

# Optimization Module

*User's Guide*



# Optimization Module User's Guide

© 1998–2012 COMSOL

Protected by U.S. Patents 7,519,518; 7,596,474; and 7,623,991. Patents pending.

This Documentation and the Programs described herein are furnished under the COMSOL Software License Agreement ([www.comsol.com/sla](http://www.comsol.com/sla)) and may be used or copied only under the terms of the license agreement.

COMSOL, COMSOL Desktop, COMSOL Multiphysics, and LiveLink are registered trademarks or trademarks of COMSOL AB. Other product or brand names are trademarks or registered trademarks of their respective holders.

Version:

May 2012

COMSOL 4.3

## Contact Information

Visit [www.comsol.com/contact](http://www.comsol.com/contact) for a searchable list of all COMSOL offices and local representatives. From this web page, search the contacts and find a local sales representative, go to other COMSOL websites, request information and pricing, submit technical support queries, subscribe to the monthly eNews email newsletter, and much more.

If you need to contact Technical Support, an online request form is located at [www.comsol.com/support/contact](http://www.comsol.com/support/contact).

Other useful links include:

- Technical Support [www.comsol.com/support](http://www.comsol.com/support)
- Software updates: [www.comsol.com/support/updates](http://www.comsol.com/support/updates)
- Online community: [www.comsol.com/community](http://www.comsol.com/community)
- Events, conferences, and training: [www.comsol.com/events](http://www.comsol.com/events)
- Tutorials: [www.comsol.com/products/tutorials](http://www.comsol.com/products/tutorials)
- Knowledge Base: [www.comsol.com/support/knowledgebase](http://www.comsol.com/support/knowledgebase)

C o n t e n t s

Chapter 1: Introduction

<b>Optimization Module Overview</b>	<b>8</b>
What Can the Optimization Module Do? . . . . .	8
Where Do I Find the Documentation and Model Library? . . . . .	9
Typographical Conventions . . . . .	11

Chapter 2: Optimization and Sensitivity Theory

<b>Theory for the Optimization Interface</b>	<b>16</b>
Basic Optimization Concepts . . . . .	16
Optimization Problem Formulation . . . . .	16
PDE-Constrained Optimization . . . . .	17
 <b>Theory for the Sensitivity Interface</b>	 <b>21</b>
About Sensitivity Analysis . . . . .	21
Sensitivity Problem Formulation . . . . .	22
Theory for Stationary Sensitivity Analysis . . . . .	22
Theory for Time-Dependent Sensitivity . . . . .	24
Specification of the Objective Function . . . . .	26
Choosing a Sensitivity Method. . . . .	27
Issues to Consider Regarding the Control Variable . . . . .	28
Issues to Consider Regarding the Objective Function . . . . .	29

Chapter 3: The Optimization Interface

<b>Adding Optimization to a Model</b>	<b>32</b>
 <b>The Optimization Interface</b>	 <b>33</b>
Least-Squares Objective . . . . .	34
Value Column . . . . .	35

Time Column . . . . .	35
Parameter Column . . . . .	35
Coordinate Column. . . . .	36
Ignored Column . . . . .	36
Integral Objective . . . . .	36
Probe Objective . . . . .	37
Integral Inequality Constraint . . . . .	37
Pointwise Inequality Constraint . . . . .	38
Control Variable Field . . . . .	39
Global Objective . . . . .	39
Global Least-Squares Objective . . . . .	39
Global Inequality Constraint . . . . .	40
Global Control Variables . . . . .	40

Chapter 4: The Sensitivity Interface

<b>The Sensitivity Interface</b>	<b>42</b>
Integral Objective . . . . .	43
Probe Objective . . . . .	43
Control Variable Field . . . . .	44
Global Objective . . . . .	45
Global Control Variables . . . . .	45
Adding a Sensitivity Interface to a Model . . . . .	45

Chapter 5: The Optimization Solvers

<b>The Optimization Solver Settings</b>	<b>50</b>
 <b>Optimization Solver Properties</b>	 <b>56</b>
Feastol . . . . .	56
Funcprec . . . . .	57
Hessupd . . . . .	58
Majfeastol . . . . .	58
Opttol . . . . .	59
Qpsolver . . . . .	59

References for the Optimization Solvers . . . . . 60

Chapter 6: Glossary

Glossary of Terms 62



# Introduction

Welcome to the *Optimization Module User's Guide*. The capabilities of the Optimization Module can be used in conjunction with any combination of other COMSOL products. This guide is a supplement to the *COMSOL Multiphysics User's Guide*. This chapter contains a short [Optimization Module Overview](#).

# Optimization Module Overview

In this section:

- [What Can the Optimization Module Do?](#)
- [Where Do I Find the Documentation and Model Library?](#)
- [Typographical Conventions](#)

## *What Can the Optimization Module Do?*

---

The Optimization Module can be used throughout the COMSOL Multiphysics product family—it is a general interface for calculating optimal solutions to engineering problems. Any model inputs, be it geometric dimensions, part shapes, material properties, or material distribution, can be treated as control variables, and any model output can be an objective function.

There are two optimization algorithms available in the module. The first algorithm is based on the SNOPT code developed by Philip E. Gill of the University of California San Diego, and Walter Murray and Michael A. Saunders of Stanford University. When using SNOPT, the objective function can have any form and any constraints can be applied. The algorithm uses a gradient-based optimization technique to find optimal designs and when the underlying PDE is stationary or time-dependent, analytic sensitivities of the objective function with respect to the control variables can be used.

The second algorithm is a Levenberg-Marquardt solver. When this solver is used, the objective function must be of least squares type. Also, constraints are not supported. Since the Levenberg-Marquardt method is derived to solve problems of least squares type, it typically converges faster than SNOPT for such problems.

Simulation is a powerful tool in science and engineering for predicting the behavior of physical systems, particularly those governed by partial differential equations. In many cases a single or a few simulations are not enough to provide sufficient understanding of a system. Two important classes of problems whose resolution relies on a more systematic exploratory process are:

- *Design problems* with a single objective. Here, the problem is to find the values of control variables or design variables that yield the best performance of the output of a simulation model when the latter is quantified by means of a single function.



Problems of this kind arise, for example, in structural optimization, antenna design, and process optimization.

- *Inverse problems*, and in particular *parameter estimation* in multiphysics models. Here, the problem is to reliably determine the values of a set of parameters that provide simulated data which best matches measured data. Such problems arise in, for example, geophysical imaging, nondestructive testing, and biomedical imaging.

It is often possible to reformulate problems of the above type as *optimization problems*. The Optimization interface in COMSOL Multiphysics is useful for solving design problems as well as inverse problems and parameter estimation.

### *Where Do I Find the Documentation and Model Library?*

---



There is no specific folder only for optimization models. However, you can find related models by entering **optimization** in the **Search** field.

---

A number of Internet resources provide more information about COMSOL Multiphysics, including licensing and technical information. The electronic documentation, Dynamic Help, and the Model Library are all accessed through the COMSOL Desktop.



If you are reading the documentation as a PDF file on your computer, the [blue links](#) do not work to open a model or content referenced in a different user's guide. However, if you are using the online help in COMSOL Multiphysics, these links work to other modules, model examples, and documentation sets.


---

## **THE DOCUMENTATION**

The *COMSOL Multiphysics User's Guide* and *COMSOL Multiphysics Reference Guide* describe all interfaces and functionality included with the basic COMSOL Multiphysics license. These guides also have instructions about how to use COMSOL Multiphysics and how to access the documentation electronically through the COMSOL Multiphysics help desk.

To locate and search all the documentation, in COMSOL Multiphysics:

- Press F1 for Dynamic Help,



- Click the buttons on the toolbar, or
- Select **Help>Documentation** (  ) or **Help>Dynamic Help** (  ) from the main menu


and then either enter a search term or look under a specific module in the documentation tree.

### THE MODEL LIBRARY

Each model comes with documentation that includes a theoretical background and step-by-step instructions to create the model. The models are available in COMSOL as MPH-files that you can open for further investigation. You can use the step-by-step instructions and the actual models as a template for your own modeling and applications.

SI units are used to describe the relevant properties, parameters, and dimensions in most examples, but other unit systems are available.

To open the Model Library, select **View>Model Library** (  ) from the main menu, and then search by model name or browse under a module folder name. Click to highlight any model of interest, and select **Open Model and PDF** to open both the model and the documentation explaining how to build the model. Alternatively, click the **Dynamic Help** button (  ) or select **Help>Documentation** in COMSOL to search by name or browse by module.

The model libraries are updated on a regular basis by COMSOL in order to add new models and to improve existing models. Choose **View>Model Library Update** (  ) to update your model library to include the latest versions of the model examples.

If you have any feedback or suggestions for additional models for the library (including those developed by you), feel free to contact us at [info@comsol.com](mailto:info@comsol.com).

### CONTACTING COMSOL BY EMAIL

For general product information, contact COMSOL at [info@comsol.com](mailto:info@comsol.com).

To receive technical support from COMSOL for the COMSOL products, please contact your local COMSOL representative or send your questions to [support@comsol.com](mailto:support@comsol.com). An automatic notification and case number is sent to you by email.



COMSOL WEB SITES

Main Corporate web site	<a href="http://www.comsol.com">www.comsol.com</a>
Worldwide contact information	<a href="http://www.comsol.com/contact">www.comsol.com/contact</a>
Technical Support main page	<a href="http://www.comsol.com/support">www.comsol.com/support</a>
Support Knowledge Base	<a href="http://www.comsol.com/support/knowledgebase">www.comsol.com/support/knowledgebase</a>
Product updates	<a href="http://www.comsol.com/support/updates">www.comsol.com/support/updates</a>
COMSOL User Community	<a href="http://www.comsol.com/community">www.comsol.com/community</a>

*Typographical Conventions*

All COMSOL user’s guides use a set of consistent typographical conventions that make it easier to follow the discussion, understand what you can expect to see on the graphical user interface (GUI), and know which data must be entered into various data-entry fields.






In particular, these conventions are used throughout the documentation:








CONVENTION	EXAMPLE
text <b>highlighted in blue</b>	Click text <b>highlighted in blue</b> to go to other information in the PDF. When you are using the online help desk in COMSOL Multiphysics, these links also work to other modules, model examples, and documentation sets.
<b>boldface font</b>	A <b>boldface</b> font indicates that the given word(s) appear exactly that way on the COMSOL Desktop (or, for toolbar buttons, in the corresponding tip). For example, the <b>Model Builder</b> window (  ) is often referred to and this is the window that contains the model tree. As another example, the instructions might say to click the <b>Zoom Extents</b> button (  ) , and this means that when you hover over the button with your mouse, the same label displays on the COMSOL Desktop.
Forward arrow symbol >	The forward arrow symbol > is instructing you to select a series of menu items in a specific order. For example, <b>Options&gt;Preferences</b> is equivalent to: From the <b>Options</b> menu, choose <b>Preferences</b> .
Code (monospace) font	A Code (monospace) font indicates you are to make a keyboard entry in the user interface. You might see an instruction such as “Enter (or type) 1.25 in the <b>Current density</b> field.” The monospace font also is an indication of programming code or a variable name.

CONVENTION	EXAMPLE
Italic <i>Code</i> (monospace) font	An italic <i>Code</i> (monospace) font indicates user inputs and parts of names that can vary or be defined by the user.
Arrow brackets <> following the Code (monospace) or <i>Code</i> (italic) fonts	<p>The arrow brackets included in round brackets after either a monospace Code or an italic <i>Code</i> font means that the content in the string can be freely chosen or entered by the user, such as feature tags. For example, <code>model.geom(&lt;tag&gt;)</code> where &lt;tag&gt; is the geometry's tag (an identifier of your choice).</p> <p>When the string is predefined by COMSOL, no bracket is used and this indicates that this is a finite set, such as a feature name.</p>

### KEY TO THE GRAPHICS

Throughout the documentation, additional icons are used to help navigate the information. These categories are used to draw your attention to the information based on the level of importance, although it is always recommended that you read these text boxes.

ICON	NAME	DESCRIPTION
	Caution	A Caution icon is used to indicate that the user should proceed carefully and consider the next steps. It might mean that an action is required, or if the instructions are not followed, that there will be problems with the model solution.
	Important	An Important icon is used to indicate that the information provided is key to the model building, design, or solution. The information is of higher importance than a note or tip, and the user should endeavor to follow the instructions.
	Note	A Note icon is used to indicate that the information may be of use to the user. It is recommended that the user read the text.
	Tip	A Tip icon is used to provide information, reminders, short cuts, suggestions of how to improve model design, and other information that may or may not be useful to the user.
	See Also	The See Also icon indicates that other useful information is located in the named section. If you are working on line, click the hyperlink to go to the information directly. When the link is outside of the current PDF document, the text indicates this, for example See <a href="#">The Laminar Flow Interface</a> in the <i>COMSOL Multiphysics User's Guide</i> . Note that if you are in COMSOL Multiphysics' online help, the link will work.

ICON	NAME	DESCRIPTION
	Model	<p>The Model icon is used in the documentation as well as in COMSOL Multiphysics from the View&gt;Model Library menu. If you are working online, click the link to go to the PDF version of the step-by-step instructions. In some cases, a model is only available if you have a license for a specific module. These examples occur in the COMSOL Multiphysics User's Guide. The Model Library path describes how to find the actual model in COMSOL Multiphysics, for example</p> <p>If you have the RF Module, see <a href="#">Radar Cross Section: Model Library path RF_Module/Tutorial_Models/radar_cross_section</a></p>
	Space Dimension	<p>Another set of icons are also used in the Model Builder—the model space dimension is indicated by 0D , 1D , 1D axial symmetry , 2D , 2D axial symmetry , and 3D  icons. These icons are also used in the documentation to clearly list the differences to an interface, feature node, or theory section, which are based on space dimension.</p>



# Optimization and Sensitivity Theory

This chapter discusses the theory for the Optimization and Sensitivity interfaces.  
In this chapter:

- [Theory for the Optimization Interface](#)
- [Theory for the Sensitivity Interface](#)

# Theory for the Optimization Interface

The [Optimization Interface](#) theory is described in this section:

- [Basic Optimization Concepts](#)
- [PDE-Constrained Optimization](#)
- [Optimization Problem Formulation](#)
- [Theory for the Sensitivity Interface](#)

## *Basic Optimization Concepts*

---

In general there are two fundamental parts of an optimization problem—the *control variables* and the *objective function*.

The optimization problem is to find the value of the control variables that minimizes (or maximizes) the objective function, subject to a number of constraints. The constraints collectively define a set, the *feasible set*, of permissible values for the control variables.

The Optimization interface provides a framework for specifying and solving optimization problems. The objective function and constraints can depend implicitly on the control variables by means of multiphysics models. See [The Optimization Interface](#).

## *Optimization Problem Formulation*

---

The Optimization interface is built around a general formulation of a minimization problem (to perform maximization, simply minimize the negative of the objective function). Specify the objective function, constraints, and control variables.

### **THE GENERAL OPTIMIZATION PROBLEM**

The general formulation of an optimization problem can be written as

$$\begin{cases} \min_{\xi} Q(\xi) \\ \xi \in C \end{cases} \quad (2-1)$$



Here, the control variables are denoted by  $\xi$ , the scalar-valued objective function by  $Q$ , and the feasible set, denoted by  $C$ , is assumed to be given by means of inequality constraints

$$C = \{\xi: \text{lb} \leq G(\xi) \leq \text{ub}\}$$

where  $G$  is a vector-valued function ( $G$  is scalar valued in case of a single constraint).



Note

For vectorial quantities, the inequality defining  $C$  is to be interpreted componentwise, and  $\text{lb}$  and  $\text{ub}$  are the corresponding vectors containing the upper and lower bounds, respectively.

### CLASSICAL OPTIMIZATION

In *classical optimization*,  $Q$  and  $G$  are given explicitly as closed-form expressions of the control variables  $\xi$ . However, design and parameter estimation problems often result in objective functions  $Q$  and constraints  $G$  that are not explicitly expressible by closed-form expressions of the control variables  $\xi$ .

### PDE-Constrained Optimization

In multiphysics modeling, it is often desirable to let control variables parameterize the problem and seek to optimize a function of the PDE solution. The objective function is therefore a function of both the control variables and PDE solution, which is in turn a function of the control variables. The multiphysics problem is a PDE, which after discretization is represented as a system of equations  $L(u(\xi), \xi) = 0$ , where  $u$  is the PDE solution and  $\xi$  the control variables.

The complete PDE-constrained optimization problem solved by the Optimization interface adds the PDE problem as an equality constraint to the general optimization problem:

$$\begin{cases} \min_{\xi} Q(u(\xi), \xi) \\ L(u(\xi), \xi) = 0 \\ \text{lb} \leq G(u(\xi), \xi) \leq \text{ub} \end{cases} \quad (2-2)$$

It is advantageous to single out those constraints given by  $G$  that are defined as explicit expressions of  $\xi$  only, and those that mix  $u$  and  $\xi$ . Hence, the general constraint formulation  $\text{lb} \leq G(u(\xi), \xi) \leq \text{ub}$  above is replaced by two classes of constraints:

$$\begin{aligned} \text{lb}_P &\leq P(\xi, u) \leq \text{ub}_P \\ \text{lb}_\Psi &\leq \Psi(\xi) \leq \text{ub}_\Psi \end{aligned}$$

and the optimization problem in [Equation 2-2](#) can be written as

$$\begin{cases} \min_{\xi} & Q(u(\xi), \xi) \\ & L(u(\xi), \xi) = 0 \\ & \text{lb}_P \leq P(u(\xi), \xi) \leq \text{ub}_P \\ & \text{lb}_\Psi \leq \Psi(\xi) \leq \text{ub}_\Psi \end{cases} \quad (2-3)$$

This is the general form of the optimization problem considered in the Optimization interface. With the interface, specify the objective function and the constraints in terms of expressions that are explicit in  $\xi$  and  $u$ . The relation between  $u$  and  $\xi$ , which is a system of equations written here compactly as  $L(u, \xi) = 0$ , is given by the multiphysics model defined. Furthermore, you can use the interface to let each component of the vectors  $\xi$  and  $u$  be either globally defined or defined only on a specific domain.

#### SPECIFICATION OF THE OBJECTIVE FUNCTION

The objective function may in general be a sum of a number of terms:

$$Q(u, \xi) = Q_{\text{global}}(u, \xi) + Q_{\text{probe}}(u, \xi) + \sum_{k=0}^n Q_{\text{int}, k}(u, \xi)$$

where  $n$  is the space dimension of the multiphysics model and the different contributions in the sum above are defined as follows:

- $Q_{\text{global}}$  is the *global contribution* to the objective function  $Q$ . It is given as one or more general global expressions.
- $Q_{\text{probe}}$  is a *probe contribution* to the objective function  $Q$ . It is a *probe objective* so its definition is restricted to a point on a given geometrical entity. The *probe point* used for the point evaluation is a point given by the user and has to be contained in the domain.
- $Q_{\text{int}, k}$  is an *integral contribution* to the objective function  $Q$ . It is an *integral objective* so its definition is restricted to a set of geometric entities of the same dimension. For integral contribution on points, the integration reduces to a summation.

Several global, probe, and integral contributions can be defined. In such cases, the total global, probe, and integral contribution is given as the sum of the aforementioned global, probe, and integral contributions that are actively selected in the solver feature settings for the optimization.

### SPECIFICATION OF THE CONSTRAINTS

As already mentioned, the constraints  $\text{lb} \leq G(u(\xi), \xi) \leq \text{ub}$  in the general PDE-constrained optimization problem in [Equation 2-2](#) are written as

$$\begin{aligned}\text{lb}_P &\leq P(\xi, u) \leq \text{ub}_P \\ \text{lb}_\Psi &\leq \Psi(\xi) \leq \text{ub}_\Psi\end{aligned}$$

The first row above constitutes the *implicit constraints*, which are given in terms of expressions involving both the solution variables  $u$  and control variables  $\xi$ . The second row constitutes the *explicit constraints*, which are those constraints given by explicit expressions only in the control variables  $\xi$ .



Note

The motivation for this subdivision is computational. To properly account for implicit constraints within an optimization scheme is computationally expensive whereas the explicit constraints are much easier. As an example, in gradient-based optimization methods the successive iterates in the optimization depend on the sensitivities of the solution variables  $u$  with respect to the control variables  $\xi$ .

To calculate this sensitivity is computationally demanding, see [Choosing a Sensitivity Method](#) in the *COMSOL Multiphysics User's Guide*.

Furthermore, the Optimization interface differentiates between the following constraints (in the description that follows,  $n$  denotes the dimension of the multiphysics model): *bound constraints*, *pointwise inequality constraints*, and *integral inequality constraints*, each of which are described below

- *Bounds* or *control variable bounds* are inequality constraints setting lower and upper bounds directly on each control variable degree of freedom. Hence, bound constraints correspond to constraints of the form  $\text{lb} \leq \xi \leq \text{ub}$ .
- *Pointwise inequality constraints* are inequality constraints involving an explicit expression in terms of the control variables. The constraint sets lower and upper bounds on the expression for node points in a set of geometric entities of the same dimension.

- *Global inequality constraints* set upper and lower bounds on a general global expression, possibly involving both the control variables and the PDE solution.
- *Integral inequality constraints* set upper and lower bounds on an integral of an expression, possibly involving the PDE solution and control variables, over a set of geometric entities of the same dimension. For integral inequality constraints on points, the integration reduces to a summation.



Note

The pointwise inequality constraint is actually one constraint for each node point in the geometric entity. In almost all cases of interest there are a large number of node points, so a pointwise inequality constraint results in a large number of discrete inequality constraints.



Note

Global inequality constraints and integral inequality constraints are structurally similar to the objective function and equally expensive to evaluate.


# Theory for the Sensitivity Interface

The Sensitivity Interface theory is described in this section:

- [About Sensitivity Analysis](#)
- [Sensitivity Problem Formulation](#)
- [Theory for Stationary Sensitivity Analysis](#)
- [Specification of the Objective Function](#)
- [Choosing a Sensitivity Method](#)
- [Theory for Time-Dependent Sensitivity](#)
- [Issues to Consider Regarding the Control Variable](#)
- [Issues to Consider Regarding the Objective Function](#)

## *About Sensitivity Analysis*

---

The Sensitivity interface () is special in the sense that it does not contain any physics of its own. Instead, it is a tool that makes it possible to evaluate the sensitivity of a model with respect to almost any variable.

Simulation is a powerful tool in science and engineering for predicting the behavior of physical systems, particularly those that are governed by partial differential equations. However, often a single simulation (or just a few) is not enough to provide sufficient understanding of a system. Hence, a more exploratory process might be needed, such as *sensitivity analysis*, where one is interested in the sensitivity of a specific quantity with respect to variations in certain parameters included in the model. Such an analysis can, for example, be used for estimating modeling errors caused by uncertainties in material properties or for predicting the effect of a geometrical change.

Many times it is possible to reformulate problems of the above type as the problem of calculating derivatives, so differentiation plays a central role in solving such problems. The Sensitivity interface can calculate derivatives of a scalar *objective function* with respect to a specified set of *control variables*. The objective function is in general a function of the solution to a multiphysics problem, which is in turn parameterized by the control variables.

## Sensitivity Problem Formulation

Because the Sensitivity interface does not contain any physics, it is not intended for use on its own. When the interface is added to a multiphysics model, no new equations are introduced, and the set of solution variables remains the same. Instead, an objective function and a control variable can be specified. The physics interface can perform these distinct tasks:

- Select a control variables and set their values
- Define a scalar objective function
- Compute the sensitivities efficiently using the sensitivity solver



Note

The control variables are independent variables whose value is not affected by the solution process, but they are also degrees of freedom (DOFs) stored in the solution vector. When defining a control variable, its *initial value* must be supplied. The initial value is used to initialize the control variable DOFs, which remain fixed during the solution step.

## Theory for Stationary Sensitivity Analysis

Evaluating the sensitivity of a scalar-valued objective function  $Q(\xi)$  with respect to the control variables,  $\xi$ , at a specific point,  $\xi_0$ , can be rephrased as the problem of calculating the derivative  $\partial Q / \partial \xi$  at  $\xi = \xi_0$ . In the context of a multiphysics model,  $Q$  is usually not an explicit expression in the control variables  $\xi$  alone. Rather,  $Q(u(\xi), \xi)$  is also a function of the solution variables  $u$ , which are in turn implicitly functions of  $\xi$ .

The multiphysics problem is a PDE, which after discretization is represented as a system of equations  $L(u(\xi), \xi) = 0$ . If the PDE has a unique solution  $u = L^{-1}(\xi)$ , the sensitivity problem can be informally rewritten using the chain rule as that of finding

$$\frac{d}{d\xi} Q(u(\xi), \xi) = \frac{\partial Q}{\partial \xi} + \frac{\partial Q}{\partial u} \cdot \frac{\partial u}{\partial L} \cdot \frac{\partial L}{\partial \xi}$$

The first term, which is an explicit partial derivative of the objective function with respect to the control variables, is easy to compute using symbolic differentiation. The second term is more difficult. Assuming that the PDE solution has  $N$  degrees of freedom and that there are  $n$  control variables  $\xi_i$ ,  $\partial Q / \partial u$  is an  $N$ -by-1 matrix,  $\partial u / \partial L$  is an  $N$ -by- $N$  matrix (because  $L^{-1}$  is unique), and  $\partial L / \partial \xi$  is an  $N$ -by- $n$  matrix.

The first and last factors,  $\partial Q/\partial u$  and  $\partial L/\partial \xi$  can be computed directly using symbolic differentiation. The key to evaluating the complete expression lies in noting that the middle factor can be computed as  $\partial u/\partial L = (\partial L/\partial u)^{-1}$  and that  $\partial L/\partial u$  is the PDE Jacobian at the solution point:

$$\frac{d}{d\xi} Q(u(\xi), \xi) = \frac{\partial Q}{\partial \xi} + \frac{\partial Q}{\partial u} \cdot \left( \frac{\partial L}{\partial u} \right)^{-1} \cdot \frac{\partial L}{\partial \xi} \quad (2-4)$$

However, actually evaluating the inverse of the  $N$ -by- $N$  Jacobian matrix is too expensive. In order to avoid that step, an auxiliary linear problem can be introduced. This can be done in two different ways, each requiring at least one additional linear solution step.

#### FORWARD SENSITIVITY METHOD

To use the *forward sensitivity* method, introduce the  $N$ -by- $n$  matrix of solution sensitivities

$$\frac{\partial u}{\partial \xi} = \left( \frac{\partial L}{\partial u} \right)^{-1} \cdot \frac{\partial L}{\partial \xi}$$

These can be evaluated by solving  $n$  linear systems of equations

$$\frac{\partial L}{\partial u} \cdot \frac{\partial u}{\partial \xi_i} = \frac{\partial L}{\partial \xi_i}$$

using the same Jacobian  $\partial L/\partial u$ , evaluated at  $u(\xi_0)$ . Inserting the result into [Equation 2-4](#), the desired sensitivities can be easily computed as

$$\frac{d}{d\xi} Q(u(\xi), \xi) = \frac{\partial Q}{\partial \xi} + \frac{\partial Q}{\partial u} \cdot \frac{\partial u}{\partial \xi}$$

#### ADJOINT SENSITIVITY METHOD

To use the *adjoint sensitivity* method, introduce instead the  $N$ -by-1 adjoint solution  $u^*$ , which is defined as

$$u^* = \frac{\partial Q}{\partial u} \cdot \left( \frac{\partial L}{\partial u} \right)^{-1}$$

Multiplying this relation from the right with the PDE Jacobian  $\partial L/\partial u$  and transposing leads to a single linear system of equations

$$\frac{\partial L^T}{\partial u} \cdot u^* = \frac{\partial Q}{\partial u}$$

using the transpose of the original PDE Jacobian.

### *Theory for Time-Dependent Sensitivity*

---

#### **FORWARD SENSITIVITY**

When you enable sensitivity analysis, the time-dependent solvers can compute—in addition to the basic forward solution—the sensitivity of a functional

$$Q = Q(u_\xi, \xi, T) \quad (2-5)$$

with respect to the control variables  $\xi$  evaluated at the final time  $t=T$ . The forward solution  $u_\xi$  is a solution to the parameterized discrete forward problem

$$L(u_\xi, \xi) = N_F \Lambda_\xi \quad M(u_\xi, \xi) = 0 \quad (2-6)$$

where  $\Lambda_\xi$  are the constraint Lagrange multipliers, or (generalized) reaction forces, corresponding to the constraints  $M$ . It is assumed that  $Q$  does not explicitly depend on  $\Lambda_\xi$ .

To compute the sensitivity of  $Q$  with respect to  $\xi$ , first apply the chain rule:

$$\frac{dQ}{d\xi} = \frac{\partial Q}{\partial \xi} + \frac{\partial Q}{\partial u} \frac{\partial u}{\partial \xi} \quad (2-7)$$

In this expression, the sensitivity of the solution with respect to the control variables,  $\partial u / \partial \xi$ , is still an unknown quantity. Therefore, differentiate the forward problem, [Equation 2-6](#), formally with respect to  $\xi$ :

$$D \frac{\partial \dot{u}_\xi}{\partial \xi} + K \frac{\partial u_\xi}{\partial \xi} + N_F \frac{\partial \Lambda_\xi}{\partial \xi} = \frac{\partial L}{\partial \xi} + \frac{\partial N_F}{\partial \xi} \Lambda_\xi \quad N \frac{\partial u_\xi}{\partial \xi} = \frac{\partial M}{\partial \xi}$$

Here,  $D = -\partial L / \partial \dot{u}$ ,  $K = -\partial L / \partial u$ , and  $N = -\partial M / \partial u$  as usual. Assuming that the constraint force Jacobian  $N_F$  is independent of  $\xi$  (that is,  $\partial N_F / \partial \xi = 0$ ), you can write the above relations in matrix form



$$\begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \frac{\partial \dot{u}_\xi}{\partial \xi} \\ \frac{\partial \Lambda_\xi}{\partial \xi} \end{pmatrix} + J \begin{pmatrix} \frac{\partial u_\xi}{\partial \xi} \\ \frac{\partial \Lambda_\xi}{\partial \xi} \end{pmatrix} = \begin{pmatrix} \frac{\partial L}{\partial \xi} \\ \frac{\partial M}{\partial \xi} \end{pmatrix} \quad J = \begin{bmatrix} K & N_F \\ N & 0 \end{bmatrix} \quad (2-8)$$

solve for the sensitivities  $\partial u_p / \partial \xi$  and  $\partial \Lambda_p / \partial \xi$ , with initial conditions  $\partial u_{0\xi} / \partial \xi$  and  $\partial \Lambda_{0\xi} / \partial \xi$ , respectively, and plug them back for evaluation at  $t=T$  into Equation 2-7.

If the number of individual control variables,  $\xi_j$ , is small, Equation 2-8 can be solved for each right-hand side  $[\partial L / \partial \xi_j; \partial M / \partial \xi_j]^T$  with corresponding initial conditions and the solution inserted into Equation 2-7. This is the *forward method*, which in addition to the sensitivity  $dQ/d\xi$  returns the sensitivity of the solution,  $\partial u_\xi / \partial \xi$ .

If there are many control variables and the sensitivity of the solution itself,  $\partial u_\xi / \partial \xi$ , is not required, the *adjoint method* is more efficient.

#### ADJOINT SENSITIVITY

The adjoint sensitivity method is based on using solution variables  $u^*$  and  $U^*$  known as the *adjoint solution*, to rewrite Equation 2-7:

$$\begin{aligned} \frac{dQ}{d\xi} &= \left( \frac{\partial Q}{\partial \xi} - U^* \frac{\partial L}{\partial \xi} \right) \Big|_{t=T} - u^* D \frac{\partial u}{\partial \xi} \Big|_{t=0} - \int_0^T u^* \frac{\partial L}{\partial \xi} dt \\ \frac{d}{dt}(u^* D) - u^* K &= 0 \end{aligned}$$

Note that it has been assumed that

$$\frac{d}{dT} \langle U^* D \frac{\partial u}{\partial \xi} \rangle = 0$$

The homogeneous adjoint equations are solved backward in time and requires “final” conditions for initialization. The final conditions for  $U^*$  and  $u^*$  are computed as:

$$\begin{aligned} U^* D \Big|_{t=T} &= 0 \\ u^* D \Big|_{t=T} &= - \frac{\partial Q}{\partial u} - U^* K \Big|_{t=T} \end{aligned}$$

On this form, only one forward and one backward (adjoint) problem must be solved regardless of the number of control variables, followed by an evaluation of the gradient

for each variable. Obviously, this is much faster than the forward method if the number of variables is large with the drawback that the forward solution must be available at all times during the backward solution of the adjoint. To reduce the memory requirements for this, a checkpointing strategy is employed. This means that at a number of checkpoints, the forward solution is stored in memory such that a hot start of the time-dependent solver can be performed to produce the forward solution in higher resolution between checkpoints when needed. This reduces the memory requirement at the cost of one additional forward solution.

### *Specification of the Objective Function*

---

The objective function may in general be a sum of a number of terms:

$$Q(u, \xi) = Q_{\text{global}}(u, \xi) + Q_{\text{probe}}(u, \xi) + \sum_{k=0}^n Q_{\text{int},k}(u, \xi)$$

where  $n$  is the space dimension of the multiphysics model and the different contributions in the sum above are defined as follows:

- $Q_{\text{global}}$  is the *global contribution* to the objective function  $Q$ . It is given as one or more general global expressions.
- $Q_{\text{probe}}$  is a *probe contribution* to the objective function  $Q$ . It is a *probe objective* so its definition is restricted to a point on a given geometrical entity. The *probe point* used for the point evaluation is a point given by the user and has to be contained in the domain.
- $Q_{\text{int},k}$  is an *integral contribution* to the objective function  $Q$ . It is an *integral objective* so its definition is restricted to a specific set of geometrical entities of the same dimension. For integral contributions on points, the integration reduces to a summation.

Several global, probe, and integral contributions can be defined. In such cases, the total global, probe, and integral contribution is given as the sum of the aforementioned global, probe, and integral contributions that are actively selected in the solver feature settings for the optimization.

## *Choosing a Sensitivity Method*

---

To evaluate sensitivities as part of a multiphysics problem solution, an auxiliary linear problem must be solved, in addition to the original equation. Choose between these methods:

- Select the *forward sensitivity* method to evaluate the derivatives of all solution variables and an optional objective function.
- Select the *adjoint sensitivity* method to look only at derivatives of a scalar objective function.

### **FORWARD SENSITIVITY**

Use the forward sensitivity method to solve for the derivatives of all dependent variables, plus an optional scalar objective function, with respect to a small number of control variables. The forward method requires one extra linear system solution for each control variable.

The linear system that must be solved is the same as the last linearization needed for solving the forward model. Thus, when using a direct solver (for example, PARDISO) the extra work amounts only to one back-substitution per control variable DOF. The iterative linear and segregated solvers can reuse preconditioners and other data but must otherwise perform a complete solution each time.

### **ADJOINT SENSITIVITY**

The adjoint method solves for the derivatives of a single scalar objective function with respect to any number of control variables, requiring only one single additional linear system solution. In addition to the objective function gradient, the discrete adjoint solution is computed. This quantity represents the sensitivity of the objective function with respect to an additional generalized force applied as a nodal force to the corresponding solution component.

The auxiliary linear system is in this case the transpose of the last linearization needed for solving the forward model. The MUMPS and PARDISO linear solvers can solve the transposed problem at the cost of a back-substitution, while the SPOOLES linear solver needs to do a new factorization if the problem is not symmetric or Hermitian.

The iterative solvers can reuse most preconditioning information as can the segregated solver, which, however, loops over the segregated steps in reversed order.



Sensitivity analysis can be used together with all stationary and parametric standard solvers and with the BDF solver for transient studies. The available solvers are described in the section [Solvers and Study Types](#). For technical details about the solution procedure, see the *COMSOL Multiphysics Reference Guide*.

---

### *Issues to Consider Regarding the Control Variable*

---

#### **THE EFFECT OF DISCRETIZATION**

The sensitivity analysis is always performed on the discretized system of equations. As already mentioned, the control variable can be a scalar/vector or an element in some infinite-dimensional function space. In the latter case it is represented on the finite element mesh, just like the solution variables, or global scalar quantities. When using a control variable field represented on the finite element mesh, the sensitivities are therefore associated with individual control variable degrees of freedom rather than with the field value at each point. This makes it difficult to interpret the result. For example, if a domain control variable is set up using a first-order Lagrange shape function representation to control the material density in your model, the solution contains the sensitivity of the objective function with respect to the discrete density value at each node point in the mesh. Because each node influences the density in a small surrounding region, the size of which varies from node to node, the individual sensitivities are not directly comparable to each other.

Displaying such domain control variables results in a plot that is not smooth due to the varying element size. It must therefore not be used to draw any conclusions about the physics and the effect of changing the physical field represented by the control variable. Some insight may, however, be gained by looking at the sensitivities divided by the mesh volume scale factor `dvo1`. This makes the sensitivities in the plot comparable between different parts of the surface but still not mathematically well defined. In

particular, using discontinuous constant shape functions together with the division by `dv01` results in a plot that is proportional to the true pointwise sensitivity.



Note

If the plan is to use the sensitivities in an automatic optimization procedure, as is done through the Optimization interface available with the optional Optimization Module, the discrete nature of the sensitivities causes no additional complication. The optimization solver searches for optimum values of the discrete control variables using the discrete gradient provided by the sensitivity analysis.

### GEOMETRICAL SENSITIVITY

Use the control variables directly to parameterize any aspect of the physics that is controlled by an expression. This applies, for example, to material properties, boundary conditions, loads, and sources. However, the shape, size, and position of parts of the geometry cannot be changed as easily at solution time, and therefore require special attention.

Control variables cannot be used directly in the geometry description. Instead, the model must be set up using the *ALE* (arbitrary Lagrangian-Eulerian) method and tie all physics to an *ALE frame* controlled by a Deformed Geometry physics interface (in 2D only) or a Moving Mesh (ALE) physics interface. Then use control variables to control the mesh movement, effectively parameterizing the geometry.



Note

See [The Deformed Geometry and Moving Mesh Interfaces](#) in the *COMSOL Multiphysics User's Guide* for details about these physics interfaces and the ALE method in general.

## *Issues to Consider Regarding the Objective Function*

### THE PRINCIPLE OF VIRTUAL WORK

*Potential energy* has a special status among scalar objective functions, because its derivatives with respect to scalar control variables can in many cases be interpreted as (true or generalized) forces.

### COMPLEX-VALUED OBJECTIVE FUNCTIONS

Sensitivity analysis can be applied only when the objective function is a real-valued differentiable function of the control variables. This is usually not a very severe

constraint, even for frequency-domain models where the PDE solution variables are complex valued. The reason is that physical quantities of interest to the analyst are always real valued, and if complex-valued control variables are required, it is possible to treat the real and imaginary parts separately.


Many common quantities of interest are time averages that can be written in the form  $Q = \text{real}(a \cdot \text{conj}(b))$ , where  $a$  and  $b$  are complex-valued linear functions of the solution variables and therefore implicit functions of the control variables. The problem with this expression is that while  $Q$  is indeed a real-valued differentiable function of the control variables, it is not an analytical function of  $a$  and  $b$ . This complicates matters slightly because the sensitivity solver relies on partial differentiation and the chain rule.

While the partial derivatives of  $Q$  with respect to  $a$  and  $b$  are, strictly speaking, undefined, it can be proven that if they are chosen such that

$$Q(a + \delta a, b + \delta b) \approx Q(a, b) + \text{real}\left(\frac{\partial Q}{\partial a} \delta a + \frac{\partial Q}{\partial b} \delta b\right) \quad (2-9)$$

for any small complex increments  $\delta a$  and  $\delta b$ , the final sensitivities are evaluated correctly. The special function `realdot(a, b)` is identical to `real(a*conj(b))` when evaluated but implements partial derivatives according to [Equation 2-9](#). For that reason, use it in the definition of any time-average quantity set as objective function in a sensitivity analysis.

# The Optimization Interface

The Optimization interface, found under the **Mathematics>Optimization and Sensitivity** branch (  ) in the **Model Wizard**, is designed to facilitate setting up and solving optimization problems.





To optimize a model, add the Optimization interface along with the physics interfaces in the model. The Optimization interface lets you set objective function, constraints, and bounds and to introduce the control variables.

In this chapter:

- [Adding Optimization to a Model](#)
- [The Optimization Interface](#)

# Adding Optimization to a Model


Add a **Optimization** interface when creating a new model or at any time during modeling. For a new model, physics interfaces are selected as the second step in the **Model Wizard** (after specifying the space dimension). In an active model, right-click a **Model** node in the Model Tree and choose **Add Physics**. In both cases, the list of physics interfaces appears. Take the following steps to add an Optimization interface to the model:

- 1 Expand the **Mathematics>Optimization and Sensitivity** node in the list of physics interfaces.
- 2 Select **Optimization** (  ).
- 3 Click the **Add Selected** button (  ) underneath the list to add the selected physics interface to the model. The physics interface then appears in the list under **Selected physics**.
- 4 Click the **Next** button (  ) in the upper-right corner of the **Model Wizard** window.
- 5 Optionally, choose an study type for the optimization on the **Select Study Type** page.
- 6 Click the **Finish** button (  ) in the upper-right corner of the **Model Wizard** window.

Also, to include the Optimization solver in the generated solver sequence, select the **Optimization** check box in the **Study Extensions** section, which you find in the settings window for the Stationary and Time Dependent study types.



# The Optimization Interface

The **Optimization** interface () has the tools for defining and solving optimization problems. The main purpose of the interface is its ability to solve PDE-constrained optimization problems.

You can use the Optimization interface to define objective functions and constraints in terms of control and solution variables (the latter are given as the solution to the differential equations defined by the multiphysics model) and restrict these to specific domains or make them globally available. This flexibility is reflected in the user interface by grouping these settings according to the dimension of the geometric entity to which they apply. In such a group of settings, the following settings can be specified, to each of which corresponds a separate feature and its settings window:

- [Least-Squares Objective](#)
- [Integral Objective](#)
- [Probe Objective](#)
- [Integral Inequality Constraint](#)
- [Pointwise Inequality Constraint](#)
- [Control Variable Field](#) (which includes the settings for the associated bound constraints)

## INTERFACE IDENTIFIER

The interface identifier is a text string that can be used to reference the respective physics interface if appropriate. Such situations could occur when coupling this interface to another physics interface, or when trying to identify and use variables defined by this physics interface, which is used to reach the fields and variables in expressions, for example. It can be changed to any unique string in the **Identifier** field.

The default identifier (for the first interface in the model) is **opt**.

## DOMAIN SELECTION

The default setting is to include **All domains** in the model to the control variables and expressions that enter into the formulation of the optimization problem. To choose specific domains, select **Manual** from the **Selection** list.



See Also

## Theory for the Optimization Interface

### *Least-Squares Objective*

For creating a least-squares objective, you import an **Experimental Data** file containing comma-separated or semicolon-separated columns of measurement data from a single experiment. Each **Least-Squares Objective** feature corresponds to an experiment where the measurements have been obtained using given values for a set of **Experimental Parameters** (for example, the temperature during the experiment). The squared sum of the difference between the measurement values and the corresponding expressions evaluated in the model, when solved for the given parameter values, is added as a contribution to the total least-squares objective function. Right-click the node to add column subnodes—**Value Column**, **Time Column**, **Parameter Column**, **Coordinate Column**, and **Ignored Column**—assigning meaning to the individual columns as values, times, parameter values, coordinate data, or values to ignore, respectively. One column subnode must be added for each column in the data file and in the same order as the columns appear in the file.




Tip




Move column nodes up and down using the context menu or a keyboard combination of the Ctrl key and an arrow key.

### EXPERIMENTAL DATA

Enter a **Filename** or click the **Browse** button to specify a measurement data file containing comma-separated or semicolon-separated columns of measurements in the dialog box that opens. The files are typically CSV files (\*.csv), data files (\*.dat), or plain text files (\*.txt).

### EXPERIMENTAL PARAMETERS

Click the **Add** button (  ) below the table to add an experimental parameter. The experimental parameters are separate from any other parameters used in the least-squares data, so you cannot use these experimental parameters for other

parameterizations in the model. Experimental parameters are useful for including additional parameters that represent model conditions for the experimental data and that are valid for the current experimental data file. In the **Name** column, choose a parameter name from the global parameters defined in the model. Enter a global-scope expression or value in the **Expression** column to assign a value to the parameter in this experiment. Use the **Load from File** () and **Save to File** () buttons to load and save experimental parameter names and expressions from and to a file. Use the **Delete** button () to remove the selected parameter from the table.

### *Value Column*

---

Add a **Value Column** node to identify a column in the experimental data file as containing measurement values. Enter a corresponding **Expression**, which must be available for evaluation according to the selection in the current model. Enter a corresponding **Column contribution weight**, which must strictly positive and be available for evaluation in the global scope in the current model. Optionally a **Variable name** can be specified to enable access to the data from the file for postprocessing. The difference between the **Expression** and the value from the file will be squared and multiplied with the **Column contribution weight** and a factor 0.5 to give the contribution to the total objective for each measured value.

### *Time Column*

---

Add a **Time Column** node to identify a column in the experimental data file as containing the times at which measurements in the value columns were made. When computing the total least-squares objective value, the value column expressions are evaluated at these times in a forward transient solution.

### *Parameter Column*

---

Add a **Parameter Column** node to identify a column in the experimental data file as containing the parameter values for which measurements in the value columns were made. When computing the total least-squares objective value, the value column expressions are evaluated for these parameter values. The **Parameter name** has to correspond to one of the global parameters defined in the model.

### *Coordinate Column*

---

Add a **Coordinate Column** node to identify a column in the experimental data file as containing the global coordinates at which measurements in the value columns were made. Select the applicable coordinate and frame type from the **Coordinate** and **Frame** lists. The number of coordinates must correspond to the number of dimensions in the model.

### *Ignored Column*

---

Add an **Ignored Column** node to identify a column in the experimental data file that should not be used.

### *Integral Objective*

---

An **Integral Objective** is defined as the integral of a closed form expression of control and solution variables (the latter are given as the solution to the differential equations defined by the multiphysics model) that are either global or available in the domain in question. Hence, its definition is restricted to a specific set of geometric entities of the same dimension. For integral objectives on points, the integration reduces to a summation.

#### **DOMAIN, EDGE, BOUNDARY, OR POINT SELECTION**

From the **Selection** list, choose the geometric entity (domains, boundaries, edges, or points) used in the integration for the integral objective.

#### **OBJECTIVE**

Enter an **Objective expression** that is integrated over the geometric entity level in the integral objective.

#### **QUADRATURE SETTINGS**

Specify the settings for the **Quadrature** used to numerically evaluate the integral in the integral objective: the integration order (default: 4) in the **Integration order** field and the frame to integrate on (default: the spatial frame), which is selected from the **Integrate on frame** list.

## *Probe Objective*

---

A **Probe Objective** is defined as a point evaluation of a closed form expression of control and solution variables (the latter are given as the solution to the differential equations defined by the multiphysics model) that are either global or available in the domain in question. The point used for the point evaluation has to be contained in the domain.

### **DOMAIN SELECTION**

From the **Selection** list, choose the domain containing the point used for the point evaluation.

### **OBJECTIVE**

Enter an **Objective expression** that is evaluated at the point in the domain.

### **PROBE COORDINATES**

Specify the **Probe coordinates** for the point in the domain where the expression for the objective is evaluated. After specifying the probe coordinates, select an option from the **Evaluate in frame**—**Spatial** (the default), **Material**, or **Mesh**.

## *Integral Inequality Constraint*

---

**Integral Inequality Constraints** are given as restrictions to the values of the integral of a closed form expression taken over a specific set of geometric entities of the same dimension.

The expression is a closed form expression of control and solution variables (the solution variables are given as the solution to the differential equations defined by the multiphysics model) that are either global or available in the domain in question. For integral inequality constraints on points, the integration reduces to a summation.

### **DOMAIN, EDGE, BOUNDARY, OR POINT SELECTION**

From the **Selection** list, choose the geometric entity (domains, boundaries, edges, or points) used in the integration for the integral inequality constraints.

### **CONSTRAINT**

Enter a **Constraint expression** that is integrated over the domain in the integral inequality constraint.

## QUADRATURE SETTINGS

Specify the settings for the **Quadrature** used to numerically evaluate the integral in the integral objective: the integration order (default: 4) in the **Integration order** field and the frame to integrate on (default: the spatial frame), which is selected from the **Integrate on frame** list.

## BOUNDS

By default, the **Lower bound** and **Upper bound** check boxes are selected to activate the required bounds. To specify equality constraints, simply make sure the upper and lower bounds have the same value.

### *Pointwise Inequality Constraint*

---

A **Pointwise Inequality Constraint** is given as a restriction to the values of a closed form expression at *all* points in a set of geometric entities of the same dimension. Due to computational issues, the expression has to be a closed-form expression of *only* control variables. Furthermore, only those control variables that are either global or available in the domain in question are usable.

## DOMAIN, EDGE, BOUNDARY, OR POINT SELECTION

From the **Selection** list, choose the geometric entity (domains, boundaries, edges, or points) to apply a pointwise inequality constraint.

## CONSTRAINT

Enter a **Constraint expression** for the pointwise inequality constraint.

## DISCRETIZATION

This section contains settings for the element used to discretize the control variable. Select a **Shape function type**: **Lagrange** (the default) or **Discontinuous Lagrange**. Also select an **Element order**: **Linear**, **Quadratic** (the default), **Cubic**, **Quartic**, or **Quintic**. Specify the **Value type when using splitting of complex variables**—**Real** or **Complex** (the default).

## BOUNDS

By default, the **Lower bound** and **Upper bound** check boxes are selected to activate the required bounds. To specify equality constraints, make sure that the upper and lower bounds have the same value.

## *Control Variable Field*

---

Specify the **Control Variable Field** specific to the geometric entity level (domain, edge, boundary, or point) in question. Right-click the node to add a **Control Variable Bounds** feature.

### **DOMAIN, EDGE, BOUNDARY, OR POINT SELECTION**

From the **Selection** list, choose the geometric entity (domains, boundaries, edges, or points) where the control variable field is defined.

### **CONTROL VARIABLE**

Enter a **Control variable name** and **Initial value**.

### **DISCRETIZATION**

This section contains settings for the element used to discretize control variables.

Select a **Shape function type**: **Lagrange** (the default) or **Discontinuous Lagrange**. Also select an **Element order**: **Linear**, **Quadratic** (the default), **Cubic**, **Quartic**, or **Quintic**. Specify the **Value type when using splitting of complex variables**—**Real** or **Complex** (the default).

## *Global Objective*

---

Specify the **Global Objective** contribution to the function.

### **OBJECTIVE**

Enter an **Objective expression** that defines the contribution to the objective function. It can be an expression of those components of the control and solution variables (the solution variables are given as the solution to the differential equations defined by the multiphysics model) that are globally available.

## *Global Least-Squares Objective*

---

The **Global Least-Squares Objective** is similar to the **Least-Squares Objective** (see [Least-Squares Objective](#)) but is intended for global control variables and therefore does not contain selections and **Coordinate Column** subnodes.



See [Material Property Fitting](#) for an example of fitting material properties to measured data using a global least-squares objective: Model Library path **Optimization\_Module/Tutorial\_Models/material\_property\_fitting**

## *Global Inequality Constraint*

---

Specify the **Global Inequality Constraint** that applies globally. Due to computational issues, the expression has to be a closed form expression of *only* control variables. Furthermore, only global control variables are usable.

### **CONSTRAINT**

Enter a **Constraint expression** whose values at all points in all domains are to be constrained.

### **BOUNDS**

By default, the **Lower bound** and **Upper bound** check boxes are selected to activate the required bounds. To specify equality constraints, simply make sure the upper and lower bounds have the same value.






## *Global Control Variables*

---

Specify those components of the **Global Control Variable** that are globally available.


### **CONTROL VARIABLES**

In the table, enter **Variable** names, **Initial values**, and **Lower** and **Upper Bounds** of global control variables. To specify equality constraints, simply make sure the upper and lower bounds have the same value.

Move control variable rows up and down using the **Move Up** (  ) and **Move Down** (  ) buttons. To remove a control variable, select some part of that variable's row in the table and click the **Delete** button (  ). Also save the definitions of the global control variables to a text file by clicking the **Save to File** button (  ) and using the **Save to File** dialog box that appears. To load a text file with global control variables, use the **Load from File** button (  ) and using the **Load from File** dialog box that appears. Data must be separated by spaces or tabs.



# The Sensitivity Interface



The Sensitivity interface, found under the **Mathematics>Optimization and Sensitivity** branch (  ) in the **Model Wizard**, is designed to facilitate setting up and solving sensitivity problems.

To find the sensitivity of a model, add the Sensitivity interface along with the physics interfaces in the model. The Sensitivity interface lets you set objective function and to introduce the control variables.

In this chapter:

- [The Sensitivity Interface](#)

# The Sensitivity Interface

The **Sensitivity** interface () found under the **Mathematics>Optimization and Sensitivity** () branch in the **Model Wizard**, has the tools for defining and solving sensitivity problems.



For a more extensive introduction to the mathematics implemented by this interface, see the [Theory for the Sensitivity Interface](#).

The **Sensitivity** interface includes the tools for defining and solving sensitivity problems. The main purpose of the interface is its ability to solve stationary and time-dependent PDE-constrained sensitivity problems.

The objective functions and constraints are defined in terms of control and solution variables (the latter are given as the solution to the differential equations defined by the multiphysics model) and restrict these to specific domains or make them globally available. This flexibility is reflected in the user interface by grouping these settings according to the dimension of the domain to which they apply. In such a group of settings, the following settings can be specified, to each of which corresponds a separate feature and its settings window:

- [Integral Objective](#)
- [Probe Objective](#)
- [Control Variable Field](#)

The main Sensitivity interface node's settings window contains the following section:

## INTERFACE IDENTIFIER

The interface identifier is a text string that can be used to reference the respective physics interface if appropriate. Such situations could occur when coupling this interface to another physics interface, or when trying to identify and use variables defined by this physics interface, which is used to reach the fields and variables in expressions, for example. It can be changed to any unique string in the **Identifier** field.

The default identifier (for the first interface in the model) is **sens**.



See Also

- [Global Objective](#)
- [Global Control Variables](#)
- [Adding a Sensitivity Interface to a Model](#)

---

### *Integral Objective*

---

An **Integral Objective** is defined as the integral of a closed form expression of control and solution variables (the latter are given as the solution to the differential equations defined by the multiphysics model) that are either global or available in the domain in question. Hence, its definition is restricted to a set of geometric entities of the same dimension. For integral objectives on points, the integration reduces to a summation.

#### **DOMAIN, EDGE, BOUNDARY, OR POINT SELECTION**

From the **Selection** list, choose the geometric entity (domains, boundaries, edges, or points) used in the integration for the integral objective.

#### **OBJECTIVE**

Enter an **Objective expression** that is integrated over the geometric entity level in the integral objective.

#### **PAIR SELECTION**

If **Integral Objective** is selected from the **Pairs** menu, choose the pair to define. An identity pair has to be created first. Ctrl-click to deselect.

#### **QUADRATURE SETTINGS**

Specify the settings for the **Quadrature** used to numerically evaluate the integral in the integral objective: the integration order (default: 4) in the **Integration order** field and the frame to integrate on (default: the spatial frame), which is selected from the **Integrate on frame** list.

---

### *Probe Objective*

---

A **Probe Objective** is defined as a point evaluation of a closed form expression of control and solution variables (the latter are given as the solution to the differential equations defined by the multiphysics model) that are either global or available in the domain in question. The point used for the point evaluation has to be contained in the domain.

**DOMAIN SELECTION**

From the **Selection** list, choose the domain containing the point used for the point evaluation.

**OBJECTIVE**

Enter an **Objective expression** that is evaluated at the point in the domain.

**PROBE COORDINATES**

Specify the **Probe coordinates** for the point in the domain where the expression for the objective is evaluated. After specifying the probe coordinates, select an option from the **Evaluate in frame**—**Spatial** (the default), **Material**, or **Mesh**.

*Control Variable Field*

---

Specify the **Control Variable Field** specific to the geometric entity level (domain, edge, boundary, or point) in question.

**DOMAIN, EDGE, BOUNDARY, OR POINT SELECTION**

From the **Selection** list, choose the geometric entity (domains, boundaries, edges, or points) where the control variable field is defined.

**PAIR SELECTION**

If **Control Variable Field** is selected from the **Pairs** menu, choose the pair to define. An identity pair has to be created first. Ctrl-click to deselect.



See Also

In the *COMSOL Multiphysics User's Guide*:

- [Identity and Contact Pairs](#)
  - [Specifying Boundary Conditions for Identity Pairs](#)
- 

**CONTROL VARIABLE**

Enter a **Control variable name** and **Initial value**.

**DISCRETIZATION**

This section contains settings for the element used to discretize the control variable. Select a **Shape function type**: **Lagrange** (the default) or **Discontinuous Lagrange**. Also select an **Element order**: **Linear**, **Quadratic** (the default), **Cubic**, **Quartic**, or **Quintic**.

## *Global Objective*

---

Specify the **Global Objective** contribution to the function.

### **OBJECTIVE**

Enter an **Objective expression** that defines the contribution to the objective function. It can be an expression of those components of the control and solution variable (the solution variable is given as the solution to the differential equations defined by the multiphysics model) that are globally available.



## *Global Control Variables*

---

Specify any globally available control variables.

### **CONTROL VARIABLES**




In the table, enter **Variable** names and **Initial values** of the control variables that are globally available.


To add a control variable to the table, click the **Add** button (  ). To remove a control variable and its values from the table, click the **Delete** button (  ).

## *Adding a Sensitivity Interface to a Model*

---

You can add a **Sensitivity** interface when creating a new model or at any time during modeling. For a new model, select physics interfaces as the second step in the **Model Wizard** (after specifying the space dimension). In an active model, right-click the model node in the **Model Builder** and select **Add Physics**. In both cases, the **Add Physics** page appears with a list of interfaces.

- 1 Expand the **Mathematics>Optimization and Sensitivity** node in the list of physics interfaces.
- 2 Select **Sensitivity** (  ).
- 3 Click the **Add Selected** button (  ) underneath the list to add the selected physics interface to the model. The physics interface then appears in the list under **Selected physics**.
- 4 When you are ready click the **Next** button (  ) in the upper-right corner of the **Model Wizard** window.
- 5 Optionally, choose a study type for the sensitivity analysis on the **Select Study Type** page.


6 Click the **Finish** button (  ) in the upper-right corner of the **Model Wizard** window.

You also need to add a **Sensitivity** subnode to the **Solver** node in the solver configuration to fully define the sensitivity analysis.


# The Optimization Solvers

The Optimization interface is designed to facilitate setting up and solving optimization problems. This chapter provides information about the settings and properties of the optimization solvers.

In this chapter:

- [The Optimization Solver Settings](#) 
- [Optimization Solver Properties](#)

# The Optimization Solver Settings

The **Optimization Solver** () solver node has the settings necessary for solving PDE-constrained optimization problems using the SNOPT and Levenberg-Marquardt solvers.



Note

This section describes **Solver** features available with the Optimization Module. See also [Solvers and Study Types](#) in the *COMSOL Multiphysics User's Guide* for more information about solvers in general.



See Also

For a more extensive introduction to the mathematics implemented by this interface, see the [Theory for the Optimization Interface](#).

## GENERAL

Specify those parts of the solver settings that are independent of the particular method employed by the solver.

### *Optimality Tolerance*

Specify the **Optimality tolerance**, which has the default value 1e-6.

For SNOPT, this parameter, which is called the *major optimality tolerance* in the *SNOPT User's Guide*, is used by the linear and quadratic solvers to determine, on the basis of the reduced-gradient size, whether optimality has been reached. More precisely, it regulates the accuracy to which the final iterate in SNOPT is required to fulfill the first-order conditions for optimality. In this context, the declaration of optimality in SNOPT means that the final iterate satisfies the first-order optimality conditions for [Equation 2-1](#) up to a user-defined tolerance level. Then the major optimality tolerance specifies the parameter  $\tau_D$  introduced in section 2.11 of [Ref. 2](#), which in turn enters into the definition of a tolerance used to make sure the first-order conditions are satisfied.

When Levenberg-Marquardt is used, the maximum absolute value of the gradient of the objective is used to determine when optimality has been reached. More precisely, the solver stops iterating when



$$\|\nabla F\|_{\infty} \frac{\|x\|_{\infty}}{|F|}$$

is below the optimality tolerance, where  $F$  is the objective and  $x$  is the control variables.

This is an important setting and it can play tricks on you if your objective function or your optimization variables are badly scaled. Preferably, the objective function and scalar constraints, as well as the optimization variables, should be of the same order, and ideally of order 1. Tweaking the **Optimality tolerance** parameter might be necessary if you are confronted with problems related to convergence. As an example, if the optimization solver reports a converged solution after just a few iterations, try to restart it with a tighter tolerance to make sure it has actually found the solution. If, on the contrary, it seems to iterate forever—despite the value of the objective function having converged (check the output on the **Log** page in the **Progress** window)—chances are that the tolerance value is too strict. Examples of scenarios of the latter type are the following:

- The warning message “Requested accuracy could not be achieved” refers to the case when a feasible solution has been found, but the requested accuracy not be achieved. Hence, an abnormal termination has occurred, but the solver is within good reach of satisfying the **Optimality tolerance**. If this happens, check that the **Optimality tolerance** is not too small.
- The warning message “The current point cannot be improved upon” can occur in cases when the objective or constraint evaluation requires an expensive iterative process. In such case the evaluation might be accurate to rather few significant figures, and gradients are probably available but unreliable. Theoretically the **Optimality tolerance** should not be set smaller than the square-root of the function precision. The latter is the expected stability of the numerical model rather than its accuracy as a model of physical reality. When using a direct linear solver on a linear model, the function precision should be set to a value of the same order as the inverse of the condition number. For a nonlinear or iterative solver, you can expect the precision to be of the same order as the solver tolerances, which is the numerical precision in the evaluation of the objective and constraints. Furthermore, even when you set the **Optimality tolerance** according to the function precision, the same exit

condition might occur. At present, the only remedy is to increase the accuracy of the function calculation.



Note

The final SNOPT iterate is not guaranteed to be a constrained local minimizer for Equation 2-1 despite a successful run. For example, the constraint qualification might not hold at the final iterate. Similarly, the final iterate might satisfy the first-order but not the second-order conditions for optimality. Verifying second-order conditions requires second derivatives. See section 2.11 in Ref. 2 and p. 76 of the *SNOPT User's Guide* (Ref. 1) for further details.

#### *Maximum Number of Objective Evaluations*

Specify the **Maximum number of objective evaluations**, which has default of 500. This number limits the number of times the objective function is evaluated, which in most cases is related to the number of times the multiphysics system is simulated for different values of the optimization control variable. Note, however, that it is not equal to the number of iterations taken by the optimizer because each iteration may invoke more than a single objective function evaluation. Furthermore, by setting this parameter to a smaller value and calling the optimization solver repeatedly, you can study the convergence rate and stop when further iterations with the optimization solver no longer have any significant impact on the value of the objective function.

### **OPTIMIZATION SOLVERS**

This section contains settings related to the numerical methods that the solvers use.

#### *Method*

The two available choices are **SNOPT** (the default) and **Levenberg-Marquardt**. The Levenberg-Marquardt method can only be used for problems of least-squares type without bounds on the control variables, while SNOPT can solve any type of optimization problem.

#### *Objective Contributions*

When **SNOPT** is used, the expression used as objective function can be controlled through this setting. The default is **All**, in which case the sum of all objective contributions present in the model are used as objective function. By selecting **Manual**, you can enter an expression that will be used as the objective function in the **Objective expression** field. The expression **all\_obj\_contrib** means that all objective contributions present in the model will be used. Hence, this expressions is equivalent to selecting **All**.

When you use **Levenberg-Marquardt**, the objective function is the sum of all least-squares objective contributions present in the model. The cost function is

$$V(\bar{\eta}) = \frac{1}{2} \sum_{m=1}^M \sum_{j=1}^{J_m} \sum_{k=1}^{K_{jm}} w_{jm} f_{jm}^2(\overline{x_{jmk}}, \bar{u}_m(\bar{x}, p_{jm}, \bar{\eta}), \bar{\eta}, \bar{C}_m)$$

where  $M$  is the number of *series* (measurement series),  $J_m$  is the number of *measurements*, and  $K_{jm}$  is the number of *points*. The variable  $\bar{x}$  is the space dimension,  $\bar{\eta}$  are the parameters for which the cost function should be minimized, and  $\bar{u}_m(\bar{x}, p, \bar{\eta})$  solves a given PDE/ODE. The variable  $p$  is time if the PDE/ODE is time-dependent but it may also represent any parameter when the forward problem is stationary. The functions  $w_{jm}$  are weight functions and  $f_{jm}$  represent the difference between some model function  $g_{jm}$  and some measured data  $\hat{g}_{jmk}$ ; that is,  $f_{jm}$  can be written as

$$f_{jm}(\overline{x_{jmk}}, \bar{u}_m(\bar{x}, p_{jm}, \bar{\eta}), \bar{\eta}, \bar{C}_m) = g_{jm}(\overline{x_{jmk}}, \bar{u}_m(\bar{x}, p_{jm}, \bar{\eta}), \bar{\eta}, \bar{C}_m) - \hat{g}_{jmk}$$

#### Gradient Method

SNOPT and Levenberg-Marquardt are gradient-based methods. The gradient can be computed according to the choices **Automatic**, **analytic** (default for SNOPT); **Forward**; **Adjoint**; and **Numeric** (default for Levenberg-Marquardt). When **Automatic**, **analytic** is chosen, either the *adjoint method* or the *forward method* is used to compute the gradient analytically. The adjoint method is used when the number of optimization degrees of freedom is larger than the number of objective functionals plus two, otherwise the forward method is used. It is also possible to explicitly choose to use either the adjoint or forward method using the corresponding alternatives from the **Gradient method** list. When **Numeric** is chosen, finite differences are used to compute the gradient numerically.



Note

When the number of control variables is large, calculating the gradient numerically or with forward sensitivity can be time consuming.

For the Levenberg-Marquardt method you can choose the **Gradient approximation order**. Selecting **First** gives a less accurate gradient, while selecting **Second** gives a better approximation of the gradient. However, **Second** requires twice as many evaluations of

the objective function for each gradient compared to **First**. In many applications, the increased accuracy obtained by choosing **Second** is not worth this extra cost.

#### *Difference Interval*

Specify the relative magnitude of the numerical perturbation to use for first-order gradient approximation.

#### *Central Difference Interval*

Specify the relative magnitude of the numerical perturbation to use for second-order gradient approximation.

#### *Initial Damping Factor*

The Levenberg-Marquardt method controls the step length and direction through a positive numerical scalar. A value close to zero means that the optimization solver takes a step close to a full Gauss-Newton step. A large value means that it takes a small step close to the steepest-descent direction. The Levenberg-Marquardt method controls this factor internally and tries to have as small factor as possible, but the initial value can be controlled by altering the value here. A small value means that the solver tries to be aggressive initially, while a large value means that the solver is more cautious.

#### *QP Solver*

You have the possibility to specify which solver to use within SQOPT for solving the QP subproblems that are formed during each major SQP iteration. The following active-set algorithms for the QP subproblem are available:

- **Cholesky**—This option holds the full Cholesky factor of the reduced Hessian. As the QP iterations (minor iterates) proceed, the dimension of the Cholesky factor changes with the number of superbasic variables. If the number of superbasic variables increases beyond the value of reduced Hessian dimension, the reduced Hessian cannot be stored and the solver switches to conjugate gradient.
- **Conjugate gradient**—This method uses an active-set method similar to quasi-Newton but uses the conjugate-gradient method to solve all systems involving the reduced Hessian. It can be appropriate when the number of superbasics is large but each QP subproblem requires relatively few minor iterations. A limited-memory procedure is used to store a fixed number of BFGS update vectors and a diagonal Hessian approximation.
- **Quasi-Newton**—This method implements the quasi-Newton method using a quasi-Newton approximate Hessian. It can be an appropriate choice when the

number of superbasics is large but each QP subproblem requires relatively few minor iterations.



See Also

- Find more details in the references listed in the [Theory for the Optimization Interface](#).
- See page 80 of the *SNOPT User's Guide* ([Ref. 1](#) in the [Theory for the Optimization Interface](#)) for further details.

#### Use Step Condition

In the **Use step condition** field you can enter an expression that tells the optimization solver to reduce the step length in the current line search used by SNOPT to generate the next iterate.

The solver uses the condition to restrain the iterates from entering into areas in the control-variable space where the PDE problem is not well defined. A typical example is when a mesh element becomes inverted during geometry optimization using a Moving Mesh (ALE) interface. The step limit condition that identifies this situation takes the form `minqual1_a1e-0.05`, where 0.05 is a threshold value for the mesh quality. This step limit condition has a direct analog in the stop condition for the time-dependent and parametric solvers.



Tip

Only use the step limit condition as a last resort to keep the optimization solver in a feasible region. Instead, if possible, use pointwise constraints on the optimization variables to enforce the condition.

When the step limit condition is violated, the solver reduces the line-search step until an acceptable point is found. However, because no Jacobian is computed for the step limit condition, there is no mechanism to prevent the solver from immediately attempting another step in the same infeasible direction. As a result, the solver might get stuck at the same point without converging until it reaches the maximum number of model evaluations or you stop the iteration manually.

#### RESULTS WHILE SOLVING

Select the **Plot** check box to plot the results while solving the model. Select a **Plot group** from the list and any applicable **Probes**.

#### LOG

The **Log** displays the information about the progress of the solver.

# Optimization Solver Properties

This section provides detailed explanations of the properties that control the behavior of SNOPT—the main optimization solver that comes with the Optimization Module.



These properties are available when using the COMSOL Java API or the optional LiveLink for MATLAB.

When solving multiphysics optimization problems in the COMSOL Desktop using the **Optimization** interface, some of the properties listed in this section can be controlled while others, the *advanced properties*, always take their default values. Modifying the value of an advanced property requires that the value is changed using LiveLink for MATLAB or by running a compiled COMSOL Java history file. A list of all available optimization solver properties also appears in the *COMSOL Java API Reference Guide* entry for [Optimization](#).



In the following sections,  $\epsilon$  represents the machine precision (available as `eps`) and is approximately equal to  $2.2 \cdot 10^{-16}$ .

In this section:

- [Feastol](#)
- [Funcprec](#)
- [Hessupd](#)
- [Majfeastol](#)
- [Opttol](#)
- [Qpsolver](#)

---

## *Feastol*

*Feasibility tolerance*

*Type: numeric*

*Default:  $1.0 \cdot 10^{-6}$*

The solver tries to ensure that all bound and linear constraints are eventually satisfied to within the feasibility tolerance  $t$ . (Feasibility with respect to nonlinear constraints is instead judged by the major feasibility tolerance, `majorfeasol`.)

If the bounds and linear constraints cannot be satisfied to within  $t$ , the problem is declared infeasible. Let `sInf` be the corresponding sum of infeasibilities. If `sInf` is quite small, it might be appropriate to raise  $t$  by a factor of 10 or 100. Otherwise you should suspect some error in the data.

Nonlinear functions are evaluated only at points that satisfy the bound and linear constraints. If there are regions where a function is undefined, every attempt should be made to eliminate these regions from the problem. For example, if

$$f(x) = \sqrt{x_1} + \log x_2$$

it is essential to place lower bounds on both variables. If  $t = 10^{-6}$ , the bounds

$$x_1 \geq 10^{-5} \text{ and } x_2 \geq 10^{-4}$$

might be appropriate. (The log singularity is more serious. In general, keep  $x$  as far away from singularities as possible.)

In practice, the solver uses  $t$  as a feasibility tolerance for satisfying the bound and linear constraints in each QP subproblem. If the sum of infeasibilities cannot be reduced to zero, the QP subproblem is declared infeasible. The solver is then in the Elastic mode thereafter (with only the linearized nonlinear constraints defined to be elastic).

## *Funcprec*

---

*Function precision*

*Type: numeric*

*Default:  $\epsilon^{0.8} \approx 3.8 \cdot 10^{-11}$*

The relative function precision is intended to be a measure of the relative accuracy with which the nonlinear functions can be computed. For example, if  $f(x)$  is computed as 1000.56789 for some relevant  $x$  and if the first 6 significant digits are known to be correct, the appropriate value for the function precision would be  $10^{-6}$ . (Ideally the functions should have a magnitude of order 1. If all functions are substantially less than 1 in magnitude, the function precision should be the absolute precision. For example, if  $f(x) = 1.23456789 \cdot 10^{-4}$  at some point and if the first 6 significant digits are known to be correct, the appropriate precision would be  $10^{-10}$ .)

The default value is appropriate for simple analytic functions.

In some cases the function values are the result of extensive computations, possibly involving an iterative procedure that can provide rather few digits of precision at reasonable cost. Specifying an appropriate function precision might lead to savings by allowing the line search procedure to terminate when the difference between function values along the search direction becomes as small as the absolute error in the values.

### *Hessupd*

---

#### *Hessian updates*

*Type: integer*

*Default: 10*

When the number of nonlinear variables is large (more than 75) or when the QP problem solver is set to conjugate-gradient, a limited-memory procedure stores a fixed number of BFGS update vectors and a diagonal Hessian approximation. In this case, if `hessupd` BFGS updates have already been carried out, all but the diagonal elements of the accumulated updates are discarded and the updating process starts again. Broadly speaking, the more updates stored, the better the quality of the approximate Hessian. However, the more vectors stored, the greater the cost of each QP iteration. The default value is likely to give a robust algorithm without significant expense, but faster convergence can sometimes be obtained with significantly fewer updates (for example, `hessupd = 5`).

### *Majfeastol*

---

#### *Major feasibility tolerance*

*Type: numeric*

*Default:  $1.0 \cdot 10^{-6}$*

This parameter specifies how accurately the nonlinear constraints should be satisfied. The default value of  $1.0 \cdot 10^{-6}$  is appropriate when the linear and nonlinear constraints contain data to roughly that accuracy.

Let `rowerr` be the maximum nonlinear constraint violation, normalized by the size of the solution. It is required to satisfy

$$\text{rowerr} = \max_i \text{viol}_i / (\|x\| + 1) \leq \text{majfeastol}$$



where  $\text{viol}_i$  is the violation of the  $i$ th nonlinear constraint. If some of the problem functions are known to be of low accuracy, a larger major feasibility tolerance might be appropriate.

## *Opttol*

---

*Optimality tolerance*

Type: *numeric*

Default:  $1.0 \cdot 10^{-6}$

This is the major optimality tolerance and specifies the final accuracy of the dual variables. On successful termination, the solver computes a solution  $(x, s, \pi)$  such that

$$\max_j \text{Comp}_j / \|\pi\| \leq \text{opttol}$$

where  $\text{Comp}_j$  is an estimate of the complementarity slackness for variable  $j$ . The values  $\text{Comp}_j$  are computed from the final QP solution using the reduced gradients  $d_j = g_j - \pi^T a_j$ , as above. Hence you have

$$\text{Comp}_j = \begin{cases} d_j \min \{x_j - l_j, 1\} & \text{if } d_j \geq 0 \\ -d_j \min \{u_j - x_j, 1\} & \text{if } d_j < 0 \end{cases}$$



Note

See the *SNOPT User's Guide* for further details.

---

## *Qpsolver*

---

*QP problem solver*

Type: *string* 'cholesky', 'cg', or 'qn'

Default: 'cholesky'

Specifies the active-set algorithm used to solve the QP problem, or in the nonlinear case, the QP subproblem.

'cholesky' holds the full Cholesky factor  $R$  of the reduced Hessian  $Z^T H Z$ . As the QP iterations proceed, the dimension of  $R$  changes with the number of superbasic variables.

'qn' solves the QP subproblem using a quasi-Newton method. In this case,  $R$  is the factor of a quasi-Newton approximate Hessian.

'cg' uses an active-set method similar to 'qn' but uses the conjugate-gradient method to solve all systems involving the reduced Hessian.

The Cholesky QP solver is the most robust but might require a significant amount of computation if the number of superbasics is large.

The quasi-Newton QP solver does not require the computation of the exact  $R$  at the start of each QP and might be appropriate when the number of superbasics is large but each QP subproblem requires relatively few minor iterations.

The conjugate-gradient QP solver is appropriate for problems with large numbers of degrees of freedom (many superbasic variables). The Hessian memory option 'hessmem' is defaulted to 'limited' when this solver is used.

### *References for the Optimization Solvers*

---

1. P.E. Gill, W. Murray, and M.A. Saunders, *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, Systems Optimization Laboratory (SOL), Stanford University, 2006.
2. P.E. Gill, W. Murray, and M.A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, vol. 47, no. 1, pp. 99–131, 2005.

## Glossary

This [Glossary of Terms](#) contains modeling terms in an optimization and sensitivity context. For general mathematical and finite element terms, and geometry and CAD terms specific to the COMSOL Multiphysics software and documentation see the glossary in the *COMSOL Multiphysics User's Guide*. For references to more information about a term, see the index.

# Glossary of Terms

**bounds** An inequality constraint setting lower and upper bounds directly on each control variable degree of freedom.

**contributions to objective function** The objective function is a scalar function of the control variables. In the Optimization interface, the objective is formed by the summation of contributions from *global contributions*, *probe contributions*, and *integral contributions* to the objective functions.

**control variable** The *control variables* parameterize the optimization or sensitivity problem. The objective function and constraint are expressed in the terms of the control variables. In the mathematical and engineering literature, the control variables are sometimes also referred to as *optimization variables*, *design variables*, or *decision variable*.

**design problem** An optimization problem where the objective function quantifies the performance in a multiphysics model. For such problems, the control variable is sometimes referred to as the *design variables*. Problems of this kind arise in, for example, structural optimization, antenna design, and process optimization.

**feasible set** The control variables may be constrained to a feasible set. The feasible set is typically expressed by a set of constraints acting on the control variables. The feasible set may also be implicitly limited by the existence of a solution to a multiphysics problem.

**global inequality constraint** A constraint that sets upper and lower bounds on a general global expression, possibly involving both the control variables and the PDE solution.

**integral inequality constraint** A constraint that sets upper and lower bounds on an integral of an expression, possibly involving the PDE solution and control variables, over a set of geometric entities of the same dimension

**objective function** A single-valued function of the PDE solution and control variables representing the performance of a multiphysics model or how well a parameterized model matches measured data. Alternative terminology used for the objective function is *cost function*, *goal function*, or *quantity of interest*.

**optimization problem** The *optimization problem* is to find values of the control variables, belonging to a given feasible set, such that the objective function attains its minimum (or maximum) value.

**parameter estimation problem** An inverse problem where the objective function defines how well a parameterized model matches measured data. Replacing the parameters with control variables leads to an optimization problem. Such problems arise in, for example, geophysical imaging, nondestructive testing, and biomedical imaging.

**PDE-constrained optimization problem** An optimization problem where the feasible set is limited by the condition that a given multiphysics model, represented as a PDE, has a unique solution.

**PDE solution** The solution to a multiphysics problem in response to specific values of the control variables.

**pointwise inequality constraint** An inequality constraint in a PDE-constrained optimization problem involving an explicit expression in terms of the control variables. The constraint sets lower and upper bounds on the expression for node points in a set of geometric entities of the same dimension.

**sensitivity problem** The sensitivity problem determines the gradient of an objective function with respect to the control variables.

**solution variables** Designates variables that are not control variables, for example, field variables and global variables.



# I n d e x

- A**
  - adjoint methods 25
  - adjoint sensitivity 23, 27
  - advanced properties 56
  - ALE method 29
- B**
  - BFGS 54
  - bounds 19
- C**
  - Cholesky solver 54
  - classical optimization 17
  - conjugate-gradient solver 54, 60
  - constraints
    - bounds 19
    - explicit 19
    - expression 37
    - implicit 19
    - integral inequality 20, 37
    - pointwise inequality 19, 38
  - contribution
    - global 18, 26, 62
    - integral 18, 26, 62
    - probe 18, 26, 62
  - control variable bounds 19
  - control variable field 44
  - control variable field (node) 39
  - control variables 16–17, 33, 40, 62
  - convergence 51
  - coordinate column (node) 36
  - cost function 62
- D**
  - decision variables 62
  - design problems 8
  - design variables 62
  - direct linear solver 51
  - documentation, finding 9
- E**
  - emailing COMSOL 10
  - equality constraints 38
  - evaluations, objective 52
  - explicit constraints 19
  - expression, objective 36, 39
- F**
  - feasibility tolerance 56
  - feasible set 16–17
  - forward methods 25
  - forward sensitivity 23–24, 27
  - function
    - cost 62
    - goal 62
  - function precision 57
  - functions
    - objective 8, 16–17
- G**
  - geometrical sensitivity 29
  - global
    - contribution 18, 26, 62
    - control variable (node) 40
    - inequality constraint (node) 40
    - objective (node) 39
  - global control variable (node) 45
  - global inequality constraint 20
  - global least-squares objective (node) 39
  - global objective (node) 45
  - global optimization 33
  - goal function 62
  - gradient-based optimization 19, 52–53
- H**
  - Hessian
    - reduced 54, 59
    - updates 58
- I**
  - ignored column (node) 36
  - implicit constraints 19
  - inequality constraints
    - global 40
    - integral 37
    - pointwise 38
  - integral

- contribution 18, 26
  - inequality constraints 20
  - objective (node) 36
- integral contribution 62
- integral inequality constraints (node) 37
- integral objective 43
- Internet resources 9
- inverse problems 9, 63
- iterative solver 51
- K** knowledge base, COMSOL 11
- L** least squares objective 39
  - least-squares objective (node) 34
  - Levenberg-Marquardt method 53
  - LiveLink for MATLAB 56
  - lower bound 38
- M** major
  - feasibility tolerance 57–58
  - optimality tolerance 50, 59
- mathematics branch 32
- model inputs
  - optimizing 8
- Model Library 10
- Model Wizard 32
- Moving Mesh interface 55
- MPH-files 10
- N** nonlinear solver 51
- O** objective
  - evaluations 52
  - expression 36, 39
  - functions 8, 16–17
  - global 39
  - integral 36
- objective function 21, 33, 42
  - complex-valued 29
  - specification of 26
- optimality tolerance 50, 59
- optimization
  - gradient-based 19, 52–53
  - problem formulations 16
  - problems 9
  - solver interface 50
- optimization interface 33
  - theory 16
- Optimization Module 8
- optimization problem 63
- optimization variable 62
- P** parameter column (node) 35
- parameter estimation 9
- PDE-constrained optimization problem 17
- pointwise inequality constraint (node) 38
- pointwise inequality constraints 19
- potential energy 29
- precision, function 57
- principle of virtual work 29
- probe contribution 18, 26, 62
- probe objective 18, 26, 43
- probe objective (node) 37
- Q** QP problem solver 59
- quadrature 36
- quantity of interest 62
- quasi-Newton solver 54, 60
- S** sensitivity analysis 21
  - discretization effect 28
- sensitivity interface 42
  - theory 21
- SNOPT 8, 50
- solution variable 19, 33
- solver
  - Cholesky 54
  - conjugate-gradient 54, 60
  - optimization 50
  - quasi-Newton 54, 60
- step condition 55



- superbasic variable 54, 59
- T** technical support, COMSOL 10
- theory
  - optimization interface 16
  - sensitivity interface 21
- time column (node) 35
- tolerance
  - feasibility 56, 58
  - optimality 50, 59
  - value 51
- typographical conventions 11
- U** upper bound 38
- user community, COMSOL 11
- V** value column (node) 35
- variables
  - control 16–17
  - decision 62
  - global 40
  - sensitivity solvers 24
  - solution 19
  - superbasic 54, 59
- W** warning messages 51
- web sites, COMSOL 11

