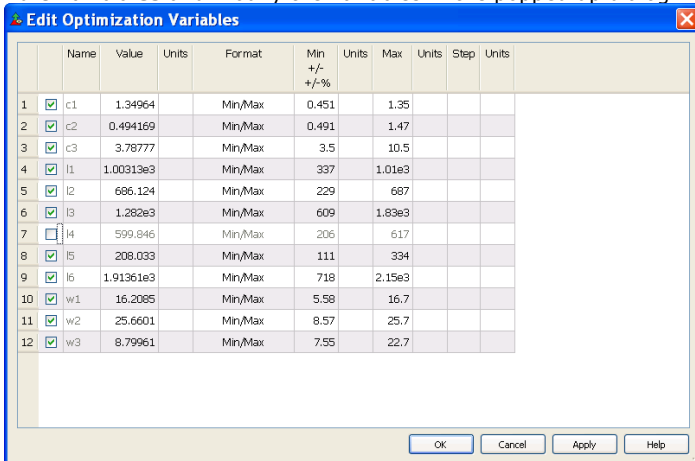


- Click **Continue** to continue the optimization with *Gradient* optimizer. The amount of error drops very quickly. The optimization finishes but not all of the targets are met.

Optimal Status after Gradient Optimization

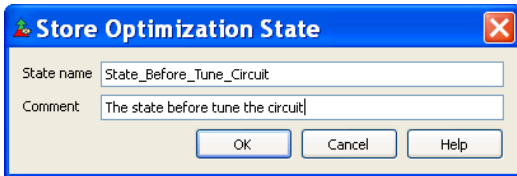


- Now from the slider view of the variables, several variables are in their minimum or maximum limits. From the variable history plot, it seems that I4 is not sensitive. Increase their limits and disable I4 to see if you can achieve all of the targets. Click **Edit Variables** and modify the variables in the popped up dialog.

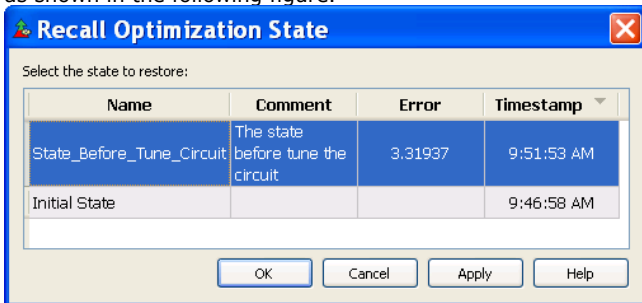


- Click **Continue** to continue the optimization. Soon, Variable $c1$ and $I1$ reach their limits again. Continue to modify the variable setup to see the changes. Click **Pause** to pause the optimizer, and then **Edit Variables** to modify the variables. Or, tune the circuit to see how the variables affect the performance. Before tuning the circuit, store the state first. The state will include the algorithm information, the variable information and the goal information. Click **Store**.

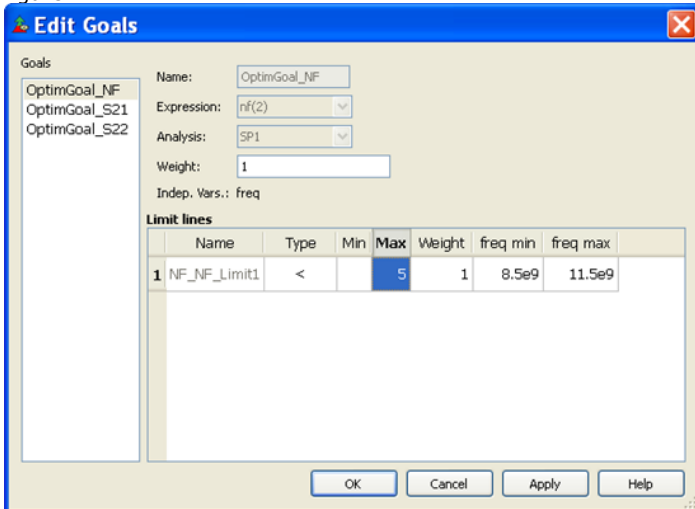
Store State



- Click **Start Tuning** and select the tuning mode as *Simulate While Slider Moves...* As the slider moves, the goal plots on the right will be updated in real-time.
- After doing the tuning a bit, you didn't find better results. So you would like to go back to the state before tuning. Click **Recall** and choose the previously saved state as shown in the following figure.



- Click **Continue** to continue the optimization. From the goal plots, the noise figure is far away from the specification. You would like to loose the targets to see what happens. Click **Edit Goals** to loose target for noise figure as shown in the following figure.

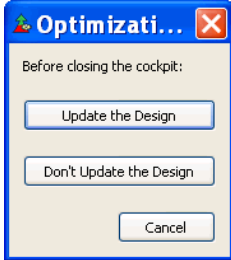


- Click **Continue** to let the optimizer find the optimal results for the loosing targets.

Optimal Results for Loosing Targets

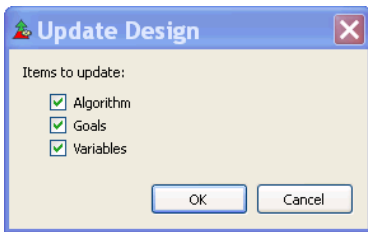


16. If desired results are achieved, you can close the optimization process. Click **Close**. A dialog box will pop-up for the back-annotation.



After selecting **Update Design**, the following dialog allows you to select the specific items to be back annotated. Select all of them, that means all of the variable information, goal information and optimization controller information will be back annotated to the design.

Back Annotation



After the back annotation, the optimization cockpit is closed, and the simulation will stop after the final analysis (if there is any). The DDS will pop up for the final results.

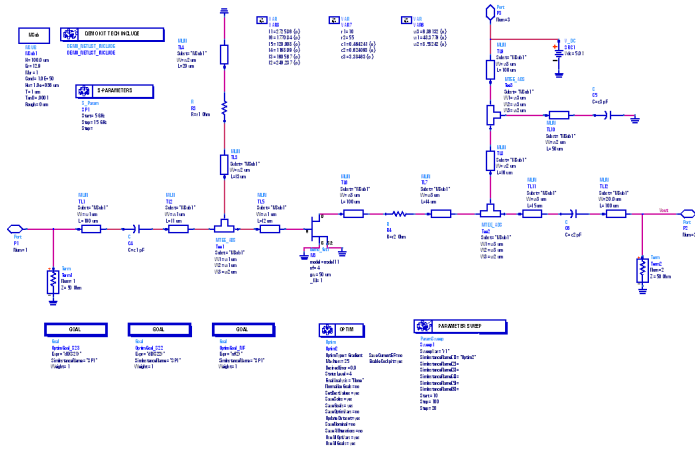
The above example only used the limited functionalities of the optimization cockpit. For more detailed information regarding the interactive optimization cockpit running mode, refer to *Interactive Full Mode* (optstat).

Swept Optimization Cockpit Example

This example is also in *OptimCockpit_wrk*, and it is located in the directory *\$HPPEESOF_DIR/examples/Tutorial*. To access this example workspace and enable simulation, open the example by choosing **File > Open > Example** from the ADS Main window.

The design is SweptLNA_Full. The design is the optimal results from LNA_Full. In this design, we will sweep r1 to see how it affects the optimal design and the circuit stability. The following figures show the design and the view-only optimization cockpit mode.

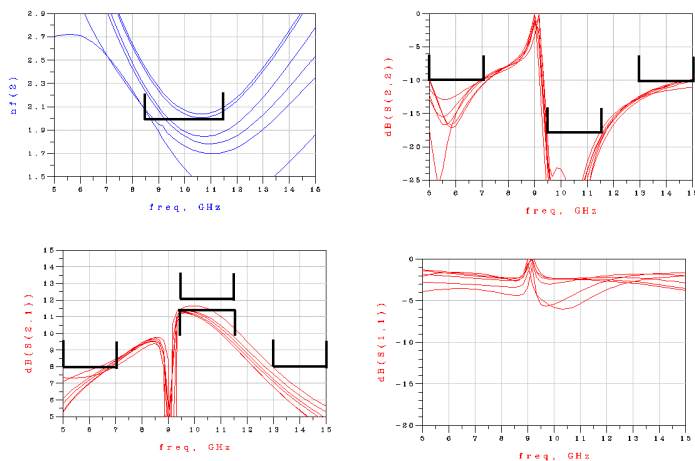
Sweep Optimization Design



View-Only Optimization Cockpit Mode

Variable	Value
c1	0.451081
c2	0.530416
c3	3.11091
I1	1.57448e3
I2	1.19362e3
I3	1.59031e3
I4	901.126
I5	111.767
I6	1.90536e3
w1	49.8322
w2	8.65448
w3	3.52708

Results from Swept Optimization Shown on DDS



For the optimization methodology other than the single optimization, the optimization cockpit will be in the view-only mode since we treat all of the analysis in the design as batch simulation. In the view-only optimization cockpit mode, there is only one active state, which is the one current viewing. The back-annotation will only back-annotate the current view for variables, goals and optimization controller. For more information regarding the view-only optimization cockpit mode, refer to *View-Only Mode (optstat)*.

Troubleshooting

This section provides general information on optimization that may help answer some of your questions when faced with difficulties.

Understanding How ADS Names Optimization Variables

ADS names optimization (as well as yield and DOE) variables in a specific way to avoid confusion in the dataset or Status window. Ignoring that part of the name introduced by the optimization controller, optimization variable names have the following format:

```
<enclosing_definition_name> . <instance_name> . variable_name .
```

For example, you might have a name such as Def1.Inst1.X, where the variable X is defined in the definition Def1.

The instantiation path has the enclosing definition name pre-pended.

Optimization input variables may be derived from variables, or derived from component parameters. The former are always associated with the definition and so have no instantiation path. The latter are associated with an instantiation and therefore have an instantiation path, with the exception of parameters where the default value is an optimization input variable. In this case, if the user does not provide a value, the parameter is associated with the definition.

Understanding Optimization Variable Types

ADS supports both continuous type and discrete type. The optimizer will decide the search range based on the optimization variable setup.

For continuous type, the format include:

1. min/max: the search range will be [min, max]
2. +/- Delta%: the search range will be [nominalValue-nominalValue*Delta%, nominalValue+nominalValue*Delta%]
3. +/- Delta: the search range will be [nominalValue-Delta, nominalValue+Delta]
4. unconstrained: the search range is [0, 2*nominalValue] if the nominal value is greater than 0. Or, [-2*nominalValue, 0] if the nominal value is less than zero. Please pay attention to this type: it does not set the range from negative infinite to plus infinite. It actually set up very narrow range in order to avoid to non-convergence issues for all of the ADS models. It is highly recommended not to use this format.

For discrete type, the format is:

1. min/max/step: the search range is the list based on the min/max/step

Failures that Occur in Evaluating Goals

The most common error from optimization is related with the evaluation of goal expressions. You will see an error in the simulation window such as:

```
Optimization/Statistics Error:
Please check for valid 'Expr' field in OptimGoal/fieldSpec item
Check the following device:
OptimGoal
```

The *Expr* field in a Goal component is a *MeasEqn* (Measurement Equation) component. When the optimization starts, it calls the underlying analysis specified in the *SimInstanceName* field of the Goal component and evaluates the *MeasEqn* before it returns to the optimization process. The error shown above usually indicates that the *MeasEqn* has failed to obtain the underlying analysis. An easy way to verify this is to:

1. Deactivate the Optim components in the design.
2. Deactivate other simulation control components but leave the simulation control component, and others associated with it, indicated by the *SimInstanceName* field of the Goal component that has a problem.
3. Add a *MeasEqn* which equals the *Expr* field of Goal component that has the problem.
4. Run the simulation. When you run the simulation at this time, there are two cases:
 - Error information saying that the simulator cannot resolve/evaluate the *MeasEqn*
 - No warning information. However, when you check the dataset, there is no output for the *MeasEqn*

For such cases, you have to check and make sure the required responses are valid expressions from the underlying analysis (Please remember the basic requirements to apply optimization in the first section in this topic). Otherwise, optimization is not applicable.

Global vs. Local Search

For general information on global and local search, refer to *Obtaining Global Optimal Results* (optstat).

Locating Optimization Examples

To find optimization examples that may help you with understanding a feature, use the *Examples Search* tool in ADS. From the ADS Main window,

1. Click **Tools > Example Search**. The Examples Search tool appears.
2. Enter the keyword *Optim* with/without other keywords.

See also *Optimization Examples* (optstat).

Improving Optimization Speed

Because it requires multiple circuit analyses per iteration of the optimizer, optimization is an inherently slow process. The following precautions can help minimize optimization time:

- Use as few frequency points as possible during optimization
- Minimize the number of variables and circuit nodes
- If a structure is used repeatedly in your circuit, enter it as a subcircuit
- Simplify the optimization criteria as much as possible
- Do not be afraid to experiment with optimizers, goals, and weights
- Begin with a reasonably good initial design
- Ensure that you have not defined conflicting optimization goals

Meeting Troubles in the Cockpit Running Mode

If there is any unexpected error while running the Cockpit mode, disable the cockpit mode and try to run it in stand-alone mode as a workaround. Follow the steps below to disable the cockpit mode:

- Double click the optimization controller
- Click *Parameters* tab
- Uncheck **Enable Optimization Cockpit** option within Other group.

Using Statistical Design

Statistical design is the process of:

- Accounting for the random (statistical) variations in the parameters of a design.
- Measuring the effects of these variations.
- Modifying the design to minimize these effects.

Yield analysis is the process of varying a set of parameter values, using specified probability distributions, to determine how many possible combinations result in satisfying predetermined performance specifications.

Yield is the unit of measure for statistical design. It is defined as the ratio of the number of designs that pass the performance specifications to the total number of designs that are produced. It may also be thought of as the probability that a given design sample will pass the specifications.

Because the total number of designs produced may be large or unknown, yield is usually measured over a finite number of design samples or trials in the process known as yield estimation. As the number of trials becomes large, the yield estimate approaches the true design yield. Parameter values that have statistical variations are referred to as yield variables.

Three statistical design options are available:

- **Yield analysis** This process involves simulating the design over a given number of trials in which the yield variables have values that vary randomly about their nominal values with specified probability distribution functions. The numbers of passing and failing trials are recorded and these numbers are used to compute an estimate of the yield.
- **Monte Carlo analysis** This process involves simulating the design over a given number of trials in which the yield variables have values that vary randomly about their nominal values with specified probability distribution functions.
- **Yield optimization** Also known as design centering, this process involves multiple yield analyses with the goal of adjusting the yield variable nominal values to maximize the yield estimate. During yield optimization, each yield improvement is referred to as a design iteration.

Yield and Monte Carlo analysis and yield optimization for ADS are supported as follows:

- **Analog/RF Systems** Any analysis type (such as AC, DC, S-Parameter, Harmonic Balance, Circuit Envelope, and Transient simulation types).
- **Signal Processing** For ADS Ptolemy simulation.

Statistical Design Minimum Requirements

Prior to performing a statistical design, you need:

- At least one component parameter in your design identified as a yield variable. You specify details in the Component Parameter dialog box by choosing the **Tune/Opt/Stat/DOE Setup** button.
- At least one yield specification (*YieldSpec*) component specified, then placed in the design window. A yield specification defines a single (or double) sided range of acceptable performance for a given response. For each trial during yield analysis, the yield spec is compared to the simulated response to determine the pass/fail status of the current trial.
- One yield analysis (*Yield*) or yield optimization (*YldOpt*) component placed in the design window to specify all *YieldSpec* components to be included, data to be saved, Shadow model type (if used), enabling of post production tuning, and parameters to be displayed.
- One simulation control component (for example, an AC, DC, S-Parameter, Harmonic Balance, Circuit Envelope or Transient component for Analog/RF Systems or a Data Flow Controller for Signal Processing).

The design components needed for yield analysis are located in the *Optim/Stat/Yield/DOE* library or palette for Analog/RF Systems and the *Controllers* library or palette for Signal Processing. They include the following:

- Yield analysis (*Yield*)
- Monte Carlo analysis (*MC*)
- Specification for yield analysis (*YieldSpec*)
- Yield Optimization (*YldOpt*)

Performing Yield Analysis

Yield analysis determines the percentage of acceptable and unacceptable units based on the *YieldSpec* component definitions. Yield analysis randomly varies network parameter values according to statistical distributions while comparing network measurements to the user-specified performance criteria found in the *YieldSpec*.

Yield analysis is based on the Monte Carlo method. A series of trials is run in which random values are assigned to all of your design's statistical variables, a simulation is performed, and the yield specifications are checked against the simulated measurement values. The number of passing and failing simulations is accumulated over the set of trials and used to compute the yield estimate.

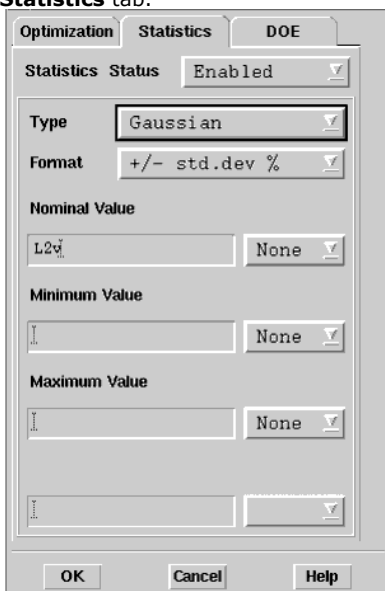
Other capabilities of yield analysis include the following:

- Accumulated sets of selected network responses can be viewed or plotted.
- Performance histograms can be viewed or plotted to display the distribution of measured circuit responses. Refer to [Creating a Measurement Histogram](#) for full details. Refer to *Editing Traces* (data) for details on viewing these histograms.
- Overall performance variation can be assessed as a result of random variations in component parameter values.

Specifying Component Parameters for Yield Analysis

The procedure for specifying components for yield analysis is as follows:

1. Select and place an appropriate component from one of the component palettes or component libraries. For example, place a parallel resistor-inductor-capacitor (PRLC) from the Lumped Components palette.
2. Double-click on the component in the design window to access its associated dialog box.
3. From the dialog box, highlight the parameter that you want to vary in the Select Parameters box (for example R for parallel resistance), then choose the **Tune/Opt/Stat/DOE Setup** button, which will only appear for valid statistical parameters. The Setup dialog box appears, with the Optimization tab active. Click the **Statistics** tab.



4. From the Statistics Status drop-down list, select **Enabled** so that you can set specification of the appropriate fields. *Enabled* causes the parameter to be varied when the simulation is run. *Disabled* allows you to temporarily suspend any parameter variation previously assigned, and *Clear* removes the values you previously applied to the design.
5. From the Type drop-down list, select an appropriate statistical Value Type from:
 - Gaussian
 - Uniform
 - Discrete
 - LogNormal
 For descriptions of Value Types, refer to the section *Value Types for Statistical Design* (optstat) in *Available Value Types* (optstat).
6. From the Format drop-down list, select an appropriate statistical value format:
 - For Gaussian, choose *+/- std.dev. %* or *+/- std.dev.*
 - for Uniform, choose *min/max*, *+/- Delta %* or *+/- Delta*.

- for Discrete, only *min/max/step* is available.
 - for LogNormal, choose *+/- std.dev. %* or *+/- std.dev.*
- For complete descriptions of the available format, refer to the section *Value Types for Statistical Design* (optstat) in *Available Value Types* (optstat).
7. If you selected *std.dev* or *+/- std.dev %* or *+/-Delta* or *+/-Delta %* formats, specify the deviation value. For these formats, the units can also be specified in the drop-down list next to each input field.

Note
For a *Gaussian* or *LogNormal* distribution, *std.dev* refers to the standard deviation (or sigma), either as a percentage of the nominal value or an absolute value. For a *Uniform* distribution, *Delta* refers to the distribution limit values, either as a percentage of the nominal value or an absolute value.

8. If you selected a *min/max* format, you can optionally enter values for nominal, minimum, and maximum in the appropriate boxes, and select an appropriate unit assignment for each from the drop-down list next to the boxes.

Note
Unit specification via the Setup dialog box is not possible for variables defined in the *Var/Eqn* component.

9. From the Nominal Value field and the Units drop-down list, the value and units in your design for this component are displayed. You can change these if you wish.
10. If you intend to include any of the parameters of this component for post production tuning, click the **Optimization** tab and click the Post Production Tuning checkbox. For more details, refer to the section, [Enabling Post Production Tuning](#).
11. Choose **OK**.

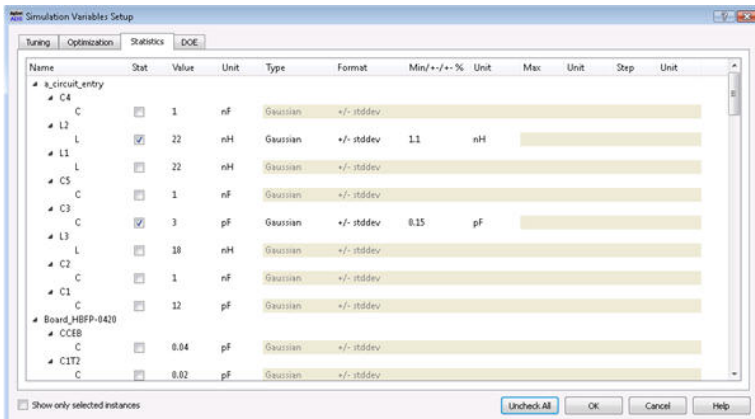
Specifying Multiple Parameters for Yield Analysis

The *Simulation Variables Setup* dialog lists all the component parameters in a design while allowing the simultaneous adjustment of a number of parameters across multiple components on a single schematic. To view the dialog, in the *Schematic* window, click **Simulate > Simulation Variables Setup**.

To specify components for yield analysis, follow these steps:

1. Select the checkbox in the *Statistics* column to enable yield analysis for a particular component.
2. In the *Type* drop-down list, select the appropriate statistical Value Type from, Gaussian, Uniform, Discrete and Lognormal.
3. From the *Format* drop-down list, select an appropriate statistical value format:
 - For Gaussian, choose *+/- std.dev. %* or *+/- std.dev.*
 - for Uniform, choose *min/max* , *+/- Delta %* or *+/- Delta*.
 - for Discrete, only *min/max/step* is available.
 - for LogNormal, choose *+/- std.dev. %* or *+/- std.dev.*

For complete descriptions of the available format, refer to the section *Value Types for Statistical Design* (optstat) in *Available Value Types* (optstat).
1. The default values for *Min*, *Max*, and *Step* are displayed as appropriate. To modify a *Min*, *Max*, and *Step* value, enter the desired value in the appropriate field. The default values for *Min*, *Max*, and *Step* are 50%, 150% and 10% of the nominal value of the parameter, respectively.
2. To disable yield analysis, deselect the checkbox in the *Statistics* column.
3. Select the *Show selected items only* checkbox to display only the selected components in the schematic.
4. Click **Deselect All** to deselect all parameters.
5. Click **Ok** to close the dialog.



Placing an Appropriate Simulation Control Component for Yield

Analysis

An appropriate simulation control component must be placed in the design prior to initiating a yield analysis.

For Analog/RF Systems simulation, all analysis types are supported, for example place one of the following components:

- **AC** from the AC Simulation palette or library
- **DC** from the DC Simulation palette or library
- **S-Param** from the S-Param Simulation palette or library
- **Harmonic Balance** from the HB Simulation palette or library
- **ENV** from the Envelope Simulation palette or library
- **Tran** from the Transient Simulation library

For Signal Processing, place a:

- **Data Flow** controller

For details on specifying parameters for each of these control components, refer to *Using Circuit Simulators (cktsim)* or *ADS Ptolemy Simulation (ptolemy)*.

Setting Up a Yield Specification

Yield specifications are defined by placing a *YieldSpec* component (which is accessed from the Optim/Stat/Yield/DOE palette or library for Analog/RF Systems or from the Controllers palette or library for Signal Processing). Once placed, you can double-click it to display the *Specification for Yield Analysis* dialog box.

You can place more than one *YieldSpec* component if needed. The *YieldSpec* s to be used are referenced in the Yield Simulation dialog box, as described in the section, [Setting Job Parameters for Yield Analysis](#).

To set appropriate yield specifications in this dialog box:

1. If desired, enter a name in the Instance Name field that is different from the assigned default name shown.
2. In the Select Parameter list box on the left, click on each parameter that you want to set up, then make other associated specifications in the box on the right. When you select a parameter, such as Expr, all relevant items in your design will be displayed in the box. The style of this box varies depending on the parameter, as described in the table below.

Parameter	Description	Use Model
Expr	A valid AEL expression that operates on the simulation results, such as mag(S11). For more information on AEL expressions, refer to <i>AEL (ael)</i> or <i>Measurement Expressions (expmeas)</i> .	The list box label becomes Measurement Equations. All associated expressions are displayed in the box. Select the one you want to use and it will appear just below in the Selection box. For expressions unrelated to MeasEqns, you must type them in the Selection box.
SimInstanceName	Enter the instance name for the simulation control component that you placed in your design, which will generate the data used by the Expr field.	The list box label becomes Analysis Components. Select the analysis component (simulation controller), such as S-parameter, that you want to use and it will appear just below in the Selection box.
Min	Enter a number for a minimum acceptable response value.	Fields for Parameter Entry Mode and Equation editor are used as in any component parameter dialog box. Type a value in the box. Note: Both Min and Max do not have to be specified, but at least one does.
Max	Enter a number for a maximum acceptable response value.	Same as above.
Weight	The <i>Weight</i> parameter is irrelevant in YieldSpec component.	Fields for Parameter Entry Mode and Equation editor are used as in any component parameter dialog box. Type a value in the box.
RangeVar	Independent variable name.	Same as above, but note that this parameter is "indexable" and can be applied to more than one independent variable.
RangeMin	Minimum limit of range for independent variable during optimization.	Same as above.
RangeMax	Maximum limit of range for independent variable during optimization.	Same as above.

Setting Job Parameters for Yield Analysis

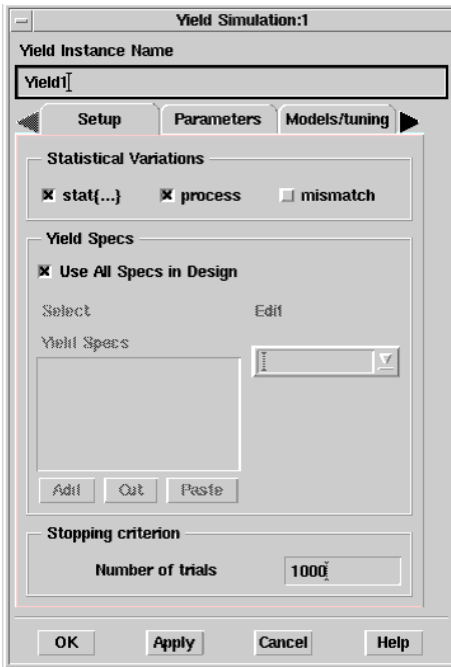
To set job parameters, you need to specify appropriate data in the Yield Simulation dialog box.

This four-tabbed dialog appears when you place a Yield Analysis component (labeled *Yield*). Do the following:

1. Place the *Yield Analysis* component in the appropriate design window.
2. Double-click the component to bring up the dialog box. The Setup tab is active.
3. Make specifications in each tab (Setup, Parameters, Models/tuning, and Display) of the dialog box, as described in the next sections.

Selecting a Yield Specification

First select the Setup tab of the Yield Simulation dialog box to set up a yield analysis.



- The options in the Statistical Variations box support two different statistical models:
 - ADS traditional model - The statistical variables use ADS syntax and are set up as $R=50 \text{ stat}\{\text{gauss } +/-1\%\}$.
 - Spectre statistical model - The variations *process* and *mismatch* use Spectre syntax and are provided only for use in ADS Dynamic Link. They are specified in the Statistics block in the foundry files, and define the process variations and mismatch variations.

When choosing options in the Statistical Variations box:

- `_stat_{...}` refers to the statistical variables defined in ADS traditional model. Choose `_stat_{...}` to set the ADS syntax for statistical variations.
 - process* and *mismatch* refer to the statistical variables defined in a Spectre model.
 - If there are no variables set, checking any option will not affect the simulation. For example, the ADS traditional model uses only `stat_{...}` variables. Selecting *process* or *mismatch* will not change the results.
- In the Yield Specs box, accept the default *Use All Specs in Design* checkbox. This is the best approach for most designs, and all Yield Spec components placed in a design will be implicitly associated with the Yield controller. To associate a *subset* of all Yield Specs with a given Yield controller, deselect the *Use All Specs in Design* checkbox. Select a Yield Spec from the Edit drop-down list, which will include all yield specification components that are currently placed in the design, as described in the section, [Setting Up a Yield Specification](#). Choose *Add* to place in the *Yield Specs* box, and repeat this procedure if necessary. Choose the *Cut* or *Paste* buttons, if necessary to make any changes in the *YieldSpecs* box.
 - Under *Stopping criterion*, specify the number of desired trials to use during the yield analysis process.
 - Choose **Apply** to retain the specifications that you have made while you enter data into the Parameters tab, as described in the next section.

Setting Parameter Information for Yield Analysis

You set parameter information in the Parameters tab of the Yield Simulation dialog box, such as what data to save and parameter attributes. To do this, follow these steps:

The screenshot shows the 'Yield Simulation' dialog box with the following sections:

- Output Data:**
 - Analysis outputs
 - YieldSpec expressions
 - Random variables
- Output Data Control:**
 - Save data for all trials
 - Update display during Yield Analysis
- Levels:**
 - Status level:
- Other:**
 - Seed:

- In the Output Data field, specify which data you want to retain in your dataset following yield analysis. Check the following choices that apply.
 - Analysis outputs* sends all measurements (including measurement equations) to the dataset. This can create a substantial amount of data.
 - YieldSpec expressions* (default) sends the result of each active YieldSpec's *Expr* field to the dataset.
 - Random variables* sends the values of all random variables to the dataset for each Monte Carlo trial.
- In the Output Data Control field, specify whether you want to:
 - Save data for all trials*. Data for all trials is saved. This can create a substantial amount of data.

Note
For yield analysis, enabling this feature can slow the analysis time considerably when many trials are being run. The default is off, where only the first and last trials are saved to the dataset.

- Update display during Yield Analysis* (default). This updates the dataset on each yield analysis trial so you can see the results in the Data Display window as they occur (instead of waiting to the end where all the traces are displayed at once).
- In the *Levels* field, enter a number for the desired annotation level. Levels are 0-4, with increasing information displayed in the Status window. (2 is the default.)
 - In the *Other* field, specify a seed value. *Seed* is a value for the random number generator used by the simulator to initiate yield analysis. If *Seed* is not specified, the simulator chooses its own seed, which will be different each time a yield analysis is performed.
 - Choose **Apply** to retain the specifications that you have made while you enter data into the Models/tuning tab, as described in the next section.

Selecting a Shadow Model Type for Yield Analysis

You use the Models/tuning tab of the Yield Simulation dialog box to select the Shadow Model, an optional method of yield analysis.

The Shadow Model works as follows: A series of trials is run in which the random variations in your design's statistical parameters are used in a mathematical model of the design's performance to compute the yield. This allows a greater number of trials and therefore greater accuracy in the yield estimate without a significant increase in the computation time required.

To enable one of two available methods of Shadow Model analysis:

The screenshot shows the 'Shadow model type' dialog box with three radio button options:

- None
- Maximally flat quadratic (faster)
- Agilent EEsof (more accurate)

Click the selection of your choice, using either of two methods:

- Maximally flat quadratic Shadow Model (which is usually faster)
- Agilent EEsof Shadow Model (which is usually more accurate)

If **None** is selected, the Monte Carlo method will be applied to the simulator, not to the mathematical Shadow Model.

Consult the following references for details concerning the Monte Carlo method and the Maximally Flat Quadratic Approximation model.

- R. Spence and R. S. Soin. *Tolerance Design of Electronic Circuits*, Addison-Wesley, 1988.
- Radoslaw (Radek) Biernacki, John Bandler, Jian Song, and QI-Jun Zhang. *Efficient Quadratic Approximation for Statistical Design*, IEEE Transactions on Circuits and Systems, vol. 36, No. 11, November 1989.

Enabling Post Production Tuning

The Models/tuning tab of the Yield Simulation dialog box is also used to enable post production tuning. This section includes a description of the post production tuning feature, followed by a section on setting up post production tuning.

Purpose of Post Production Tuning

During yield analysis, the component parameters that are specified as yield variables are allowed to vary statistically about their nominal values using given probability distribution functions. The yield estimate that results is derived from a given number of trials with these parameter variations.

If, for example, your circuit contained several fixed valued resistors and capacitors with manufactured values known to vary about their nominal values, this yield estimate would take this variation into account.

Suppose that in addition to these fixed valued components, your circuit also contains a trimmer-capacitor and potentiometer that you can tune at the end of production in one last effort to meet specifications. The standard yield estimate would not take this tunability into account.

For such a situation, the simulator can employ Post production tuning. This feature, combined with yield analysis, allows certain parameter values to receive additional tuning if they first resulted in a failure to meet the yield specifications. The parameters that receive this special treatment are referred to as post production tunables.

Setting Up Post Production Tuning

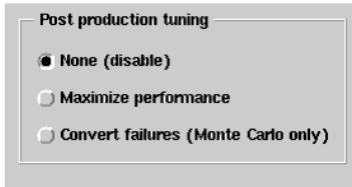
Post production tunables are allowed to take on values over a range as specified in the Optimization tab of the parameter component Setup dialog box. To set up post production tuning:

- First use the Optimization tab of the Component Parameter Setup dialog box (which you access by selecting the **Tune/Opt/Stat/DOE Setup** button) to specify the optimization range for the Post Production Tuning parameter, and be sure to check the **Post Production Tuning** checkbox to enable the variable for tuning. For more information, refer to *Specifying Component Parameters for Optimization* (optstat).
- Remember that a parameter designated for Post Production Tuning must have a yield distribution assigned to it.
For example, if you wanted to specify "R" as a post production tuning variable with discrete optimization range and discrete statistical distribution, the format for R requires the following:
`R=10 opt{ppt discrete 0 to 15 by 1} stat{discrete 0 to 15 by 1}`

Post production tuning is supported for both yield analysis and yield optimization. Note that some trials take longer because of the tuning that must take place in the background.

To set specifications for post production tuning, select one of the following from the Models/tuning tab of the Yield Simulation dialog box:

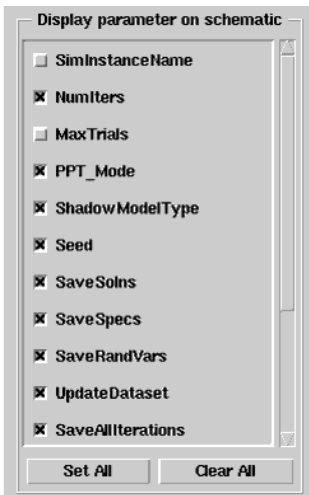
- **None** (to disable post production tuning)
- **Maximize performance** (unconditionally attempts to find the best performance for each trial)
- **Convert failures** (attempts to convert any failed trial into a passing trial, but does not attempt to tune for maximum performance)



- Choose **Apply** to retain the specifications that you have made while you enter data into the Display tab of the Yield Simulation dialog box, as described in the next section.

Displaying Analysis Data on the Schematic

Selecting the yield analysis parameters that will be displayed on your schematic is done the same way as in nominal optimization. Refer to *Displaying Analysis Data on the Schematic* (optstat) for details. Below is a yield analysis example.



When you have finished setting up all the tabs in the Yield Simulation dialog box, click **OK**.

Note

MaxTrials, *Enable* and *RestoreNom* are reserved parameters and are not currently available. Selecting these parameters in the *Display parameter on schematic* section will display the parameters on the schematic; however, changes to the parameter values will not be recognized.

Initiating Yield Analysis

To initiate a yield analysis, select **Simulate** or click the **Simulate** button on the toolbar. The analysis status is displayed in the Status window. Upon completion of the analysis, the simulator ceases analysis and indicates success.

Note

If the yield analysis process becomes exceedingly long, you can use the Stop Simulation command on the Simulation/Synthesis menu in the status window to interrupt the process.

Swept Yield Analysis

Yield analysis can be swept as any other ADS analysis. When the Yield analysis controller is referenced by a parameter sweep controller, the yield analysis is performed for each value of the sweep variable and the results output as a function of the sweep variable. For more information, refer to *Swept Optimization* (optstat).

Sweeping Temperature in a Yield Analysis

Varying the temperature randomly while doing a Yield analysis is a bit complicated. The temperature, *temp*, can be swept in a Parameter Sweep but you are not allowed to specify it as a variable in a VarEqn. ADS allows sweeping temperature in a Parameter Sweep during a Yield analysis. These steps show how to set up a Yield analysis where the temperature is randomized:

1. Make the *YieldSpec* refer to a Parameter Sweep instead of the actual analysis.
2. Make the Parameter Sweep sweep the desired analysis.
3. Set the Parameter Sweep's variable to *temp*.
4. Create a variable (or whatever is needed): `vtemp = 25 stat{uniform 0 to 125}`
5. Make the Parameter Sweep vary between *vtemp* and *vtemp*, with a single sweep point:

```
Start=vtemp
Stop =vtemp
Nr Points = 1
```

Yield Analysis Example

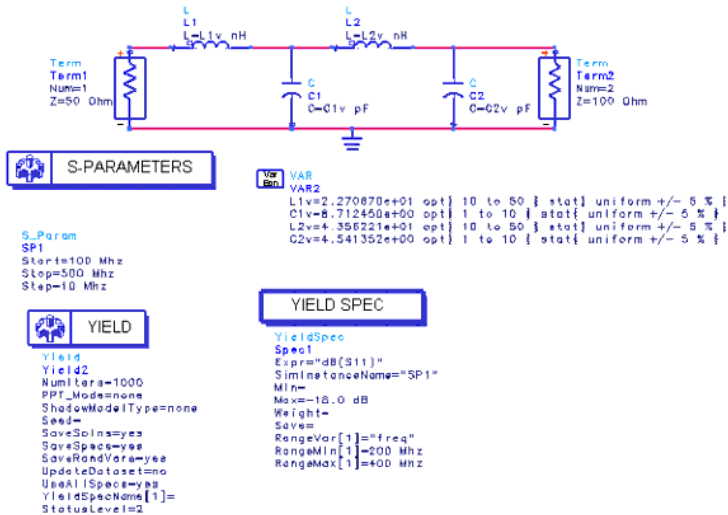
Note

The example used in this topic is from Analog/RF Systems simulation. However, the yield analysis procedure is the same for ADS Ptolemy Signal Processing simulation.

For the following yield analysis example, suppose that we start with the octave-band, 2-to-1 matching transformer that we optimized to have an optimal equal-ripple response in the previous example. Now suppose there is a specification on the design that it must have an input return loss of at least 18 dB from 200 MHz to 400 MHz.

This example is called *yield1_wrk*, and it is located in the directory *\$HPESOF_DIR/examples/Tutorial*. To access this example workspace and enable simulation, first copy it to a work directory. To open an example choose **File > Open > Example** from the ADS Main window.

1. A *YieldSpec* component is added to the design to define that specification, as shown in the following figure.



Design Including YieldSpec Component

2. Assume the following:
 - You can obtain inductances of any nominal value between 10 and 50 nH.
 - You can obtain capacitors having any nominal value between 1 and 10 pF.
 - The tolerance on the inductance and capacitance values is +/- 5%.
This information can be specified in the schematic by changing the Value Types of the appropriate parameters to *Uniform*, and the value type format to *+/-Delta%*.
The schematic represents the octave-band, equal-ripple, 2-to-1 matching transformer, modified to specify the tolerances on the inductance and capacitance values.
3. A yield analysis of the design indicates about 77% yield in the status window.

Creating a Measurement Histogram

A measurement histogram displays the frequency of occurrence of a selected single scalar measurement function. A histogram is a bar graph in which the height of each bar represents the number or the percent of the recorded measurement values that occurred within each measurement value bin.

- The scalar measurement value appears on the X-axis.
- The X-axis is divided into bins.

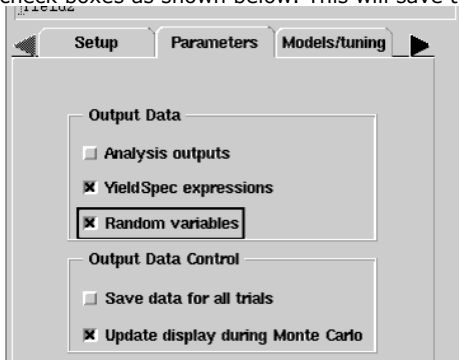
- The Y-axis displays the number or percentage of simulated measurement points that fall into the range of each measurement bin.

You can set the minimum and maximum display range on the X-axis, as well as the number of bins using the histogram AEL expression. Refer to *histogram* in *Measurement Expressions* (expmeas) for more information.

Setup Requirements

Histograms are created after your yield analysis is complete. The stored data is post-processed into histogram form via a set of equations and expressions in the Data Display window. The prerequisites for creating a measurement histogram are:

1. Set up and perform a yield analysis. For the general procedure refer to [Performing Yield Analysis](#). However, there are a few specific settings that must be selected to enable histogram generation, as described next.
2. Edit the Yield component to select the Parameters tab of the Yield Simulation dialog box.
3. In the Output Data field, select the **YieldSpec expressions** and **Random variables** check boxes as shown below. This will save the data needed for post-processing.



4. Click **OK**.
5. Initiate your yield analysis by selecting **Simulate** or click the **Simulate** button on the toolbar.

Generating the Measurement Histogram

To learn how to generate a histogram we will use the same filter design shown in the figure [Design Including YieldSpec Component](#) and, in addition, we will use the post-processing capability of the Data Display window. Do the following:

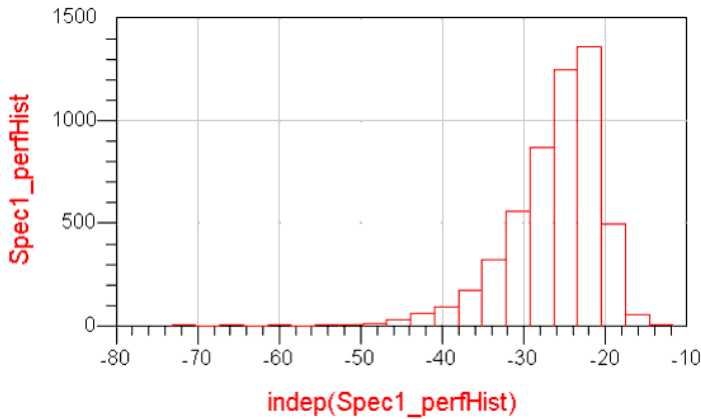
1. After the yield analysis simulation is complete, choose **Window > New Data Display**. The Data Display window appears.
2. Choose **Insert > Equation** or click the **Eqn** button on the left side of the window.
3. Position the pointer on the display and click. The *Enter Equations* dialog box appears.

Note
For more information on entering equations, refer to *Equations (data)* in the *Data Display (data)* documentation.

4. Type in the following equation and click **OK**. Or, in the finished example, access the data display file
`$HPESOF_DIR/examples/Tutorial/yldex1_wrk/measurement_hist.dds`

```
Spec1_perfHist = histogram_stat(Spec1,,200MHz,400MHz,20)
```

When the equation is entered using our example, the following histogram is generated.



Measurement Histogram

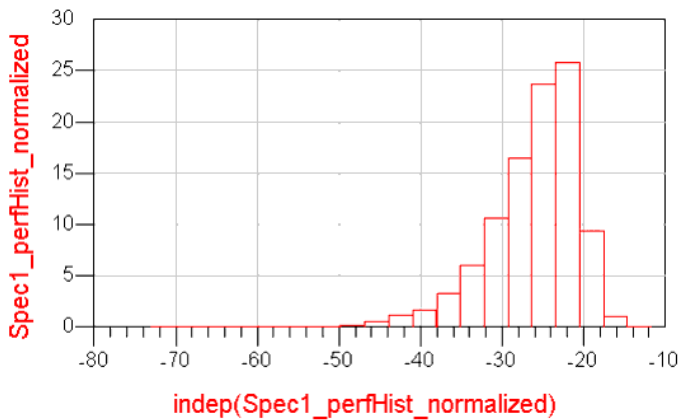
Histogram_stat(data, normalized, innermostindepLow, innermostindepHigh,numBins,minBins,maxBins) is for producing yield histogram. The first argument is required, others are options. The function gives the same functionality as the following series of equations and expressions, which are hidden inside histogram_stat.

```
freq=indep(Spec1)
Spec1_freq_low_limit=200 Mhz
Spec1_freq_high_limit=400 Mhz
Spec1_low_index=find_index(freq[0,:], Spec1_freq_low_limit)
Spec1_high_index=find_index(freq[0,:], Spec1_freq_high_limit)
Spec1_subrange=Spec1[:,Spec1_low_index::Spec1_high_index]
Spec1_subrange_freq_collapsed=collapse(Spec1_subrange)
Spec1_perfHist=histogram(Spec1_subrange_freq_collapsed)
```

The design-specific parameters shown above, such as a low-frequency limit of 200 MHz and a high-frequency limit of 400 MHz, only apply to this example. Your design will vary. The idea here is to understand the methodology of entering a series of equations and expressions to generate a histogram from your data.

1. If you want to generate a histogram with percent on the Y-axis instead of the number of outcomes, set the second argument of histogram_stat to "yes":

```
Spec1_perfHist_normalized=histogram_stat(Spec1,"yes",200MHz,400MHz,20)
```



Normalized Histogram

The following ADS expressions are used in the preceding example:

- histogram()
- find_index()
- yield_sens()
- collapse()
- histogram_sens()

For more information, refer to the *Measurement Expressions* (expmeas) documentation.

Note

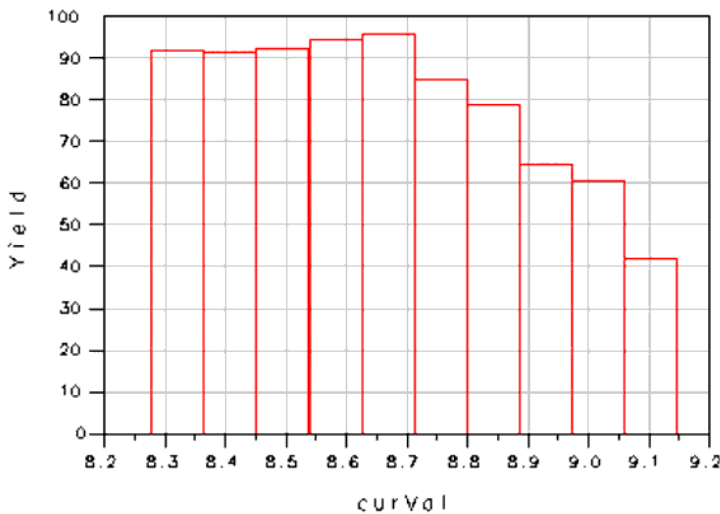
ADS names optimization/yield/DOE variables in a specific way to avoid confusion in the dataset or Status window, as follows:
 <enclosing_definition_name>.<instantiation_path>.variable_name. For more information on this naming convention, refer to *Understanding How ADS Names Optimization Variables* (optstat).

Creating a Sensitivity Histogram

A sensitivity histogram displays the sensitivity of a measurement statistical response to a selected statistical variable. See the following figure for a sample sensitivity histogram.

A sensitivity histogram is a bar graph that shows the effect on a specified statistical response versus the value of a selected statistical variable. ¹

- The X-axis shows the range of value of the selected statistical variable divided into "bins".
- The Y-axis shows, for each bin, a point estimate of the performance response. This point estimate is the statistical response. Possible examples for statistical response are yield, average performance, performance variance, as well as others. The following figure shows a sensitivity histogram with yield as the statistical response.



Parts of a Sensitivity Histogram Display

1. Michael D. Meehan and John Purviance, *Yield and Reliability in Microwave Circuit and System Design*, Artech House Inc., 1993.

Setup Requirements

Sensitivity histograms are created the same way as measurement histograms, as explained in the last section. The only difference is the additional equations and expressions entered after your yield analysis is complete. Refer to [Setup Requirements](#) for the setup procedure.

Generating the Sensitivity Histogram

Note

Generating a sensitivity histogram involves the same steps as generating a measurement histogram, described in the last section, except that additional equations and expressions are required.

To learn how to generate a sensitivity histogram, we will use the same filter design shown in the figure [Design Including YieldSpec Component](#) and we will use the post-processing capability of the Data Display window. Do the following:

1. After the yield analysis simulation is complete, choose **Window > New Data Display**. The Data Display window appears.
2. Choose **Insert > Equation** or click the **Eqn** button on the left side of the window.
3. Position the pointer on the display and click. The *Enter Equations* dialog box appears.

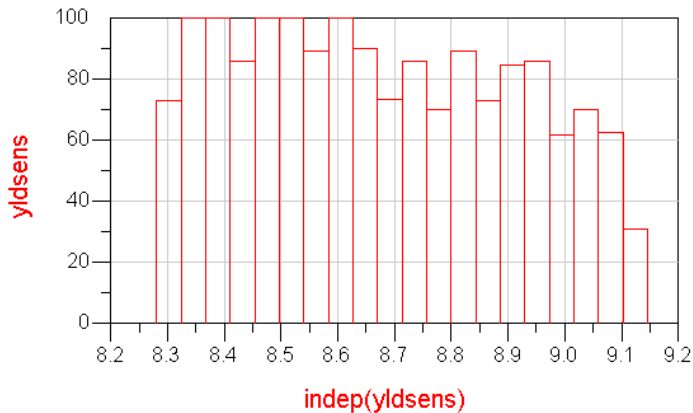
Note

For more information on entering equations, refer to *Equations (data)* in the *Data Display (data)* documentation.

4. Type in the following equation and click **OK**. Or, for the finished example, access the data display file `./yldex1_wrk/yield_sens.dds`.

```
Yldsens=histogram_sens(Spec1,C1v,-18.0,200MHz,400MHz,20)
```

When the equation is entered using our example, the following sensitivity histogram is generated:



Yield Sensitivity Histogram

`Histogram_sens(data,sensitivityVar,goalMin,goalMax,innermostindepLow,innermostindepHigh,numBins)` is for producing a sensitivity histogram. Among its arguments, *data*, *sensitivityVar*, *goalMin* and/or *goalMax* are required. Others are optional. The function gives the same functionality as the following series of equations and expressions, which are hidden inside `histogram_sens`.

```
freq=indep(Spec1)
Spec1_freq_low_limit=200 Mhz
Spec1_freq_high_limit=400 Mhz
Spec1_low_index=find_index(freq[0,:], Spec1_freq_low_limit)
Spec1_high_index=find_index(freq[0,:], Spec1_freq_high_limit)
Spec1_subrange=Spec1[:,Spec1_low_index::Spec1_high_index]
fail=0.0
pass=1.0
curVar=C1v
curSpec1_value=-18.0
maxS11=max(Spec1_subrange)
maxS11_vs_curVar=vs(maxS11, curVar)
pf_maxS11=if(maxS11_vs_curVar > curSpec1_value) then fail else pass
yldsens= yield_sens(pf_maxS11)
```

Notice that the yield is higher for smaller values of C1v. This indicates that yield can be improved by reducing the nominal value of C1v. Also notice that this sensitivity plot includes the effects of all other statistical variables in the design (C2v, L1v, and L2v). In addition, sensitivity plots provide valuable insight into component tolerances and can be used to identify *problem* components.

The design-specific parameters shown above, such as a low-frequency limit of 200 MHz and a high-frequency limit of 400 MHz, only apply to this example. Your design will vary. The idea here is to understand the methodology of entering a series of equations and expressions to generate a histogram from your data.

Performing Monte Carlo Analysis

Monte Carlo analysis is similar to Yield analysis. Both analyses randomly vary network parameter values according to statistical distributions to get the overall performance variation. The only difference is that yield analysis will determine the number of passing and failing simulations over the set of trials and is used to compute the yield estimate.

Specifying Component Parameters for Monte Carlo Analysis

The procedure for specifying components for Monte Carlo analysis is the same as described in [Specifying Component Parameters for Yield Analysis](#).

Placing an Appropriate Simulation Control Component for Monte Carlo Analysis

The procedure for placing an appropriate simulation control component for a Monte Carlo analysis is the same as described in [Placing an Appropriate Simulation Control Component for Yield Analysis](#).

Setting Up a Yield Specification for Monte Carlo Analysis

Monte Carlo analysis can use the yield specification component to specify the desired performance. The procedure for setting up a yield specification for a Monte Carlo analysis is the same as described in [Setting Up a Yield Specification](#)

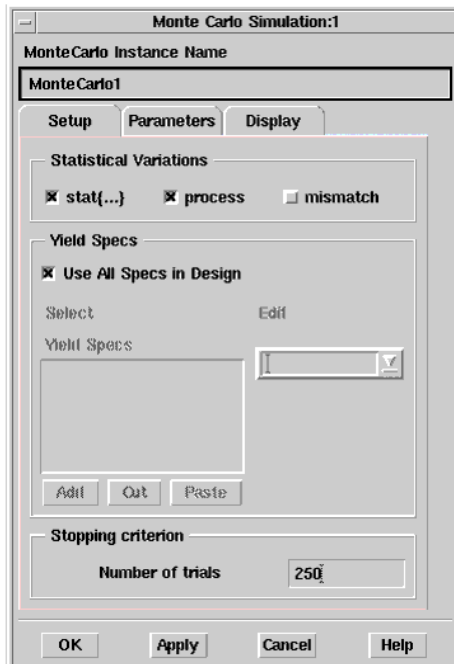
Setting Job Parameters for Monte Carlo Analysis

This three-tabbed dialog box appears when you place and double-click a Monte Carlo controller component (labeled *MC*) located in the *Optim/Stat/Yield/DOE* library or palette for Analog/RF Systems and the *Controllers* library or palette for Signal Processing). Do the following:

1. Place the *Monte Carlo (MC)* controller component in the appropriate design window.
2. Double-click the component to bring up the dialog box. The Setup tab is active.
3. Make specifications in each tab (Setup, Parameters, and Display) of the dialog box, as described in the next sections.

Selecting a Yield Specification for Monte Carlo Analysis

To set up a Monte Carlo analysis in the Setup tab of the Monte Carlo Simulation dialog box:



1. The options in the Statistical Variations box support two different statistical models:
 - ADS traditional model - The statistical variables use ADS syntax and are set up as `R=50 stat{gauss +/-1%}`.
 - Spectre statistical model - The variations *process* and *mismatch* use Spectre syntax and are provided only for use in ADS Dynamic Link. They are specified in the Statistics block in the foundry files, and define the process variations and mismatch variations.

When choosing options in the Statistical Variations box:

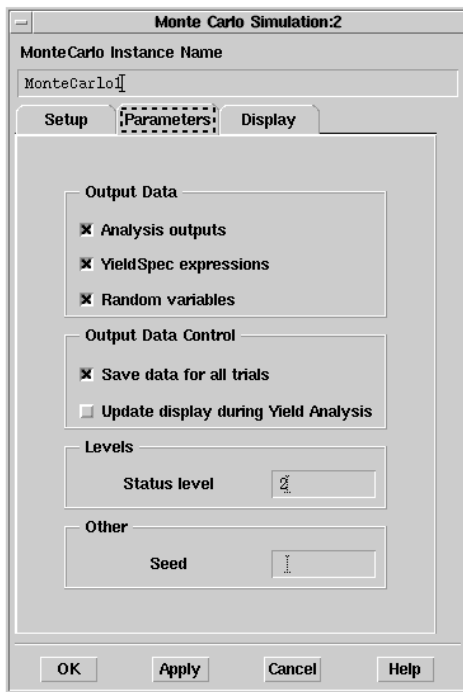
- `_stat_{...}` refers to the statistical variables defined in ADS traditional model. Choose `_stat_{...}` to set the ADS syntax for statistical variations.
- *process* and *mismatch* refer to the statistical variables defined in a Spectre

model.

- If there are no variables set, checking any option will not affect the simulation. For example, the ADS traditional model uses only `stat{...}` variables. Selecting *process* or *mismatch* will not change the results.
2. In the Yield Specs box, accept the default *Use All Specs in Design* checkbox. This is the best approach for most designs, and all Yield Spec components placed in a design will be implicitly associated with the Monte Carlo controller.
To associate a *subset* of all Specs with a given Monte Carlo controller, deselect the *Use All Specs in Design* checkbox. Select a Yield Spec from the Edit drop-down list, which will include all yield specification components that are currently placed in the design, as described in the section, [Setting Up a Yield Specification](#). Choose *Add* to place in the *Yield Specs* box, and repeat this procedure if necessary. Choose the *Cut* or *Paste* buttons, if necessary to make any changes in the *YieldSpecs* box.
 3. Under *Stopping criterion*, specify the number of desired trials to use during the Monte Carlo process.
 4. Choose **Apply** to retain the specifications that you have made while you enter data into the Parameters tab, as described in the next section.

Setting Parameter Information for Monte Carlo Analysis

To set parameter information, such as what data to save and parameter attributes, in the Parameters tab of the Monte Carlo Simulation dialog box:



1. In the Output Data field, specify which data you want to retain in your dataset following Monte Carlo analysis. Check the following choices that apply.
 - *Analysis outputs* sends all measurements (including measurement equations) to the dataset. This can create a substantial amount of data.
 - *YieldSpec expressions* (default) sends the result of each active Yield Spec's *Expr* field to the dataset.
 - *Random variables* sends the values of all random variables to the dataset.
2. In the Output Data Control field, specify whether you want to:
 - *Save data for all trials*. Data for all trials is saved. This can create a substantial amount of data.
 - *Update display during Yield Analysis* (default). This updates the dataset on each Monte Carlo trial so you can see the results in the Data Display window as they occur (instead of waiting to the end where all the traces are displayed at once).
3. In the *Levels* field, enter a number for the desired annotation level. Levels are 0-4 (default is 2), with increasing information displayed in the Status window.
4. In the *Other* field, specify a seed value. *Seed* is a value for the random number generator used to initiate a Monte Carlo analysis. If *Seed* is not specified, the simulator chooses its own seed, which will be different each time a Monte Carlo analysis is performed.
5. Choose **Apply** to retain the specifications that you have made.

Displaying Monte Carlo Analysis Data on the Schematic

Selecting the Monte Carlo analysis parameters that will be displayed on your schematic is done the same way as in nominal (performance) optimization by choosing the Display tab. Refer to *Displaying Analysis Data on the Schematic* (optstat) for details. Below is a Monte Carlo Simulation dialog box example.

!optstat-4-1-16.gif!

Note
 Monte Carlo analysis can be used without the yield specification component. In this case, the *SimInstanceName* is used to specify the underlying simulation component defined in [Placing an Appropriate Simulation Control Component for Monte Carlo Analysis](#).

Several Monte Carlo Simulation Display Parameters

Display parameter	Description
SimInstanceName	Simulation Instance Name
NumIters	Number of Iterations
MaxTrials	Maximum number of Trials
Seed	Seed
PerformNomSim	Obsolete Parameter
SaveSolns	Save Solutions
SaveSpecs	Save Specifications
SaveRandVars	Save Results and Variables
UpdateDataset	Update Dataset
SaveAllIterations	Save All Iterations
UseAllSpecs	Use All Specifications
YieldSpecName	Yield Specification Name
StatusLevel	Status Level
Enable	Enable
RestoreNomValues	Restore Nominal Values
StartTrial †	Start Trial Number (default = 0)
StopTrial † †	Stop Trial Number (default = NumTrials).

† StartTrial and StopTrial can be specified so that the Monte Carlo Analysis will run from StartTrial to StopTrial. For more information, refer to [Using StartTrial and StopTrial](#)

† The Number of Trials set in the Stopping Criterion section of the Setup tab is the default value; however, a -1 is displayed on the schematic to indicate that NumTrials will be used.

When you have finished setting up all the tabs in the Monte Carlo Simulation dialog box, click **OK** .

Using StartTrial and StopTrial

Two control parameters can be used to specify a Monte Carlo or Yield Analysis run-start and stop, *StartTrial* and *StopTrial* . These two parameters work together with *NumTrials* to force a Monte Carlo/Yield Analysis to run from StartTrial to StopTrial.

- *NumTrials* (required parameter) will control the number of trials for Monte Carlo or Yield Analysis. The number of trials is measured from 1 to N.
- *StartTrial* (optional parameter) will control the start trial number for Monte Carlo or Yield Analysis. The default StartTrial value is 1. If StartTrial is not specified or is less than zero, StartTrial will assume the default value of 1.
- *StopTrial* (optional parameter) will control the stop trial number for Monte Carlo or Yield Analysis. If StopTrial is not specified or is less than zero, it will assume the value of NumTrials. If StopTrial is specified more than NumTrials, StopTrial will reset to the value of NumTrials internally. The only valid values for StopTrail fall in the range $0 < StopTrial < NumTrials$.

These optional parameters are used for specific cases. IC designers occasionally want to debug a design in (seeded) Monte Carlo simulations. For example, suppose a designer makes 100 runs of a design. The 40th, 53rd, and 54th runs generate out-lier performance. The designer then corrects the design and wants to re-run these three cases. In this case, the designer could set StartTrial to 40 and StopTrial to 54.

Initiating Monte Carlo Analysis

To initiate a Monte Carlo analysis, select **Simulate** or click the **Simulate** button on the toolbar. The analysis status is displayed in the Status window. Upon completion of the analysis, the simulator ceases analysis and indicates success.

Note

If the Monte Carlo analysis process becomes exceedingly long, you can use the Stop Simulation command on the Simulation/Synthesis menu in the status window to interrupt the process.

Performing Yield Optimization

Yield optimization adjusts nominal values of selected element parameters to maximize yield. Also referred to as design centering, yield optimization is the process in which the nominal values of yield variables are adjusted to maximize the yield estimate.

When you activate yield optimization, you are required to enter the number of design iterations. This is the number of yield improvements you wish the simulator to obtain. Each design iteration may require several yield analyses (yield estimations).

You are not required to enter the number of trials to be used for each yield analysis. The number of trials is a dynamic variable computed during yield optimization, varying with changing yield estimates and confidence levels. Therefore, the yield estimate derived from yield optimization often differs from that for a single yield analysis with a user-specified number of trials.

To have control over the confidence level and hence the accuracy of the yield estimate, it is recommended that you perform a yield analysis after the yield optimization is completed, using the nominal parameter values obtained from the yield optimization. Choose an appropriate number of trials based upon the following formula, where N is the number of trials.

For a 95.4% confidence level ($C_{\sigma} = 2$), an Error = $\pm 2\%$ and a yield of 80%

$$N = \left(\frac{2}{0.02}\right)^2 \cdot 0.8 \cdot (1 - 0.8)$$

$N = 1600$ trials

For more information, refer to *Monte Carlo Trials and Confidence Levels* (optstat).

Setting Job Parameters for Yield Optimization

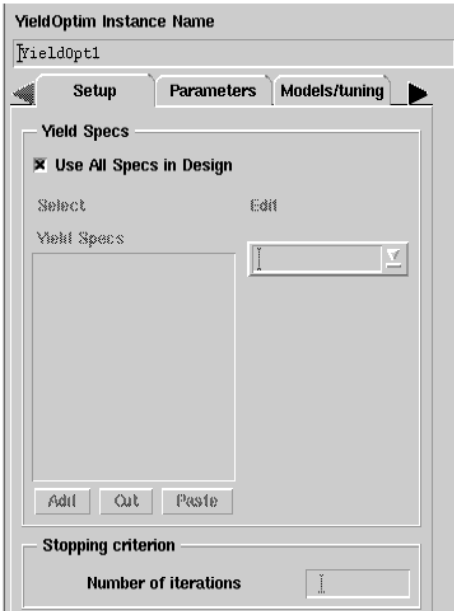
To set job parameters, you need to specify appropriate data in the Yield Optimization dialog box.

This four-tabbed dialog box appears when you place and double-click a Yield Optimization component (labeled *YldOpt*) located in the *Optim/Stat/Yield/DOE* library or palette for Analog/RF Systems and the *Controllers* library or palette for Signal Processing). Do the following:

1. Place the *Yield Optimization* (*YldOpt*) component in the appropriate design window.
2. Double-click the component to bring up the dialog box. The Setup tab is active.
3. Make specifications in each tab (Setup, Parameters, Models/tuning, and Display) of the dialog box, as described in the next sections.

Selecting a Specification for Yield Optimization

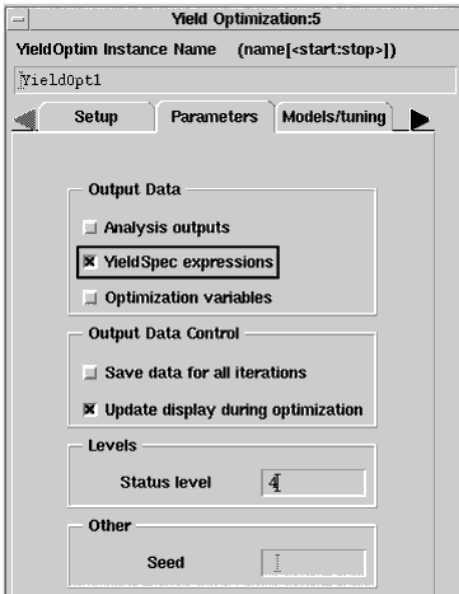
To set up a yield optimization in the Setup tab of the Yield Optimization dialog box:



1. In the Yield Specs box, accept the default *Use All Specs in Design* checkbox. This is the best approach for most designs, and all Yield Spec components placed in a design will be implicitly associated with the Yield Optimization controller.
To associate a *subset* of all Specs with a given Yield Optimization controller, deselect the *Use All Specs in Design* checkbox. Select a Yield Spec from the Edit drop-down list, which will include all yield specification components that are currently placed in the design, as described in the section, [Setting Up a Yield Specification](#). Choose *Add* to place in the *Yield Specs* box, and repeat this procedure if necessary. Choose the *Cut* or *Paste* buttons, if necessary to make any changes in the *YieldSpecs* box.
2. Under *Stopping criterion*, specify the number of desired iterations to use during the yield optimization process.
3. Choose **Apply** to retain the specifications that you have made while you enter data into the Parameters tab, as described in the next section.

Setting Parameter Information for Yield Optimization

To set parameter information, such as what data to save and parameter attributes, in the Parameters tab of the Yield Optimization dialog box:



1. In the Output Data field, specify which data you want to retain in your dataset following yield optimization. Check the following choices that apply.
 - *Analysis outputs* sends all measurements (including measurement equations) to the dataset for each yield optimization iteration. This can create a substantial amount of data.

- *YieldSpec expressions* (default) sends the result of each active Yield Spec's *Expr* field to the dataset.
 - *Optimization variables* sends the values of all optimization variables to the dataset for each improvement found during the yield optimization.
2. In the Output Data Control field, specify whether you want to:
- *Save data for all iterations* . Data for all iterations is saved. This can create a substantial amount of data.

Note
For yield optimization, enabling this feature can slow the simulation time considerably when many iterations are being run. The default is off, where only the first and last iterations are saved to the dataset.

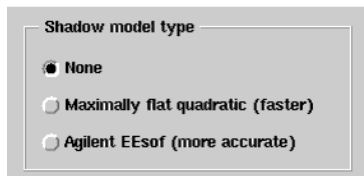
- *Update display during optimization* (default). This updates the dataset on each optimization iteration so you can see the results in the Data Display window as they occur (instead of waiting to the end where all the traces are displayed at once).
3. In the *Levels* field, enter a number for the desired annotation level. Levels are 0-4 (default is 4), with increasing information displayed in the Status window.
4. In the *Other* field, specify a seed value for use with the random optimizer. *Seed* is a value for the random number generator used to initiate an optimization. If *Seed* is not specified, the simulator chooses its own seed, which will be different each time a yield analysis or yield optimization is performed.
5. Choose **Apply** to retain the specifications that you have made while you enter data into the Models/tuning tab, as described in the next section.

Selecting a Shadow Model Type for Yield Optimization

You use the Models/tuning tab of the Yield Optimization dialog box to select the Shadow Model, an optional method of yield optimization.

The Shadow Model works as follows: A series of trials is run in which the random variations in your design's statistical parameters are used in a mathematical model of the design's performance to compute the yield. This allows a greater number of trials and therefore greater accuracy in the yield estimate without a significant increase in the computation time required.

To enable one of two available methods of Shadow Model analysis:



Click the selection of your choice, using either of two methods:

- Maximally flat quadratic Shadow Model (which is usually faster)
- Agilent EEsof Shadow Model (which is usually more accurate)

If **None** is selected, the Monte Carlo method will be applied to the simulator, not to the mathematical Shadow Model.

Consult the following references for details concerning the Monte Carlo method and the Maximally Flat Quadratic Approximation model.

- R. Spence and R. S. Sooin. *Tolerance Design of Electronic Circuits*, Addison-Wesley, 1988.
- Radoslaw (Radek) Biernacki, John Bandler, Jian Song, and QI-Jun Zhang. *Efficient Quadratic Approximation for Statistical Design* , IEEE Transactions on Circuits and Systems, vol. 36, No. 11, November 1989.

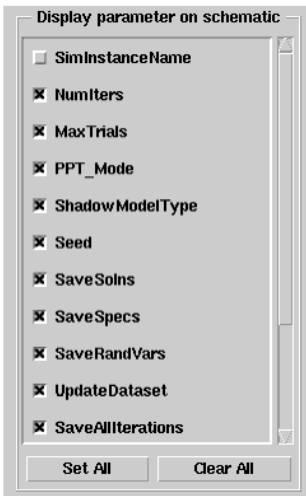
Enabling Post Production Tuning

Post Production Tuning can be used with yield optimization just as in yield analysis. The feature is enabled in the lower part of the Models/tuning tab of the Yield Optimization dialog box. Refer to [Enabling Post Production Tuning](#) for a description of this feature.

Choose **Apply** to retain the specifications that you have made while you enter data into the Display tab of the Yield Optimization dialog box, as described in the next section.

Displaying Analysis Data on the Schematic

Selecting the yield optimization parameters that will be displayed on your schematic is done the same way as in nominal (performance) optimization by choosing the Display tab. Refer to *Displaying Analysis Data on the Schematic* (optstat) for details. Below is a yield optimization dialog box example.



When you have finished setting up all the tabs in the Yield Optimization dialog box, click **OK**.

Swept Yield Optimization

Yield Optimization can be swept as any other ADS analysis. When the Yield Optimization controller is referenced by a parameter sweep controller, the yield optimization is performed for each value of the sweep variable and the results output as a function of the sweep variable. For more information, refer to *Swept Optimization* (optstat).

Yield Optimization Example

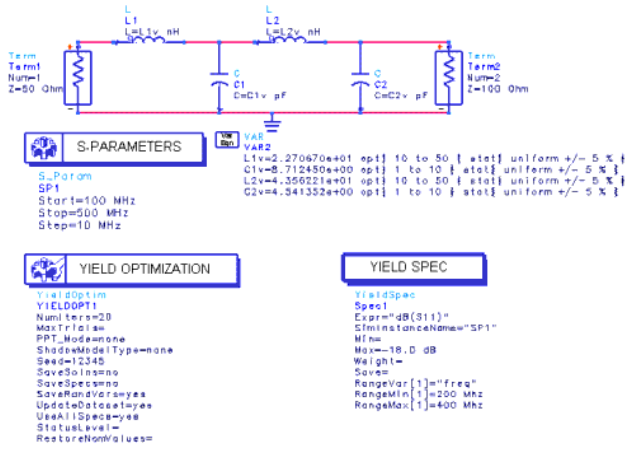
This example demonstrates how to optimize the yield of the same octave-based, 2-to-1 matching transformer that was used in the example at the beginning of this topic. Refer to the section [Yield Analysis Example](#).

This example is called *ylDoptex1_wrk*, and it is located in the directory `$HPEESOF_DIR/examples/Tutorial`. To access this example workspace and enable simulation, open the example by choosing **File > Open > Example** from the Main window.

In the Yield Analysis example, the network was optimized to have an equal-ripple response. The yield of the equal-ripple design, assuming $\pm 5\%$ tolerance on the inductances and capacitances and a minimum return loss of 18 dB, was approximately 77%. This figure is based on using 1000 trials and therefore has a margin or error between ± 2 and $\pm 3\%$ at a confidence level of 95%.

To optimize the yield of the transformer:

1. From the Schematic window, select **File > Open** to open the example *ylDoptex0*, which uses the schematic design from the *ylDex0* example (shown in the section [Yield Analysis Example](#)), except that the *YldOpt* control component is used instead of the *Yield* component. The optimization range for the nominal parameter is identical to that used in the example in the section, *Optimization Examples* (optstat) in *Nominal Optimization* (optstat). Twenty iterations are specified on the *YldOpt* control component. The initial design is shown in the following figure. The initial yield analysis, as noted in the section [Yield Analysis Example](#), indicated a yield of about 77%.



Initial Design Prior to Yield Optimization

2. Select **Simulate > Simulate** or click the tool bar **Simulate** button. The analysis status in the status window indicates progress by displaying the current iteration number and yield estimate, along with the current optimal nominal component values, as shown in the following figure. Use the scroll bar to view more iterations and variables.

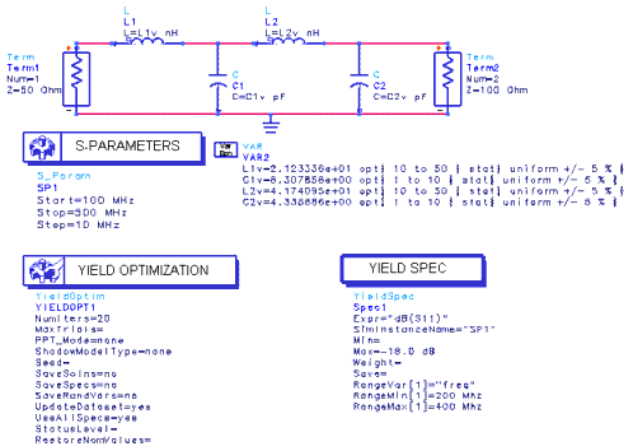
Status / Summary

Iteration #20:
Current Yield(%) : 90.5405405405405
Optimization variables:
 C2v = 4.3633954637501
 L2v = 42.6195669713859
 C1v = 8.4607016963362

Progress of Yield Optimization

Upon completion of the simulation, the optimized yield should be more than 90%, indicating less than half as many failures as the equal ripple design.

1. Select **Simulate > Update Optimization Values** to change the nominal values on the schematic to the optimized ones. The results are shown in the following figure.



Schematic Design with Optimized Values

If we look at the VarEqn component, we see that the optimized values have been updated on the schematic. !optstat-4-1-26.gif!

Note
 The values that you obtain if you run through the example may differ slightly from those shown here due to the random nature of the yield optimization algorithm.

The example shows that an equal-ripple design, having equal distance between the specification and the actual response at both ends and the middle of the frequency band is not optimal when component variations are taken into account. The yield of such a design can be improved dramatically merely by shifting the nominal value by a small amount using yield optimization.

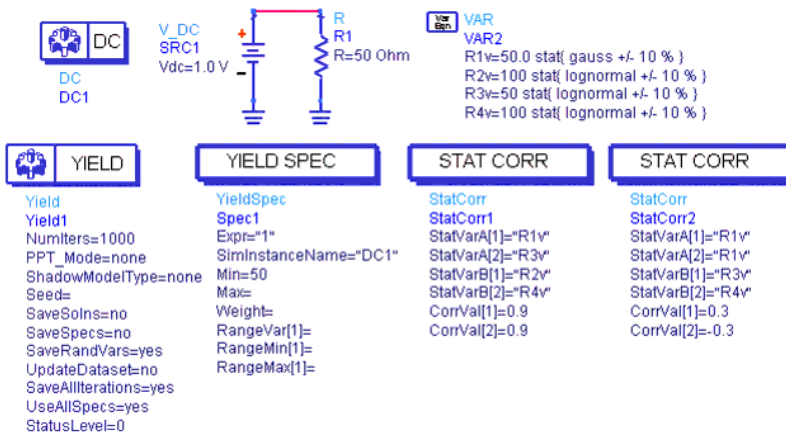
Statistical Correlation

The Statistical Correlation component (*StatCorr*) is used in yield analysis and yield optimization. The schematic shown in the following figure illustrates the use of this feature.

Any statistical variable can be correlated to another statistical variable regardless of the distribution of the variables. The correlation coefficient parameter *CorrVal* should be in the range of x , where $-1.0 < x < 1.0$. If for a system of correlated variables, the correlation matrix is not *positive definite* , a warning message will be displayed and the correlation values altered so that the analysis can proceed.

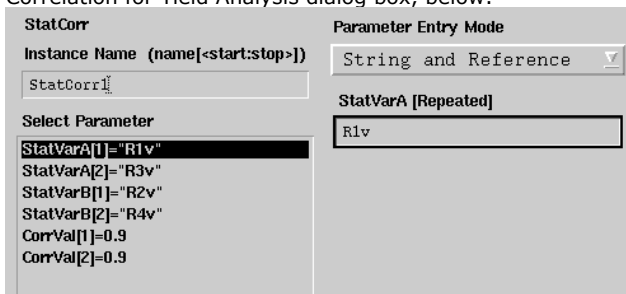
In this example, a "dummy" yield analysis is used to generate the random outcomes for R1v, R2v, R3v, and R4v. The Statistical Correlation component *StatCorr1* is used to correlate R1v with R2v and R3v with R4v at a level of 0.9, which is a strong positive relationship.

StatVarA[1]="R1v"	StatVarB[1]="R2v"	CorrVal[1]=0.9
StatVarA[2]="R3v"	StatVarB[2]="R4v"	CorrVal[2]=0.9

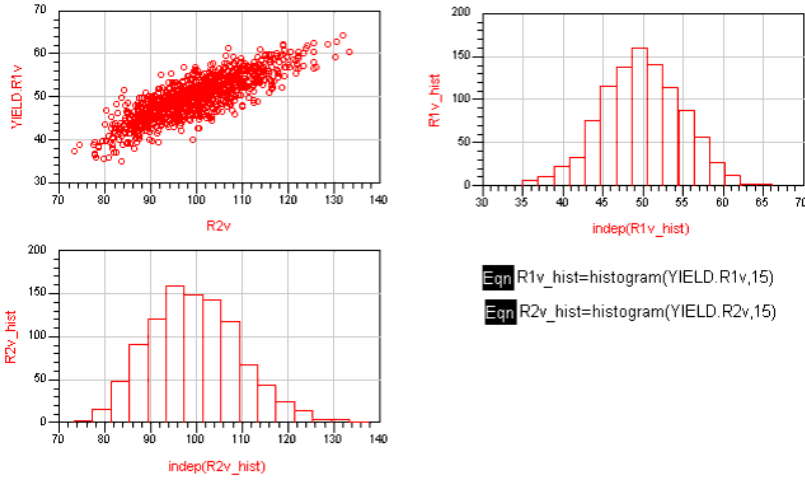


simpleCorr Schematic

The *StatCorr1* component has only a few parameters as shown in the Statistical Correlation for Yield Analysis dialog box, below:



The following figure shows a scatter plot and histograms of both random variables R1v and R2v.



Scatter Plot and Histograms

The data below shows the statistical correlation values for R4v, R3v, R2v, and R1v.

corrStatVarName	originalCorr.R4v	originalCorr.R3v	originalCorr.R2v	originalCorr.R1v
R4v	1.000	0.900	0.000	-0.300
R3v	0.900	1.000	0.000	0.300
R2v	0.000	0.000	1.000	0.900
R1v	-0.300	0.300	0.900	1.000

corrStatVarName	updatedCorr.R4v	updatedCorr.R3v	updatedCorr.R2v	updatedCorr.R1v
R4v	1.000	0.820	-0.063	-0.214
R3v	0.820	1.000	0.063	0.214
R2v	-0.063	0.063	1.000	0.833
R1v	-0.214	0.214	0.833	1.000

Using Design of Experiments (DOE)

Design of experiments (commonly referred to as DOE) is a data-driven technique for robust design. In the early 1900's, DOE was used by agricultural engineers to improve crop yields. Today circuit and system designers are applying the method as a means to the same end-yield improvement.

A typical DOE includes three primary steps:

1. Plan the experiment:
 - Assess the experimental resource budget.
 - Identify the input and response variables.
 - Assign levels (values) to input variables.
2. Perform the experiment and collect response data.
3. Analyze the data using statistical methods.

Sequential application of this methodology can be used to improve the statistical performance of a given circuit or system. Because of an inherent compromise between statistical performance prediction accuracy and the number of input variables, a *screening* experiment is used to identify variables that contribute significantly to performance variation. Next a *refining* experiment can be used to increase focus on the target statistical response.

DOE and Computer Simulation

In a general application, DOE methods are designed to accommodate errors of the type found in any experiment. But because circuit and system simulators provide identical results for any analysis having the same input values, complexity in setting up, performing, and analyzing experiments is reduced.

Since the computer is being used to perform the experiment, a more complete characterization of input/output relationships can be realized. Finally, since the computer handles the tedious tasks of bookkeeping during the experiment, there is a further reduction in the possibility of human error.

The primary purpose of DOE is to characterize an unknown process. In circuit or system simulation, the unknown process is predicting the response of the design under test (DUT). A simple technique for characterizing a DUT is to perturb each input variable (*factor*) in turn, and to record the resulting output response. However this approach breaks down if the response due to a change in one factor depends on the value of a different factor.

Minimum DOE Requirements

Prior to performing a Design of Experiments, you need:

- At least two (2) component parameters in your design identified as a DOE variables. You specify details in the *Component Parameter* dialog by clicking the **Tune/Opt/Stat/DOE Setup** button.
- At least one DOE Goal component specified, then placed in the design window.
- At least one Design of Experiments (DOE) Simulation component specified, then placed in the design window.
- One simulation analysis control component (for example, an AC, DC, S-Parameter, Harmonic Balance, Circuit Envelope, or Transient component for Analog/RF Systems).

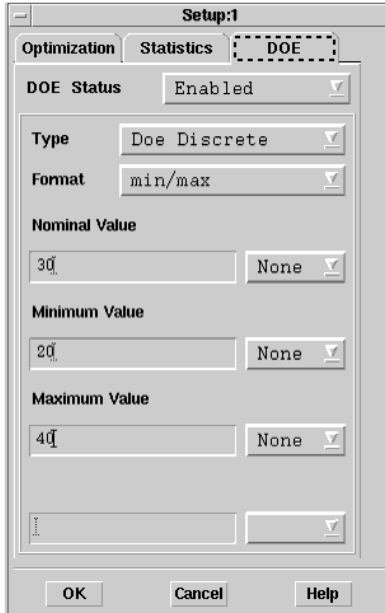
The design components needed for DOE are located in the *Optim/Stat/Yield/DOE* library or palette. Remember, you need at least two (2) DOE parameters. Otherwise, if you modify /examples/Tutorial/dae2_wrk...dae2 to have only one DOE variable, say the "A" variable set with doe enabled ("B" + "C" are set to nodoe...), then ADS 2008 will generate the following error.

```
Warning detected by hpeesofsim during Design of Experiments `DOE1'.
The minimum number for DOE factors must be larger than 2.
Warning detected by hpeesofsim during Design of Experiments `DOE1'.
DOE1 terminated due to bad number of variables.
Warning detected by hpeesofsim in device `R3' during Design of Experiments
`DOE1'.
Resistance cannot be set to zero.
Error detected by hpeesofsim during Design of Experiments `DOE1'.
Cannot set `C' to 0.
```

Specifying Component Parameters for DOE

The procedure for specifying components for DOE is as follows:

1. Select and place an appropriate component from one of the component palettes or component libraries. For example, place a parallel resistor-inductor-capacitor (PRLC) from the *Lumped Components* palette.
2. Double-click on the component in the design window to access its associated dialog box.
3. From the dialog box, highlight the parameter that you want to vary in the *Select Parameters* box (for example R for parallel resistance), then click **Tune/Opt/Stat/DOE Setup**, which will only appear for valid DOE parameters. The *Setup* dialog appears, with the *Optimization* tab active. Click the **DOE** tab.



4. From the DOE Status drop-down list, select **Enabled** so that you can set specification of the appropriate fields. *Enabled* causes the parameter to be varied when the simulation is run. *Disabled* allows you to temporarily suspend any parameter variation previously assigned, and *Clear* removes the values you previously applied to the design.
5. In the Type drop-down list, accept the default DOE Value Type of **DOE Discrete**.
6. From the Format drop-down list, select an appropriate statistical value format:

min/max
+/-Delta %
+/-Delta

For complete descriptions of the available format, refer to the section *Value Types for DOE* (optstat).

7. If you selected *+/-Delta* or *+/-Delta %* formats, specify the deviation value. For these formats, the units can also be specified in the drop-down list next to each input field.
8. If you selected a *min/max* format, you can optionally enter values for nominal, minimum, and maximum in the appropriate boxes, and select an appropriate unit assignment for each from the drop-down list next to the boxes.

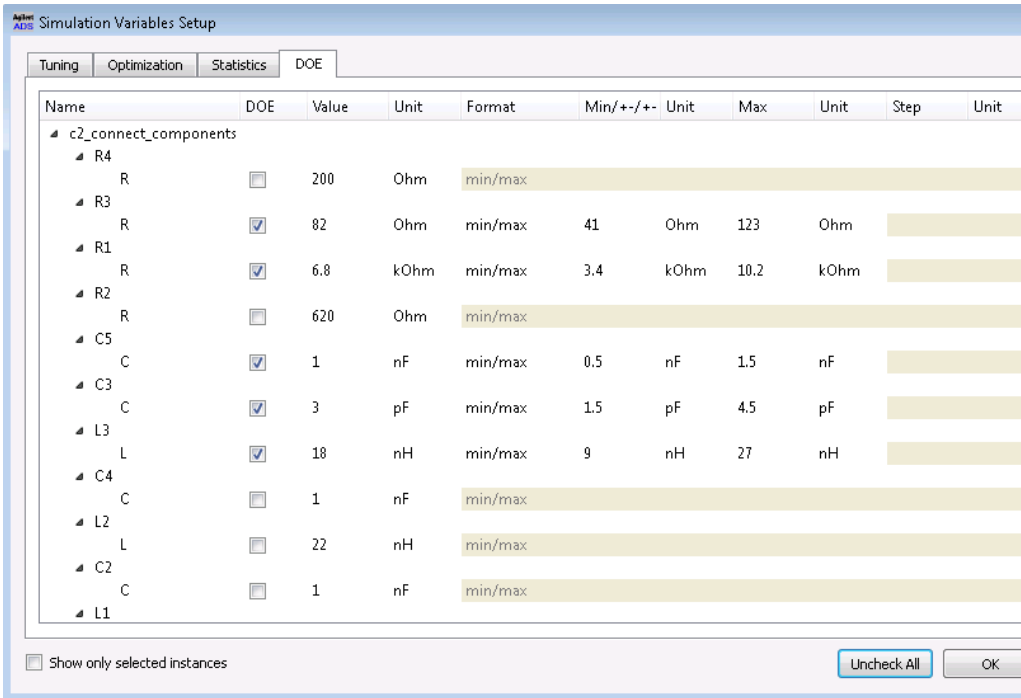
Note
Unit specification via the Setup dialog box is not possible for variables defined in the *Var/Eqn* component.

9. From the Nominal Value field and the Units drop-down list, the value and units in your design for this component are displayed. You can change these if you wish.
10. Choose **OK**.

Specifying Multiple Component Parameters for DOE

The *Simulation Variables Setup* dialog lists all the component parameters in a design while allowing the simultaneous adjustment of a number of parameters across multiple components on a single schematic.

To view the dialog, in the Schematic window, click **Simulate > Simulation Variables Setup** and click the **DOE** tab.



To specify components for DOE analysis, follow these steps:

1. Select the checkbox in the *DOE* column to enable DOE analysis for a particular component.
2. In the *Format* drop-down list, select the appropriate DOE analysis format: *Max/Min*, *+/- Delta* or *+/- Delta %*.
The default values for *Min*, *Max*, and *Step* are displayed as appropriate.
3. To modify a *Min*, *Max*, or *Step* value, enter the desired value in the appropriate field.
The default values for *Min*, *Max*, and *Step* are 50%, 150% and 10% of the nominal value of the parameter, respectively.
4. To disable DOE analysis, deselect the checkbox in the *DOE* column.
5. Select the *Show only selected instances* checkbox to display only the selected components in the schematic.
6. Click **Uncheck All** to deselect all parameters.
7. Click **OK** to close the dialog.

Placing a Simulation Control Component for DOE

An appropriate simulation control component must be placed in the design prior to initiating a DOE analysis.

For Analog/RF Systems simulation, all analysis types are supported, for example place one of the following components:

- **AC** from the AC Simulation palette or library
- **DC** from the DC Simulation palette or library
- **S-Param** from the S-Param Simulation palette or library
- **Harmonic Balance** from the HB Simulation palette or library
- **ENV** from the Envelope Simulation palette or library
- **Tran** from the Transient Simulation library

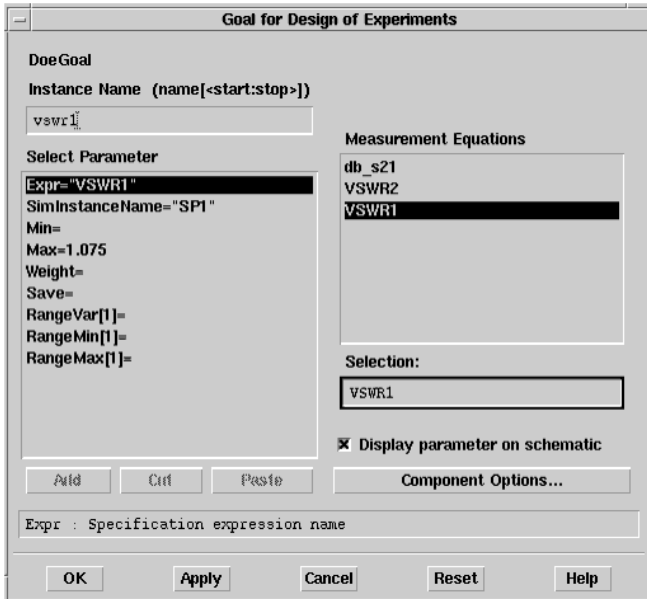
For details on specifying parameters for each of these control DOE components, *Using Circuit Simulators* (cktsim).

Setting DOE Goals

DOE goals are specified by placing a *DOE Goal* component and double-clicking it to display the *Goals for DOE* dialog. The Goal component can be found as follows:

- For Analog/RF Systems simulation, from the *Optim/Stat/Yield/DOE* palette or library
- For Signal Processing simulation, from the *Controllers* palette or library

You can specify and place more than one Goal if needed. The goals to be used are referenced by the DOE component, as described in the later section, [Setting Job Parameters for DOE](#). By default, all goals placed apply to all DOE components in a design.



To set appropriate goal specifications in this dialog box:

1. If desired, enter a name in the Instance Name field that is different from the assigned default name shown.
2. In the Select Parameter list box on the left, click on each parameter that you want to modify, then make other associated changes in the box on the right. When you select a parameter, such as Expr, all relevant items in your design will be displayed in the box. The style of this box varies depending on the parameter, as described in the table below.

Parameter Goals for Nominal Optimization

Parameter	Description	Use Model
Expr	A valid AEL expression that operates on the simulation results, such as mag(S11), or the name of a MeasEqn. For more information on AEL expressions, refer to <i>AEL (ael)</i> or <i>Measurement Expressions (expmeas)</i> .	The list box label becomes Measurement Equations. All associated expressions are displayed in the box. Select the one you want to analyze and it will appear just below in the Selection box. For expressions not related to MeasEqns, you must type them in the Selection box.
SimInstanceName	Enter the instance name for the simulation control component that you placed in your design, which will generate the data used by the Expr field.	The list box label becomes Analysis Components. Select the analysis component (simulation controller), such as S-parameter, that you want to analyze and it will appear just below in the Selection box.
Min	Enter a number for a minimum acceptable response value.	Fields for Parameter Entry Mode and Equation editor are used as in any component parameter dialog box. Type a value in the box. Note: Both Min and Max do not have to be specified, but at least one does.
Max	Enter a number for a maximum acceptable response value.	Same as above.
Weight	Enter a weighting value to be used in error function calculation. Default is 1. For more information on using the weighting factor to form the error function, refer to <i>Weighting Factors (optstat)</i> .	Fields for Parameter Entry Mode and Equation editor are used as in any component parameter dialog box. Type a value in the box.
RangeVar	Independent variable name.	Same as above, but note that this parameter is "indexable" and can be applied to more than one independent variable.
RangeMin	Minimum limit of range for independent variable during optimization.	Same as above.
RangeMax	Maximum limit of range for independent variable during optimization.	Same as above.

Setting Job Parameters for DOE

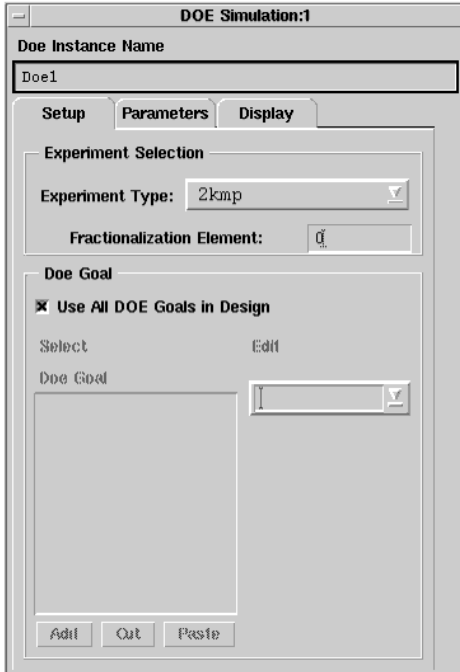
To set job parameters, you need to specify appropriate data in the DOE Simulation dialog box.

This three-tabbed dialog appears when you place a DOE Simulation component (labeled *DOE*). Do the following:

1. Place the *DOE* component in the appropriate Schematic window.
2. Double-click the component to bring up the dialog box. The Setup tab is active.
3. Make specifications in each tab (Setup, Parameters, and Display) of the dialog box, as described in the next sections.

Selecting a DOE Specification

First select the Setup tab of the DOE dialog box to set up a DOE analysis.



1. In the Experiment Selection box, select the desired Experiment Type from the drop-down list. Refer to [DOE Concepts](#) for more information on DOE theory and experiment types. The available types are as follows:

Available DOE Experiment Types

Experiment Type	Description
2kmp	2 raised to the power of k minus p, where k is the number of factors and p is the <i>fractionalization</i> element. When $p = 0$, a full factorial experiment is identified.
Plackett-Burman	Allows the study of $k=N-1$ variables in N runs, where N is a multiple of 4.
CCD	Combines a 2-level experiment with the center point and <i>star</i> points along the coordinate axis. Star points lie outside the 2-level experiment and their distance from the center point is a function of the number of factors, i.e., $d=2^{(k/4)^{1/2}}$
Box-Behnken	Consists of the zero point (nominal) and a 2-level, 2-factor factorial design for all combinations of factors, while holding other factors at their nominal value.
3k	A 3-level full factorial experiment.

1. If you select the 2kmp method, enter a *Fractionalization* element in the Fractionalization Element field.
2. In the DOE Goal box, accept the default *Use All DOE Goals in Design* checkbox. This is the best approach for most designs, and all DOE components placed in a design will be implicitly associated with the DOE Goal component.
To associate a *subset* of all DOE Goals with a given DOE analysis controller, deselect the *Use All DOE Goals in Design* checkbox. Select a DOE spec from the Edit drop-down list, which will include all DOE components that are currently placed in the design. This step is similar to the same procedure for Yield, as described in the section, *Setting Up a Yield Specification* (optstat). Choose *Add* to place in the *DOE Goal* box, and repeat this procedure if necessary. Choose the *Cut* or *Paste* buttons, if necessary to make any changes in the *DOE Goal* box.
3. Choose **Apply** to retain the specifications that you have made while you enter data into the Parameters tab, as described in the next section.

Setting Parameter Information

You set parameter information in the Parameters tab of the DOE Simulation dialog box, such as what data to save and when the data is output.

During DOE analysis, a complete set of DOE outputs (Pareto, Effects, and Interactions diagrams) are implicitly generated for each DOE Goal component. In addition to the implicitly generated outputs, an ASCII file of the experiment results is created for each DOE analysis component. This file is stored in the / *data* subdirectory. You can use this file to input your DOE results into third-party spreadsheet or statistical analysis programs.

To do this, follow these steps:

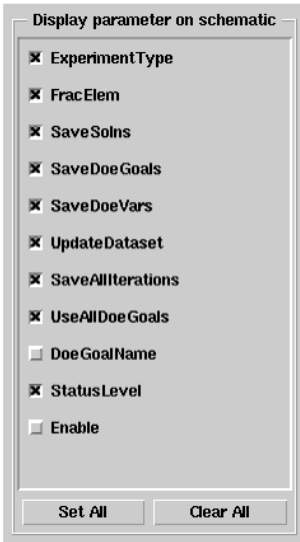
1. In the *Output Data* field, specify which data you want to retain in your dataset following DOE analysis.
 - *Analysis outputs* sends all measurements (including measurement equations) to the dataset for each trial. This can create a substantial amount of data.
 - *DOE Goals* sends the result of each Goal's *Expr* field to the dataset for each trial.
 - *DOE Experiment variables* sends the values of all DOE experiment variables to the dataset for each trial.
2. In the *Output Data Control* field, specify whether you want to:
 - *Save data for all treatment combinations*. Data for all treatment combinations is saved. This can create a substantial amount of data.

Note
 For DOE experiments, enabling this feature can slow the analysis time considerably when the experiment is large. The default is off where only the first and last treatment combinations are saved to the dataset.

- *Update display after each treatment combination* updates the dataset on each DOE treatment combination so you can see the results in the Data Display window as they occur instead of waiting to the end where all the traces are displayed at once.
3. In the *Levels* box, enter a number for the desired annotation level in the Status level field. Levels are 0-4, with increasing information displayed in the Status window. (2 is the default.)
 4. Choose **Apply** to retain the specifications that you have made while you enter data into the Display tab, as described in the next section.

Displaying Analysis Data on the Schematic

Selecting the DOE parameters that will be displayed on your schematic is done the same way as in nominal optimization. Refer to *Displaying Analysis Data on the Schematic* (optstat) for details. Below is a DOE example.



When you have finished setting up all the tabs in the DOE Simulation dialog box, click **OK**.

Initiating Design of Experiments

To initiate a DOE analysis:

1. Choose **Simulate** or click the **Simulate** button on the toolbar. The analysis status, including information about the current treatment combination number as well as result computation progress, is displayed in the Status window. Upon completion of the analysis, the simulator ceases analysis and indicates success.

Note

If the DOE analysis process becomes exceedingly long, you can use the Stop and Release Simulator command on the Simulate menu to interrupt the process.

2. When the simulation is complete, you are ready to view the DOE output, which is available for each specified DOE goal.

Swept DOE

DOE can be swept as any other ADS analysis. When the DOE controller is referenced by a parameter sweep controller, the DOE is performed for each value of the sweep variable and the results output as a function of the sweep variable. For more information, refer to *Swept Optimization* (optstat).

DOE Terminology

Following are definitions of the most frequently used *design of experiments* (DOE) terms :

- **Design of experiments** (commonly referred to as DOE). A data-driven technique for robust product design. It is used to improve the statistical performance of a given circuit or system by predicting the response of the device-under-test (DUT).
- **Multilevel experiment**. An experiment with more than 2 levels.
- **Screening experiment**. A screening experiment is used to identify the *significant* few factors that contribute the most to response variation.
- **Refining experiment**. The refining experiment is used to more thoroughly investigate how factors affect the output response. One aim of a refining experiment might be to detect curvature in the factor/response relationship by using a multilevel experiment.
- **Factor**. An input variable.
- **Levels** . Levels represent the values that an input variable will take on during the course of an experiment. For example, for a two-level experiment, variable levels might be assigned to reflect the ± 1 standard deviation of the variable value.
- **Response**. An output response due to a particular set of factor level combinations.
- **Design units** . Usually factor levels are encoded such that the maximum and minimum physical values correspond to +1 and -1 respectively. The +1 -1 notation indicates the factor values are in design units, and are obtained from physical values using the following equation for the two-level experiment:

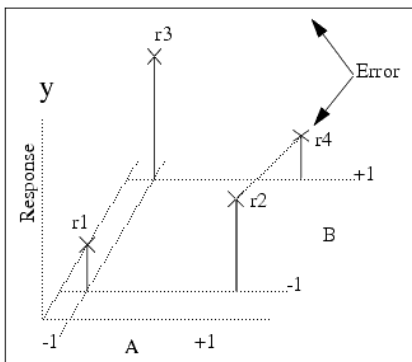
$$\frac{X - X_{mid}}{(X_{hi} - X_{lo})/2}$$

where X is the minimum (maximum) physical value of the variable, and X_{lo} , X_{mid} , and X_{hi} are the minimum, middle, and maximum physical values. For example, a capacitor value might be 100pF ±10%, leading to low, mid, and high values of 90, 100, and 110pF, respectively.

- **Interaction** . Any time the factor/response relationship changes as a function of a different factor, there is said to be an interaction between the two factors.
- **Factorial or Full Factorial experiment**. In the factorial experiment, response results are collected for *all* combinations of factor levels.
- **Fractional Factorial**. If the designer can reasonably assume that effects due to high-order interaction terms is negligible, then the information on main and low order interactions can be obtained by running a subset (fraction) of the full factorial experiment. The result is a significant reduction in the amount of work required to obtain the desired information.
- **Main effect** . The main effect for a 2-level experiment is defined as the difference in average response at the two levels of a given factor.
- **Half-effect**. This metric is simply the main effect divided by two.
- **Design matrix** . The design matrix provides a compact representation of an experiment, showing factor level combinations and associated response values tabulated in row-column format.
- **Treatment combination** . An experiment will usually have several treatment combinations (tc) where each one represents a particular set of factor level combinations. A tc is simply a row in the design matrix.
- **Orthogonal design** . There are many ways an experiment can be structured in terms of factor level combinations. If the factor level combinations are such that each column in the design matrix is linearly independent, then the design is said to be orthogonal. In short, for an orthogonal design, the total variation in the response can be decomposed into components due to each factor and interaction. This decomposition makes it possible to rank the importance of factors with respect to their contribution to total performance variance.
- **Aliasing** . The purpose of the fractional factorial experiment is to reduce the overall work required to obtain the desired information about factor/response relationships. To facilitate this reduction in work, effects due to changes in factor levels are added together (aliased) with effects from interactions between factors. As such, a fixed amount of total response variance is attributable to more than one source. Aliasing is sometimes referred to as *confounding* .
- **Saturated experiment** . A saturated experiment is one which provides for the study of $k=N-1$ variables in N runs.
- **Fractionalization component**. The fractionalization component is representative of the fraction of a full factorial experiment to be used in a given Fractional Factorial experiment. The actual fraction of the full factorial experiment is obtained using the simple formula: $(1/2)^P$, where P is the fractionalization component.
- **Pareto diagrams** . Pareto diagrams are bar charts that show the percentage of the total response variance attributable to each factor and interaction.
- **Effects plots** . Effects plots depict average response values as a function of factor level.
- **Interaction diagrams** . Interaction diagrams indicate how the change in response due to one factor changes with respect to a second factor.

DOE Concepts

The following figure shows two factors, A and B, and the associated response at various values (*levels*) of the factors.



Comparison of Responses of Factors A and B

Notice that the factors have two levels: one low (-1) and one high (+1). The ±1 notation indicates the factor values are in *design units*, and are obtained from physical values using the following equation:

$$\frac{X - X_{mid}}{(X_{hi} - X_{lo})/2}$$

where X is the minimum (maximum) physical value of the variable, and X_{lo} , X_{hi} , and X_{mid} are the minimum, middle, and maximum physical values. For example, a capacitor value might be 100pF ± 10%, leading to low, mid, and high values of 90, 100, and 110pF respectively.

If we were to note the change in response due to a change in factor A (from low to high), we would be led to believe that increasing A causes an increase in the response-the same would be observed for factor B. A model from the three response points r_1 , r_2 , and r_3 can be formulated as the plane surface which contains them:

$$y = [(r_2-r_1)/2]A + [(r_3-r_1)/2]B + BIAS$$

where the BIAS term is found by equating the response at a given factor level condition, for example, if A and B are -1, then

$$y = r_1 = \frac{r_1-r_2}{2} + \frac{r_1-r_3}{2} + BIAS$$

This leads to:

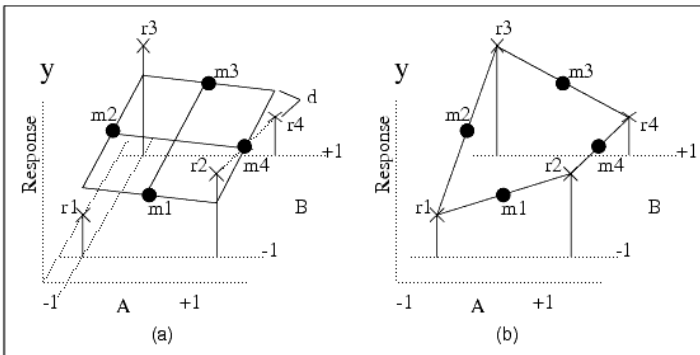
$$BIAS = \frac{r_2+r_3}{2}$$

However, if either factor A or B is held high, and the same experiment is performed, an inverse relationship exists between factor level and response. The plane surface model from the one-factor-at-a-time experiment would significantly overestimate response r_4 .

Whenever the factor-response relationship changes as a function of a different factor, there is said to be an *interaction* between the two factors.

To account for interactions, a modification to the simple *one-factor-at-a-time* experiment scheme is necessary. The *factorial* experiment is generally accepted as one of the most efficient methods for characterizing the effects of two or more factors.

In the factorial experiment, response results are collected for *all* combinations of factor levels. For 2-level factorial designs, 2^k data points must be collected for each response, where k is the number of factors. The factorial experiment not only accounts for interactions, but also is formulated using average response values as opposed to the raw ones used in the one-factor-at-a-time method. The following figure shows four response points r_1 - r_4 , as well as the orientation for the plane surface used to model the factor-response relationship.



(a) Response Points and Plane Surface Orientation, and (b) Modified Surface

The orientation is found by evenly allocating any estimation error due to factor interactions-this error is labeled as d in the figure.

The four points on the plane surface m_1 - m_4 represent the average response for the corresponding edge (i.e., $m_1 = [r_1 + r_2]/2$). Without the interaction term, the two-level factorial model equation is of the form:

$$y + S_A A + S_B B + BIAS$$

where S_A and S_B represent the slope of the average response for the given variable, i.e.,

$$S_A = [m_4-m_2]/2.$$

The BIAS term is simply the grand average of all raw response values so that

$$\text{BIAS} = \bar{y} = \frac{[r1 + r2 + r3 + r4]}{4}$$

It turns out that the two lines having endpoints m_4, m_2 , and m_3, m_1 respectively, intersect in the middle of the plane surface. You can prove this by equating responses in the center of the plane surface, as follows:

$$m_1 + S_A(1) = m_2 + S_B(1).$$

To account for the interaction, an additional term must be added to the prediction equation:

$$y = \bar{y} + S_A A + S_B B + S_{AB} AB$$

To find S_{AB} , evaluate the response at one of the response points, for example, r_1 , and solve for S_{AB} :

$$y = r_1 = \bar{y} + S_A(-1) + S_B(-1) + S_{AB}(-1)(-1)$$

so that

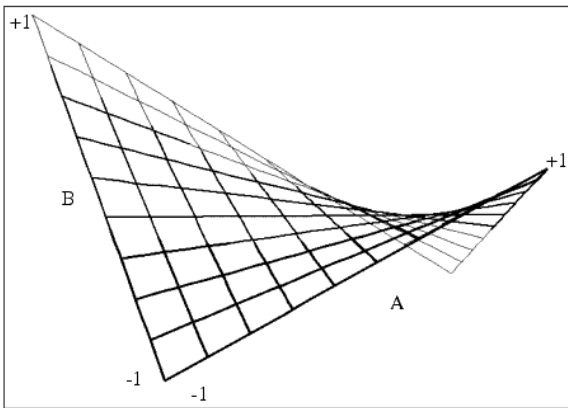
$$S_{AB} = S_A + S_B - \bar{y}$$

The solution is:

$$S_{AB} = [r_1 + r_4 - r_2 - r_3]/4$$

The response depicted in part b of the previous figure shows the modified surface.

Notice that along either factor coordinate axis, the response is linear. However, the slope of the linear model changes as a function of the other factor. For example, with $B = -1$ the response as a function of A , $y(A)$ indicates a positive slope. But as B increases, the slope of $y(A)$ decreases. Also note that the *off-axis* response contour is quadratic. The following figure shows the wire mesh plot for the new surface.



Wire Mesh Plot of the Modified Surface of the previous figure

The new prediction equation can be restated in terms related more closely to DOE:

$$y = \bar{y} + [ME_A/2]A + [ME_B/2]B + [I_{AB}/2]AB$$

where ME_A and ME_B are, in DOE parlance, the *main effect* of factor A and B respectively. I_{AB}

is referred to as the *interaction* between A and B . The main effect for a 2-level

experiment is defined as the difference in average response at the two levels of a factor.

Referring to the figure above [\(a\) Response Points and Plane Surface Orientation, and \(b\) Modified Surface](#), ME_A is simply $m_4 - m_2$. The interaction term for the same experiment is

defined as half the difference between the main effects of one factor at the two levels of a second factor.

Again using the figure [\(a\) Response Points and Plane Surface Orientation, and \(b\) Modified Surface](#), I_{AB} can be taken as half the difference in the main effect of factor A when B is

high- $ME_A (B+)$ and the main effect of factor A when B is low- $ME_A (B-)$, i.e., $I_{AB} =$

$[(r_4-r_3) - (r_2-r_1)]/2$. (For the 2-factor case, $ME_A (B+)$ and $ME_A (B-)$ are simply

differences in response values. Usually, these terms will be differences of averages.) When there are more than two factors, it is possible to define *higher order* interaction terms, such as I_{AB} C-half the difference between the two factor interaction effects at the two levels of a third factor, and so on.

There are some additional DOE terms and concepts that are easy to discuss in the context of the 2-level factorial design. First, recall that factor coefficients in the prediction equation were all divided by 1/2. Appropriately there is a DOE term called the *half-effect*, which is simply one half of the main effect. The prediction equation then becomes:
 $y = HEAA + HEBB + HEABAB$.

Design Matrix

There are many types of experiments that can be applied to any given situation. Differences in these designs are readily seen by examining the *design matrix*. The following table shows a design matrix for the 2-factor factorial experiment.

Design matrix for 2-factor factorial experiment

tc	Factors		Interactions	Response
	A	B	AB	
1	-	-	+	r1
a	+	-	-	r2
b	-	+	-	r3
ab	+	+	+	r4

Notice that columns are delineated into four main groups-tc or *treatment combination*, factors, interactions, and response. Under the tc column, a shorthand is used to indicate the unique conditions of each experiment run (trial). Lower case letters are used to indicate the factor(s) having +1 levels for the trial. (A 1 is used to indicate the run where all factors are held low.) Factor levels are designated using another shorthand where unity is implied in the symbols + and -. The levels of the interaction columns are found by taking the product of the factors involved. The response column is simply a log of the computed response.

Notice that each pair of factor/interaction columns is orthogonal, e.g., linearly independent. This construct allows independent analysis of each factor as well as interactions between factors. In short, for an *orthogonal design*, the total variation in the response can be divided into components due to each factor and interaction, thereby making it possible to rank the importance of factors. (For additional details, refer to the section [DOE References](#)).

As a final observation concerning this 2-level, 2-factor design, consider the case where little or no interaction exists between factors A and B. In this situation, the interaction term would be superfluous. But suppose there is a third factor, C, that would be desirable to study. By letting $C = AB$, a three factor experiment could be conducted in half as many treatment combinations as a factorial experiment. This experimental scheme is referred to as a *fractional factorial* experiment. (Often the factorial experiment is referred to as a *full factorial*.) Obviously significant savings can be afforded by the fractional factorial, but the downside of this approach is that a priori knowledge of the strength of interactions is usually unavailable.

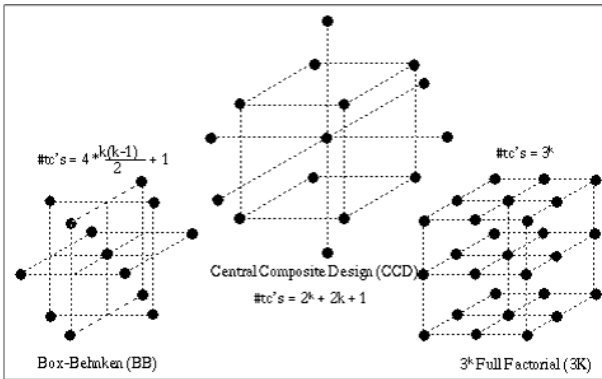
Since the change in the response due to factor C is *aliased* (or *confounded*) with interaction AB, fractional-factorial designs are usually used as screening experiments to identify a small subset of variables that contribute significantly to performance variation. One such screening experiment, the Plackett-Burman design, allows the study of $k=N-1$ variables in N runs, where N is a multiple of 4. (Any design where $k=N-1$ is referred to as *saturated*.)

A convenient way to designate all 2-level designs is with the nomenclature 2^{k-p} which stands for "2 raised to the power of k minus p," where k is the number of factors and p is the so-called *fractionalization component*. With p equal to zero, a full factorial experiment is identified. For the example above, p equals one, which denotes a $(1/2)^p$ or 1/2 fractional factorial, indicating 1/2 the number of tc's of the full factorial are required.

Multilevel Designs

Multilevel designs are characterized as those having more than two levels. These designs

are useful in detecting and modeling curvature in the response as a function of the factors. After a screening experiment is performed, and the *vital few* factors are identified, multilevel designs such as those in the following figure can be used to more accurately predict the response.



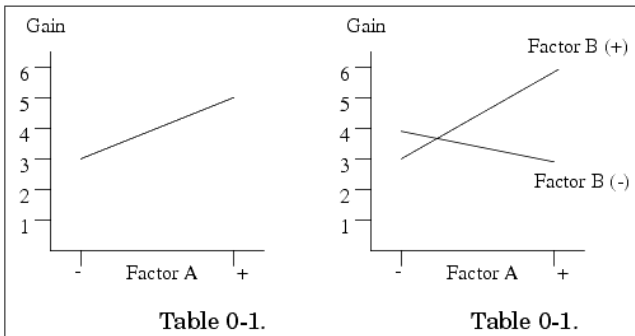
Common Multilevel Designs in Three Factors

The Central Composite Design (CCD) combines a 2-level experiment with the center point and so-called star points along the coordinate axis. The star points lie outside of the 2-level experiment and their distance from the center point is a function of the number of factors, i.e., $d = 2^{k/4}$. The Box-Behnken design consists of the zero point and a 2-level, 2-factor factorial design for all combinations of factors, while holding other factors at their middle value.

DOE Outputs

Prediction equations are not the only output from DOE. In fact, prediction equation(s) are usually obtained by examining *Effects plots*. *Pareto diagrams* are used to rank factors in order of their contribution to the total variance in the response, and *interaction diagrams* allow detection of interactions between factors.

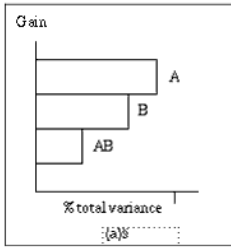
Effects plots are produced by simply computing the overall average response with the factor at each of its levels. For example, in the response shown in part a of the following figure, the average response for all tc's having A low is 3, and 5 when A is high. The coefficient of the prediction equation (2-level case) comes from the half effect (slope) of the main effect.



a) Effects Plot of Factor A vs. Gain, and b) Interaction Diagram for Factors A and B vs. Gain.

Interaction diagrams are iso-plots of two factors versus the response (see response shown in part b of the previous figure). Once again, average responses are computed over all tc's but this time with one additional constraint due to the companion factor. For example, the average response over all tc's having both A and B at low levels is 4, while it is 6 for the case when both A and B are high. Notice that the slopes of the lines change as a function of B. This indicates an interaction between the factors.

Pareto diagrams are bar charts that show the percentage of the total response variance attributable to each factor and interaction. See the following figure.



Pareto diagram of factors with respect to gain.

Response Values

In a typical industrial application of DOE, it is usually no problem to identify the response variables. In an injection molding application, the response variables might be the number of cracks produced in the casting, or the hardness of the product.

There are relatively few response variables. However, in computer-aided circuit or system design, where a continuous response over frequency, power, or other swept variable is approximated by discrete samples, the number of DOE response variables can number in the hundreds. The amount of data could be overwhelming to the designer.

Currently there are two schools of thought on accommodating response complexity. In the first approach, key points in a frequency/power comb are considered as individual DOE response variables to be analyzed *in parallel*. For example, low, mid, and high band edge samples of gain and noise figure would require six responses to be considered simultaneously.

An alternate approach involves the so-called Taguchi [4] loss-function. The overall DOE response is computed by combining each measurement's loss-function. The loss-function formulation depends on the relational operator used in defining each specification statement.

The ">" and "<" operations are interpreted as *bigger is better* and *smaller is better*, respectively. The equality constraint suggests that the response average be put "on target" with as little variance about the target as possible. The following loss-function formulations are used in the implementation of DOE:

Smaller is better (<): $-10 \log_{10} (\text{Sum}(x^2) / n)$

Larger is better (>): $-10 \log_{10} (\text{Sum}(1/x^2) / n)$

Target is best (=): $10 \log_{10} (M^2/s^2)$, where $s^2 = \text{Sum}(x-M)^2 / n-1$

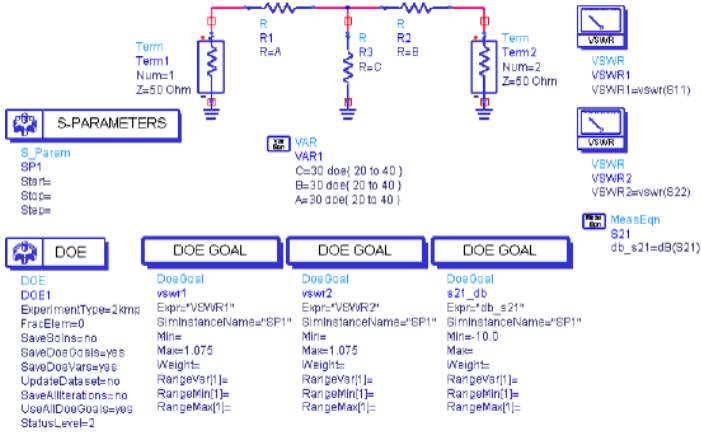
DOE Basic Example

The following example provides a basic introduction to the use of the software's DOE feature.

Set up the Schematic window as shown in the following figure, or copy it from the ADS Examples: \$HPPEESOF_DIR/examples/Tutorial/dae2_wrk.

Note

Since this example has already been set up, the steps shown in this section can be followed to learn the general approach to using the DOE feature.



Schematic Used as Starting Point for DOE Example

Setup the DOE Goal Components

1. Each DOE Goal will use a pre-specified measurement. In this case, VSWR1, VSWR2, and dB_s21. To setup each DOE Goal Component, double click the component. The Goal for Design of Experiment dialog box appears. The Goal for Design of Experiment dialog box appears. The Goal for dB_s21 is shown below.

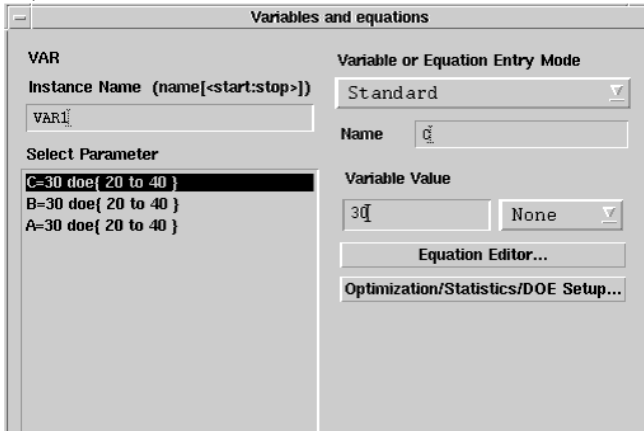
The screenshot shows the DoeGoal dialog box for the dB_s21 measurement. The Instance Name is s21_db. The Select Parameter section shows Expr="db_s21", SimInstanceName="SP1", Min=-10.0, Max=, Weight=, Save=, RangeVar[1]=, RangeMin[1]=, and RangeMax[1]=. The Measurement Equations section shows db_s21, VSWR2, and VSWR1. The Selection section shows db_s21. The Display parameter on schematic checkbox is checked. The Component Options... button is visible. The Expr field at the bottom is Specification expression name.

2. In our example, the DOE goal for the S21 measurement is already setup. The steps needed to setup a DOE goal were described in [Setting DOE Goals](#), and are similar to setting up optimization goals. In this example, note the following fields:
 - The desired measurement is **Expr="db_S21"**.
 - The Min and Max fields control the target DOE value. The Min field is set to **-10** and the Max field is left blank, which means that the target value is ≥ -10 .
 - Leave the Weight field at its default setting (**1**).
 - Leave the other fields blank in this example.
 - Click **OK** when done.
3. The procedure described in step 2, above, is repeated for the other two DOE goals for the VSWR measurements, by setting the goals as follows:
 - VSWR1 max = 1.075
 - VSWR2 max = 1.075

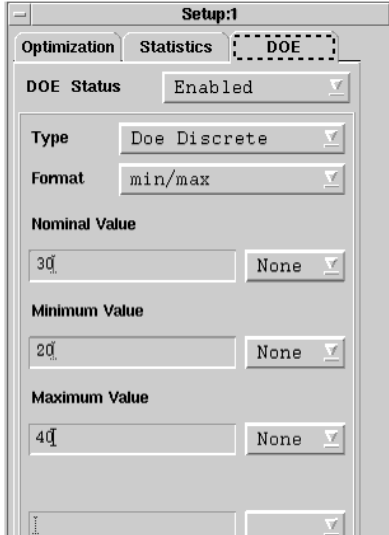
Note
 During DOE analysis, a complete set of DOE outputs (Pareto, Effects, and Interaction diagrams) are generated for each DOE goal component. For the case when there are several measurements (perhaps over frequency and/or power), the DOE response is computed by combining each measurement's loss-function. The loss-function, as defined by Taguchi [4] depends on the Min and/or Max fields used in defining each DOE goal. (Refer to the subsection "Response values" under [DOE Outputs](#)).
 If there is only a single measurement and only one sweep point, the DOE response is computed from the actual measurement minus the number in the Value field of the DOE specification. If the actual response value is desired, leave the Value field fixed and the DOE specification blank (or zero).

The variable values for resistors is set using the VAR component as follows:

4. Double-click the **Var/Eqn** component to bring up the Variables and equations dialog box, shown below.



5. In the Name field, enter **C**.
6. Click the Optimization, Statistics/DOE button.
7. In the dialog box that appears, choose the **DOE** tab.
8. In the DOE status field, select **Enabled** and Format to **min/max**.
9. Set the Nominal Value fields as follows and as shown in the figure below.
 - Nominal Value: **30**
 - Minimum Value: **20**
 - Maximum Value: **40**

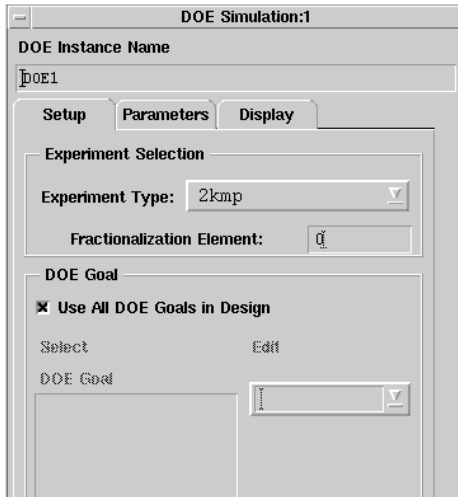


10. Choose **OK**. The following appears in the Select Parameter list box:
 C= 30 doe {20 to 40}
11. Repeat step 7, changing C to B, then choose **Add**.
12. Repeat step 7, changing B to A, then choose **Add**.
13. Choose **OK** to dismiss the dialog box.

Setup the DOE Component

Next, we will setup the DOE Component.

1. Double click the DOE Component. The DOE Simulation dialog box appears with the Setup tab active, as shown below.



In DOE, the type of experiment to run is both problem-dependent and subjective. Because there are only three factors in the current example, the 2kmp is initially used (refer to the section [DOE Concepts](#)).

2. Select **2kmp** in the Experiment Type drop-down list and leave the *Fractionalization Element* at its default setting, **0**.
3. Click **Apply**.
4. Select the Parameters tab.
5. The Parameters tab controls output data. Accept the defaults and click **Apply**.
6. Select the Display tab.
7. The Display tab controls which parameters are displayed on the schematic. Accept the defaults and click **OK**.

Start the Experiment

1. In the Schematic window, select **Simulate > Simulate**.

You see messages in the Status window showing the current treatment combination number. For the 2kmp with $k = 3$ and $p = 0$, there are 8 treatment combinations necessary for the experiment. Once the number of simulations is complete, a message appears, reporting the progress of DOE data computation and display.

Analyze the Experiment

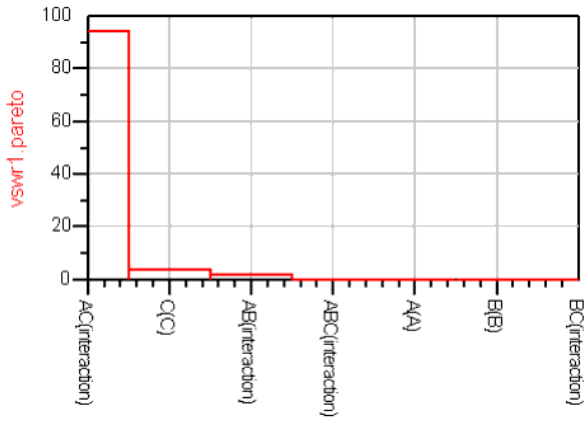
Once the treatment combinations have been simulated and the DOE data computation and display task is complete, you can access the three main DOE reporting tools (refer to the section [DOE Outputs](#) for explanations of these plots):

- Pareto diagrams
- Effects plots
- Interactions diagrams

Pareto Diagrams

To examine the Pareto diagrams:

1. Select **Window > New Data Display**.
2. Select a Rectangular Plot and place it in the middle of the window. The Plot Traces & Attributes dialog box appears.
3. From the Datasets and Equations drop-down list, select **s21_db.pareto**.
4. Choose the **Add** button.
5. Choose **OK**.
6. A diagram showing s21_db versus design variables is shown. Note that the Datasets and Equations drop-down list contains a series of plots and diagrams for each goal. Using the Plot Traces & Attributes dialog box, you can choose the diagram you are interested in, and add or delete traces in the plots. Let's select a Pareto diagram for a different DOE goal:
7. Double-click the Rectangular Plot and select **vswr1.pareto** from the Datasets and Equations drop-down list that appears. The plot is shown in the following figure.



designVariables

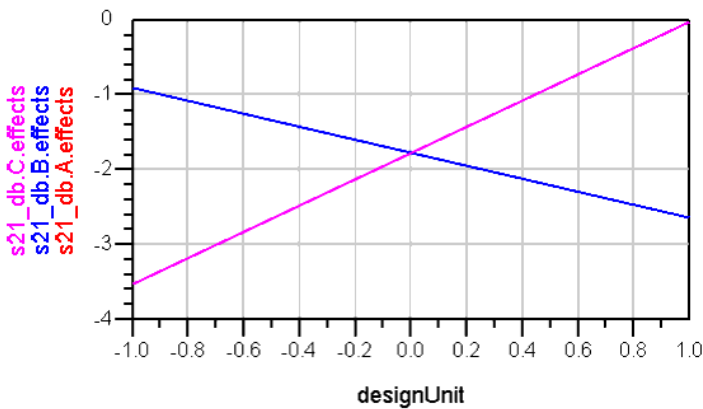
designVariables	vswr1_pareto
AC(interaction)	94.282
C(C)	3.659
AB(interaction)	1.743
ABC(interaction)	0.202
A(A)	0.068
B(B)	0.041
BC(interaction)	0.006

Pareto Diagram for VSWR1 Measurement

Effects Plots

Next let's examine the Effects plots for the S21 DOE goal:

1. From the opened Data Display window, Choose **File > New** .
2. Place a Rectangular Plot in the window. The Plot Traces & Attributes dialog box appears.
3. From the Datasets and Equations drop-down list, select **s21_db.A.effects** and click the **Add** button.
4. Next, select **s21_db.B.effects** and click **Add** .
5. Lastly, select **s21_db.C.effects** and click **Add** .
6. Click **OK** . The Effect plot appears as shown in the following figure.



Effects Plot for DOE Goal S21

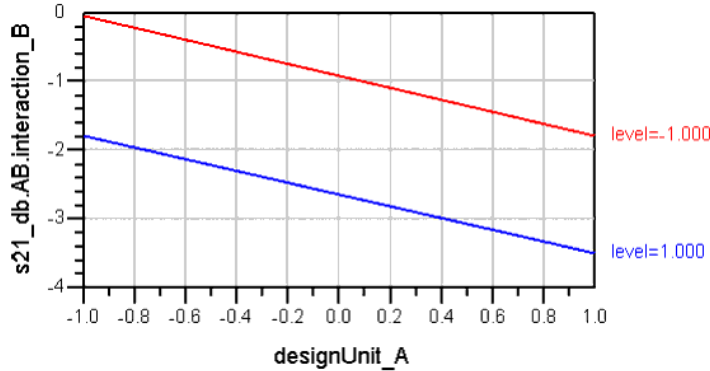
Notice that factor C has the largest slope (Effect) of the three factors. Notice also that factors A and B have identical effects. Finally, note that the response has been offset by the specified goal appearing in the DOE goal components. For example, the goal for S21 is -10 dB. Because the goal is subtracted from each response, the target response on the Effects plot is zero.

Interaction Diagrams

Interaction diagrams are used to examine the effect of one factor on a different factor.

To obtain the Interaction diagram for factors AB on the S21 DOE goal:

1. Select **Window > New Data Display** .
2. Select a Rectangular Plot and place it in the middle of the window. The Plot Traces & Attributes dialog box appears.
3. From the Datasets and Equations drop-down list, select **s21_db.AB.interaction_B** .
4. Choose the **Add** button.
5. Choose **OK** . The AB interaction diagram for DOE goal S21 appears, as shown in the following figure.



AB Interaction Diagram for S21 Measurement

Notice that the traces are parallel, indicating that there is little or no interaction between factors A and B - the change in average response as a function of factor A does not change as a function of factor B. However, there is an offset indicating that to achieve the (adjusted) target goal of zero, both A and B should be set to the low levels.

Optimizing Using DOE Outputs

In the DOE implementation, there are two methods to accomplish design improvement. The first and easiest to apply involves examining Effects plots and making approximate changes to the design factors in an effort to put the response on target. The second method involves solution of the set of model equations. In this section, the first method is examined, with the same example used previously in this topic.

By examining the Effects plot for the DOE goal S21, it is clear that an increase in C and a decrease in the value of A and B will work toward putting the nominal response on the (adjusted) target value of zero. However, while this observation is true for the S21 goal, it may not be accurate for the VSWR goals.

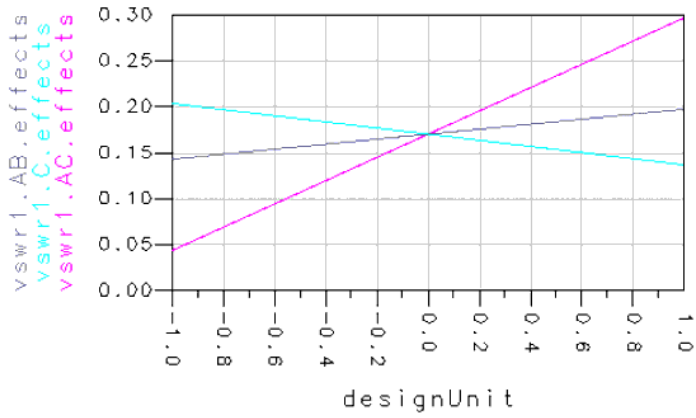
Let's continue following along with our example (../examples/Tutorial/dae2_wrk). To view the optimized DOE output, select the design *dae2b* (from the Schematic window, choose *File* and select *dae2b* from the file history list at the bottom of the menu).

To examine the Effects plot for Vswr1:

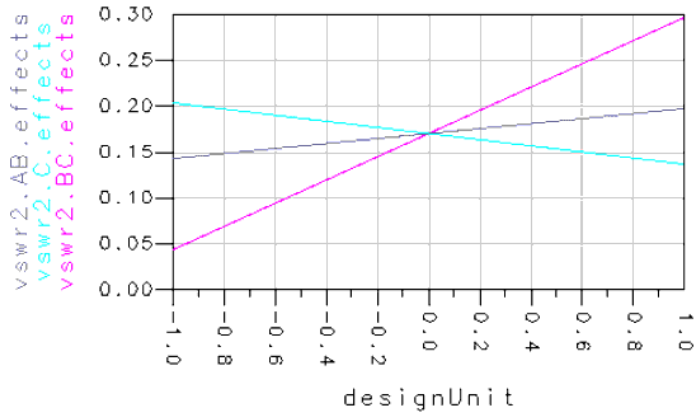
1. Select **Window > New Data Display** .
2. Select a Rectangular Plot and place it in the middle of the window. The Plot Traces & Attributes dialog box appears.
3. From the Datasets and Equations drop-down list, select **vswr1.A.effects** .
4. Choose the **Add** button.
5. Using the scroll bar of the list, locate, select and **Add** the first three components associated with Effects of Vswr1:
vswr1.B.effects
vswr1.C.effects
6. Click **OK** to review the Vswr1 Effects plot.

If you do not have a display for the S21 and Vswr2 effects, follow the above instructions to obtain any missing plot. The next three figures show the Effects plots for Vswr1, Vswr2, and S21, respectively.

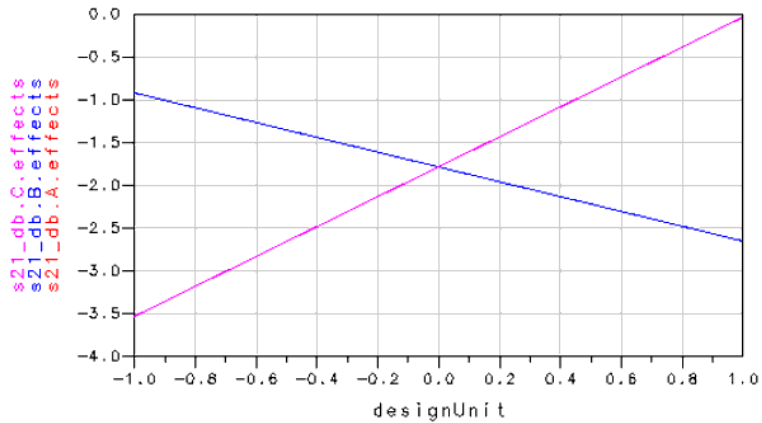
Once Effects plots for S21, Vswr1, and Vswr2 are available, arrange them so that they can be viewed simultaneously. Notice that the traces of the Vswr1 and Vswr2 plots are the same. The difference is that factor A in one plot is replaced by factor B in the other. This makes sense due to the symmetry in the network - the series resistors (factors A and B) take on the same values.



Effects Plot for Vswr1



Effects Plot for Vswr2



Effects Plot for S21

How Goals Are Affected

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

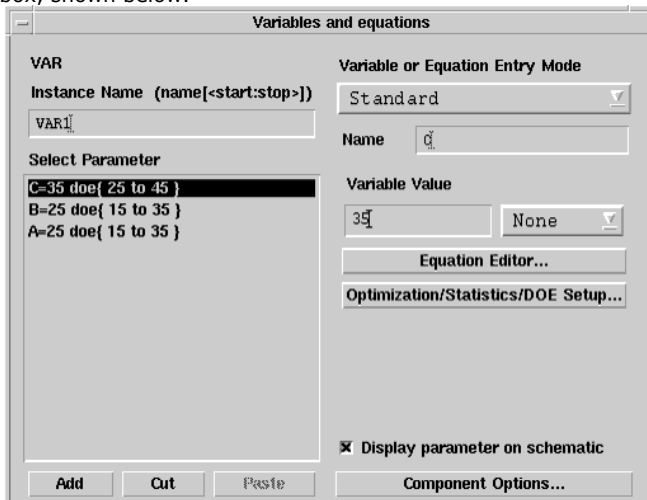
The motivation behind viewing Vswr1, Vswr2, and S21 effects concurrently is to ensure that any factor modifications are commensurate with the overall performance goals. As mentioned previously, it appears that an increase in C and a decrease in the value of A and B will work toward satisfying the S21 goal. We must also consider how this affects the Vswr goals. Noting the significant positive slope of the AC (Vswr1) and BC (Vswr2) interaction effects, it appears that an increase in C and a decrease in A and B would be favorable to our overall goals.

The only question that remains now is how much to change the factor level nominal values. Let's try an increase in C by 1/2 unit and a decrease in A and B by the same amount.

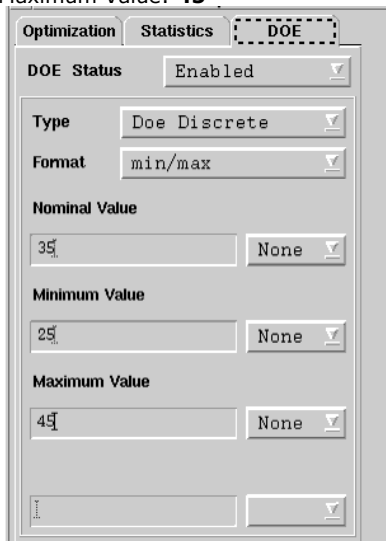
Noting that a 1/2 unit change in design units equates to a 5-ohm change in resistance values, the nominal values for series resistors drop from 30 to 25 ohms, and the shunt resistor should be changed from 30 to 35 ohms.

To modify the nominal, minimum, and maximum values for the factors:

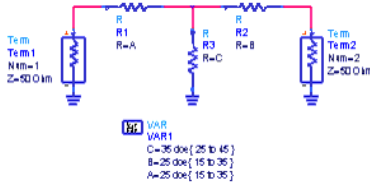
1. Double-click the **Var/Eqn** component to bring up the Variables and equations dialog box, shown below.



2. In the Name field, enter **C**.
3. Click the Optimization, Statistics/DOE button.
4. In the dialog box that appears, choose the **DOE** tab.
5. In the DOE status field, select **Enabled** and Format to **min/max**.
6. Set the Nominal Value fields as follows and as shown in the figure below.
 - Nominal Value: **35**
 - Minimum Value: **25**
 - Maximum Value: **45**



7. Choose **OK**. The following appears in the Select Parameter list box:
C= 35 doe {25 to 45}
8. Repeat step 7, changing C to B, then choose **Add**.
9. Repeat step 7, changing B to A, then choose **Add**.
10. Click **OK**. The changes in the *Var/Eqn* component are reflected in the schematic. The new schematic should look similar to the one shown in the following figure.



Schematic with Modified Var/Equ Component

Performing the DOE Confirmation Experiment

To start the experiment using the new factor nominal values:

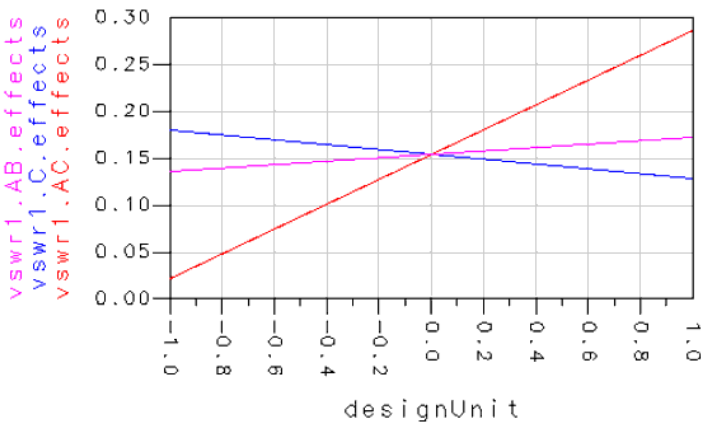
1. In the Schematic window, select **Simulate > Simulate** .
You see messages in the Status window showing the current treatment combination number. For the 2kmp with $k = 3$ and $p = 0$, there are 8 treatment combinations necessary for the experiment. Once the number of simulations is complete, a message appears, reporting the progress of DOE data computation and display.

Analyzing the DOE Confirmation Experiment

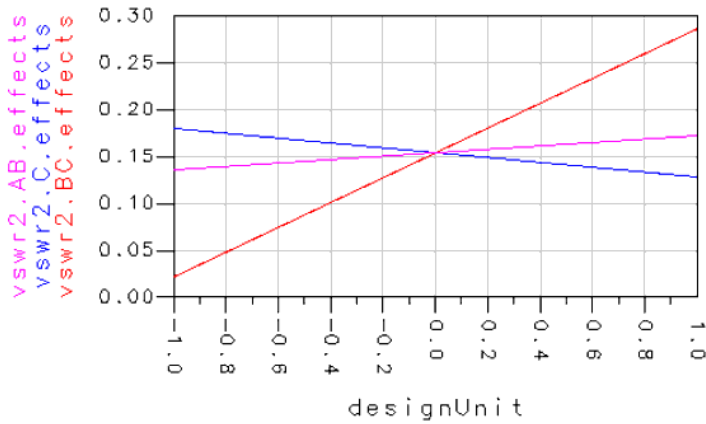
Once the treatment combinations have been simulated and the DOE data computation and display task is complete, you can re-examine the three Effects plots created earlier for Vswr and S21 DOE goals.

To examine the Effects plot for Vswr1:

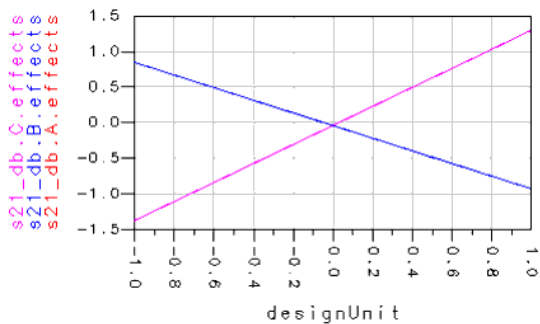
1. Repeat the steps described in the section [Effects Plots](#), except now we are using the results from the *doe2b* dataset.
2. Repeat steps 1-5 to obtain similar plots for the first three effects of S21 and Vswr2. The Effects plots for Vswr1, Vswr2, and S21 are shown in each of the following three figures respectively.



Effects Plot for Vswr1



Effects Plot for Vswr2



Main Effects Plot for S21

Once Effects plots for S21, Vswr1, and Vswr2 are available, arrange them so that they can be viewed simultaneously. Notice that our S21 target is very nearly satisfied while the VSWR goals are off target by about 0.17. The movement in the average response for VSWR is small—from about 0.16 to 0.17. If this performance were deemed unacceptable, another iteration of the procedure could be applied.

DOE References

1. Robert L. Mason, Richard F. Gunst, and James L. Hess, *Statistical Design and Analysis of Experiments (with Applications to Engineering and Science)*, New York, NY: John Wiley & Sons, 1989.
2. Douglas C. Montgomery, *Design and Analysis of Experiments, 2nd Ed.*, New York, NY: John Wiley & Sons, 1984.
3. Thomas B. Barker, *Quality by Experimental Design*, New York, NY; Marcel Dekker, 1985.
4. Thomas B. Barker, *Engineering Quality by Design - Interpreting the Taguchi Approach*, New York, NY; Marcel Dekker, 1990.
5. Stephen R. Schmidt and Robert G. Launsby, *Understanding Industrial Designed Experiments*, 3rd Ed., Colorado Springs, Co.: Air Academy Press, 1992.
6. Keki R. Bhote, *World Class Quality (Understanding Design of Experiments to Make it Happen)*, New York, NY: AMACOM.

Available Value Types

This topic provides descriptions of the available parameter value types for Nominal Optimization, Statistical Design, and Design of Experiments (DOE). For the procedures in which these value types are implemented, refer to the following topics:

- *Nominal Optimization* (optstat)
- *Using Statistical Design* (optstat)
- *Using Design of Experiments (DOE)* (optstat)

Value Types for Nominal Optimization

As described in the section, *Specifying Component Parameters for Optimization* (optstat), the *Optimization* tab of the *Setup* dialog box is used to enable or disable the optimization status of a parameter and to specify the type and format for the parameter range over which optimization is to take place.

In the *Optimization* tab, the *Type* drop-down list includes the following options:

Discrete Denotes a variable that is only allowed to take a specific list of values between a specified range. The range of discrete values is directly specified when you enter nominal value, minimum value, maximum value, and a step value. Notice that for this option, the *Format* drop-down list only includes *min/max/step*.

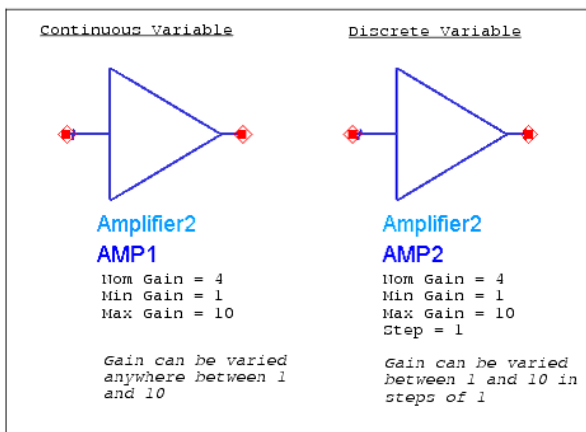
Note

The discrete variable type is compatible only with the Random, Random Minimax, Random Max, Discrete, and Genetic optimization types. (Refer to the section, *Available Optimizers*. (optstat) This variable type is ignored for all other nominal optimization methods, such as Gradient.

Continuous Denotes a variable that can be one of four types, which are selected from the *Format* drop down list, as follows:

- **min/max** Enables you to specify a nominal value, minimum value, and maximum value and to specify appropriate units for each.
- **+/- Delta %** Enables you to specify a nominal value and an appropriate unit for it, as well as Delta%.
- **+/- Delta** Enables you to specify a nominal value and an appropriate unit for it, as well as Delta.
- **Unconstrained** Enables you to specify only a nominal value and an appropriate unit for it. The range of values is constrained between zero and twice the initial nominal value automatically. Caution must be used when assigning this format to an item parameter. Many component parameters have a limited range for valid numeric assignment. Refer to *Introduction to Circuit Components* (ccsim) for component parameter value limits.

The simple example in the following figure shows the difference between continuous and discrete variables.



Continuous vs. Discrete Variables

Note that specifying continuous or discrete valued parts in your design has a direct impact on the type of optimizer that you can use in an optimization (see *Available Optimizers* (optstat)). Therefore, it is important to recognize and understand the difference before selecting an optimizer.

Value Types for Statistical Design

As described in the section, *Specifying Component Parameters for Yield Analysis (optstat)*, the *Statistics* tab of the *Setup* dialog box is used to enable or disable the yield analysis status of a parameter and to specify the type and format for the parameter range over which yield analysis is to take place.

In the *Statistics* tab, the *Type* drop-down list includes the following options:

Gaussian Denotes a Gaussian distributed statistical variable that can be one of two types, which are selected from the *Format* drop-down list, as follows:

- **+/- Std.Dev.%** Specifies the +/- 1 *sigma* deviation range as a percentage of the nominal value.
- **+/-Std.Dev.** Specifies the +/- 1 *sigma* deviation value as an absolute value.
- **Uniform** Denotes a variable that can be one of three types, which are selected from the *Format* drop down list, as follows:
 - **min/max** Enables you to specify a nominal value, minimum value, and maximum value and to specify appropriate units for each.
 - **+/- Delta %** Specifies the deviation range as a percentage of the nominal value.
 - **+/- Delta** Specifies the deviation value as an absolute value.
- **Discrete** Denotes a discrete uniform statistical variable. The set of discrete values is directly specified when you enter nominal value, minimum value, maximum value, and a step value. Notice that for this option, the *Format* drop-down list only includes *min/max/step*.
- **LogNormal** Denotes a log-normal distributed statistical variable. A log-normal distribution is a probability distribution in which the logarithm of the parameter has a normal distribution. It is the minimum-information distribution for positive quantities with a given geometric mean and standard deviation. It is also the multiplicative analog of the bell curve. This variable can be one of two types, which are selected from the *Format* drop-down list, as follows:
 - **+/- Std.Dev.%** Specifies the +/- 1 *sigma* deviation range as a percentage of the nominal value.
 - **+/-Std.Dev.** Specifies the +/- 1 *sigma* deviation value as an absolute value.

Value Types for DOE

As described in the section, *Specifying Component Parameters for DOE (optstat)*, the *DOE* tab of the *Setup* dialog box is used to enable or disable the DOE status of a parameter and to specify the type and format for the parameter range over which DOE is to take place.

In the *DOE* tab, the *Type* drop-down list includes the following options:

DOE Discrete Denotes a variable that can be one of three types, which are selected from the *Format* drop down list, as follows:

- **min/max** Enables you to specify a nominal value, minimum value, and maximum value and to specify appropriate units for each.
- **+/- Delta %** Specifies the deviation range as a percentage of the nominal value.
- **+/- Delta** Specifies the deviation value as an absolute value.

Using Monte Carlo Yield Analysis

Monte Carlo yield analysis methods have traditionally been widely used and accepted as a means to estimate yield. The method simply consists of performing a series of trials. Each trial results from randomly generating yield variable values according to statistical-distribution specifications, performing a simulation and evaluating the result against stated performance specifications.

The power of the Monte Carlo method is that the accuracy of the estimate rendered is independent of the number of statistical variables and requires no simplifying assumptions about the probability distribution of either component parameter values or performance responses.

The weakness of this method is that a full network simulation is required for each trial and that a large number of trials is required to obtain high confidence and an accurate estimate of yield. Fortunately, the simulator uses state-of-the-art techniques to significantly boost the efficiency of the Monte Carlo method [1, 2, 3] while retaining its generality.

For information about an example workspace that demonstrates Design for Manufacturing techniques to increase yields using Monte Carlo yield analysis, as well as DOE and Sensitivity Analysis, see *Design for Manufacturing Example Using Yield Sensitivity Histograms, DOE, and Sensitivity Analysis* (examples).

Consult the following references for details concerning state-of-the-art Monte Carlo techniques.

- M. D. Meehan and J. Purviance. *Yield and Reliability Design for Microwave Circuits and Systems*, Norwood, MA: Artech House, 1993.
- R. Spence and R. S. Sojn. *Tolerance Design of Electronic Circuits*, Addison-Wesley, 1988.
- D. C. Hocevar, M. R. Lightner, and T. N. Trick. "A study of variance reduction techniques for estimating circuit yields," *IEEE Trans. CAD*, vol. CAD-2, pp. 180-192, July 1983.

Monte Carlo Trials and Confidence Levels

The following discusses how to calculate the number of trials necessary for a given confidence and estimate error.

Confidence level is the area under a normal (gaussian) curve over a given number of standard deviations. Common values for confidence level are shown in the following.

Standard Deviations	Confidence Level
1	68.3%
2	95.4%
3	99.7%

Error is the absolute difference between the actual yield, Y , and the yield estimate, \bar{Y} , given by:

$$\epsilon = |Y - \bar{Y}|$$

where ϵ is the percent error. The low value limit of \bar{Y} is given by:

$$\bar{Y} = Y - \epsilon$$

The sample or trial size, N , is then calculated from:

$$N = \left(\frac{C_\sigma}{\epsilon}\right)^2 \cdot Y(1 - Y)$$

where C_σ is the confidence expressed as a number of standard deviations.

Example

For a 95.4% confidence level ($C_\sigma = 2$), an Error = $\pm 2\%$ and a yield of 80%

$$N = \left(\frac{2}{0.02}\right)^2 \cdot 0.8 \cdot (1 - 0.8)$$

$N = 1600$ trials

Refer to the section [Confidence Tables](#) for help in determining the number of trials

suitable for yield analysis.

The graphs shown in the figures [Yield for C = 1 \(68.3% confidence\)](#) through [Yield for C = 3 \(99.7% confidence\)](#) may also be helpful in determining the accuracy of a yield analysis that you've already performed. These graphs plot error bounds of actual yield versus estimated yield for various values of N (number of trials).

The graph shown in the figure [Yield for C = 1 \(68.3% confidence\)](#) plots error bounds with a confidence interval of one standard deviation, or 68.3% confidence level.

The graph shown in the figure [Yield for C = 2 \(95.4% confidence\)](#) plots error bounds with a confidence interval of two standard deviations, or 95.4% confidence level.

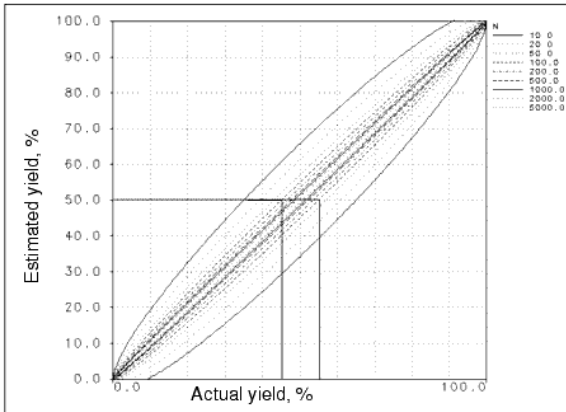
The graph shown in the figure [Yield for C = 3 \(99.7% confidence\)](#) plots error bounds with a confidence interval of three standard deviations, or 99.7% confidence level.

Suppose you ran a yield analysis on your design using 100 trials and the estimated yield was 50%. Referring to the graph in the figure [Yield for C = 1 \(68.3% confidence\)](#), the lower bound on the actual yield is 45% and the upper bound is 55%.

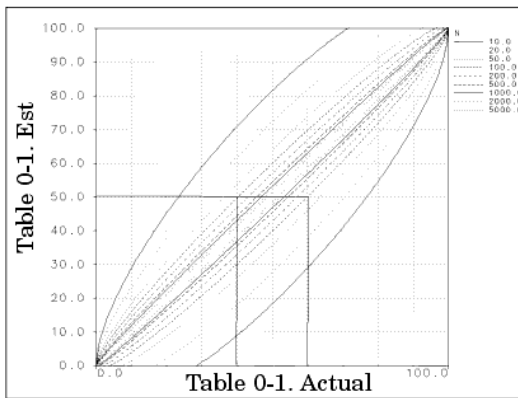
From the graph shown in the figure [Yield for C = 2 \(95.4% confidence\)](#), for 100 trials and an estimated yield of 50%, the lower bound on the actual yield is 40% and the upper bound is 60%.

Finally, from the graph shown in the figure [Yield for C = 3 \(99.7% confidence\)](#), for 100 trials and an estimated yield of 50%, the lower bound on the actual yield is about 35% and the upper bound is about 65%.

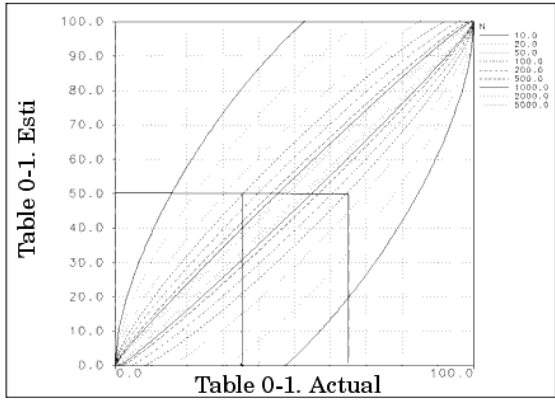
Thus if you performed a yield analysis (either Monte Carlo or shadow model) using 100 trials, and the estimated yield was 50%, you have a 68.3% probability (confidence) that the actual yield is between 45% and 55%. You have a 95.4% probability that the actual yield is between 40% and 60%, and the probability is 99.7% that the actual yield is between 35% and 65%.



[Yield for C = 1 \(68.3% confidence\)](#)



[Yield for C = 2 \(95.4% confidence\)](#)



Yield for C = 3 (99.7% confidence)

Confidence Tables

The confidence tables that follow can be used to determine the number of trials suitable for yield analysis.

Confidence = 68.3% / Actual Yield = 90%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	89.00	91.00	900
2.0	88.00	92.00	225
3.0	87.00	93.00	100
4.0	86.00	94.00	56
5.0	85.00	95.00	36
6.0	84.00	96.00	25
7.0	83.00	97.00	18
8.0	82.00	98.00	14
9.0	81.00	99.00	11
10.0	80.00	100.00	9

Confidence = 95% / Actual Yield = 90%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	89.00	91.00	3457
2.0	88.00	92.00	864
3.0	87.00	93.00	384
4.0	86.00	94.00	216
5.0	85.00	95.00	138
6.0	84.00	96.00	96
7.0	83.00	97.00	70
8.0	82.00	98.00	54
9.0	81.00	99.00	42
10.0	80.00	100.00	34

Confidence = 99% / Actual Yield = 90%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	89.00	91.00	5967
2.0	88.00	92.00	1491
3.0	87.00	93.00	663
4.0	86.00	94.00	372
5.0	85.00	95.00	238
6.0	84.00	96.00	165
7.0	83.00	97.00	121
8.0	82.00	98.00	93
9.0	81.00	99.00	73
10.0	80.00	100.00	59

Confidence = 68.3% / Actual Yield = 80%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	79.00	81.00	1600
2.0	78.00	82.00	400
3.0	77.00	83.00	177
4.0	76.00	84.00	100
5.0	75.00	85.00	64
6.0	74.00	86.00	44
7.0	73.00	87.00	32
8.0	72.00	88.00	25
9.0	71.00	89.00	19
10.0	70.00	90.00	16

Confidence = 95% / Actual Yield = 80%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	79.00	81.00	6146
2.0	78.00	82.00	1536
3.0	77.00	83.00	682
4.0	76.00	84.00	384
5.0	75.00	85.00	245
6.0	74.00	86.00	170
7.0	73.00	87.00	125
8.0	72.00	88.00	96
9.0	71.00	89.00	75
10.0	70.00	90.00	61

Confidence = 99% / Actual Yield = 80%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	79.00	81.00	10609
2.0	78.00	82.00	2652
3.0	77.00	83.00	1178
4.0	76.00	84.00	663
5.0	75.00	85.00	424
6.0	74.00	86.00	294
7.0	73.00	87.00	216
8.0	72.00	88.00	165
9.0	71.00	89.00	130
10.0	70.00	90.00	106

Confidence = 68.3% / Actual Yield = 70%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	69.00	71.00	2100
2.0	68.00	72.00	525
3.0	67.00	73.00	233
4.0	66.00	74.00	131
5.0	65.00	75.00	84
6.0	64.00	76.00	58
7.0	63.00	77.00	42
8.0	62.00	78.00	32
9.0	61.00	79.00	25
10.0	60.00	80.00	21

Confidence = 95% / Actual Yield = 70%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	69.00	71.00	8067
2.0	68.00	72.00	2016
3.0	67.00	73.00	896
4.0	66.00	74.00	504
5.0	65.00	75.00	322
6.0	64.00	76.00	224
7.0	63.00	77.00	164
8.0	62.00	78.00	126
9.0	61.00	79.00	99
10.0	60.00	80.00	80

Confidence = 99% / Actual Yield = 70%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	69.00	71.00	13924
2.0	68.00	72.00	3481
3.0	67.00	73.00	1547
4.0	66.00	74.00	870
5.0	65.00	75.00	556
6.0	64.00	76.00	386
7.0	63.00	77.00	284
8.0	62.00	78.00	217
9.0	61.00	79.00	171
10.0	60.00	80.00	139

Confidence = 68.3% / Actual Yield = 60%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	59.00	61.00	2400
2.0	58.00	62.00	600
3.0	57.00	63.00	266
4.0	56.00	64.00	150
5.0	55.00	65.00	96
6.0	54.00	66.00	66
7.0	53.00	67.00	48
8.0	52.00	68.00	37
9.0	51.00	69.00	29
10.0	50.00	70.00	24

Confidence = 95% / Actual Yield = 60%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	59.00	61.00	9219
2.0	58.00	62.00	2304
3.0	57.00	63.00	1024
4.0	56.00	64.00	576
5.0	55.00	65.00	368
6.0	54.00	66.00	256
7.0	53.00	67.00	188
8.0	52.00	68.00	144
9.0	51.00	69.00	113
10.0	50.00	70.00	92

Confidence = 99% / Actual Yield = 60%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	59.00	61.00	15913
2.0	58.00	62.00	3978
3.0	57.00	63.00	1768
4.0	56.00	64.00	994
5.0	55.00	65.00	636
6.0	54.00	66.00	442
7.0	53.00	67.00	324
8.0	52.00	68.00	248
9.0	51.00	69.00	196
10.0	50.00	70.00	159

Confidence = 68.3% / Actual Yield = 50%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	49.00	51.00	2500
2.0	48.00	52.00	625
3.0	47.00	53.00	277
4.0	46.00	54.00	156
5.0	45.00	55.00	100
6.0	44.00	56.00	69
7.0	43.00	57.00	51
8.0	42.00	58.00	39
9.0	41.00	59.00	30
10.0	40.00	60.00	25

Confidence = 95% / Actual Yield = 50%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	49.00	51.00	9604
2.0	48.00	52.00	2401
3.0	47.00	53.00	1067
4.0	46.00	54.00	600
5.0	45.00	55.00	384
6.0	44.00	56.00	266
7.0	43.00	57.00	196
8.0	42.00	58.00	150
9.0	41.00	59.00	118
10.0	40.00	60.00	96

Confidence = 99% / Actual Yield = 50%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	49.00	51.00	16576
2.0	48.00	52.00	4144
3.0	47.00	53.00	1841
4.0	46.00	54.00	1036
5.0	45.00	55.00	663
6.0	44.00	56.00	460
7.0	43.00	57.00	338
8.0	42.00	58.00	259
9.0	41.00	59.00	204
10.0	40.00	60.00	165

Confidence = 68.3% / Actual Yield = 40%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	39.00	41.00	2400
2.0	38.00	42.00	600
3.0	37.00	43.00	266
4.0	36.00	44.00	150
5.0	35.00	45.00	96
6.0	34.00	46.00	66
7.0	33.00	47.00	48
8.0	32.00	48.00	37
9.0	31.00	49.00	29
10.0	30.00	50.00	24

Confidence = 95% / Actual Yield = 40%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	39.00	41.00	9219
2.0	38.00	42.00	2304
3.0	37.00	43.00	1024
4.0	36.00	44.00	576
5.0	35.00	45.00	368
6.0	34.00	46.00	256
7.0	33.00	47.00	188
8.0	32.00	48.00	144
9.0	31.00	49.00	113
10.0	30.00	50.00	92

Confidence = 99% / Actual Yield = 40%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	39.00	41.00	15913
2.0	38.00	42.00	3978
3.0	37.00	43.00	1768
4.0	36.00	44.00	994
5.0	35.00	45.00	636
6.0	34.00	46.00	442
7.0	33.00	47.00	324
8.0	32.00	48.00	248
9.0	31.00	49.00	196
10.0	30.00	50.00	159

Confidence = 68.3% / Actual Yield = 30%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	29.00	31.00	2100
2.0	28.00	32.00	525
3.0	27.00	33.00	233
4.0	26.00	34.00	131
5.0	25.00	35.00	84
6.0	24.00	36.00	58
7.0	23.00	37.00	42
8.0	22.00	38.00	32
9.0	21.00	39.00	25
10.0	20.00	40.00	21

Confidence = 95% / Actual Yield = 30%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	29.00	31.00	8067
2.0	28.00	32.00	2016
3.0	27.00	33.00	896
4.0	26.00	34.00	504
5.0	25.00	35.00	322
6.0	24.00	36.00	224
7.0	23.00	37.00	164
8.0	22.00	38.00	126
9.0	21.00	39.00	99
10.0	20.00	40.00	80

Confidence = 99% / Actual Yield = 30%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	29.00	31.00	13924
2.0	28.00	32.00	3481
3.0	27.00	33.00	1547
4.0	26.00	34.00	870
5.0	25.00	35.00	556
6.0	24.00	36.00	386
7.0	23.00	37.00	284
8.0	22.00	38.00	217
9.0	21.00	39.00	171
10.0	20.00	40.00	139

Confidence = 68.3% / Actual Yield = 20%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	19.00	21.00	1600
2.0	18.00	22.00	400
3.0	17.00	23.00	177
4.0	16.00	24.00	100
5.0	15.00	25.00	64
6.0	14.00	26.00	44
7.0	13.00	27.00	32
8.0	12.00	28.00	25
9.0	11.00	29.00	19
10.0	10.00	30.00	16

Confidence = 95% / Actual Yield = 20%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	19.00	21.00	6146
2.0	18.00	22.00	1536
3.0	17.00	23.00	682
4.0	16.00	24.00	384
5.0	15.00	25.00	245
6.0	14.00	26.00	170
7.0	13.00	27.00	125
8.0	12.00	28.00	96
9.0	11.00	29.00	75
10.0	10.00	30.00	61

Confidence = 99% / Actual Yield = 20%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	19.00	21.00	10609
2.0	18.00	22.00	2652
3.0	17.00	23.00	1178
4.0	16.00	24.00	663
5.0	15.00	25.00	424
6.0	14.00	26.00	294
7.0	13.00	27.00	216
8.0	12.00	28.00	165
9.0	11.00	29.00	130
10.0	10.00	30.00	106

Confidence = 68.3% / Actual Yield = 10%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	19.00	21.00	899
2.0	18.00	22.00	224
3.0	17.00	23.00	100
4.0	16.00	24.00	56
5.0	15.00	25.00	36
6.0	14.00	26.00	25
7.0	13.00	27.00	18
8.0	12.00	28.00	14
9.0	11.00	29.00	11
10.0	10.00	30.00	9

Confidence = 95% / Actual Yield = 10%

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	19.00	21.00	3457
2.0	18.00	22.00	864
3.0	17.00	23.00	384
4.0	16.00	24.00	216
5.0	15.00	25.00	138
6.0	14.00	26.00	96
7.0	13.00	27.00	70
8.0	12.00	28.00	54
9.0	11.00	29.00	42
10.0	10.00	30.00	34

Confidence = 99% / Actual Yield = 10%

Advanced Design System 2011.01 - Tuning, Optimization, and Statistical Design

Error +/- %	Estimated % Yield		Numberof Trials
	Low	High	
1.0	19.00	21.00	5967
2.0	18.00	22.00	1491
3.0	17.00	23.00	663
4.0	16.00	24.00	372
5.0	15.00	25.00	238
6.0	14.00	26.00	165
7.0	13.00	27.00	121
8.0	12.00	28.00	93
9.0	11.00	29.00	73
10.0	10.00	30.00	59