

Optimization Module

User's Guide

Optimization Module User's Guide

© 1998–2025 COMSOL

Protected by patents listed on www.comsol.com/patents, or see Help > About COMSOL Multiphysics on the File menu in the COMSOL Desktop for less detailed lists of U.S. Patents that may apply. Patents pending.

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

COMSOL, the COMSOL logo, COMSOL Multiphysics, COMSOL Desktop, COMSOL Compiler, COMSOL Server, and LiveLink are either registered trademarks or trademarks of COMSOL AB. All other trademarks are the property of their respective owners, and COMSOL AB and its subsidiaries and products are not affiliated with, endorsed by, sponsored by, or supported by those trademark owners. For a list of such trademark owners, see www.comsol.com/trademarks.

Version: COMSOL 6.4

Contact Information

Visit the Contact COMSOL page at www.comsol.com/contact to submit general inquiries or search for an address and phone number. You can also visit the Worldwide Sales Offices page at www.comsol.com/contact/offices for address and contact information.

If you need to contact Support, an online request form is located on the COMSOL Access page at www.comsol.com/support/case. Useful links:

- Support Center: www.comsol.com/support
- Product Download: www.comsol.com/product-download
- Product Updates: www.comsol.com/product-update
- COMSOL Blog: www.comsol.com/blogs
- Discussion Forum: www.comsol.com/forum
- Events: www.comsol.com/events
- COMSOL Video Gallery: www.comsol.com/videos
- Support Knowledge Base: www.comsol.com/support/knowledgebase
- Learning Center: www.comsol.com/support/learning-center

Part number: CM021701

C o n t e n t s

Chapter 1: Introduction

Optimization Module Overview	10
What Can the Optimization Module Do?	10
Where Do I Access the Documentation and Application Libraries?	12

Chapter 2: Optimization and Sensitivity Theory

Optimization Theory	16
Basic Optimization Concepts	16
Optimization Problem Formulation	16
PDE-Constrained Optimization	17
 Theory for the Sensitivity Interface	 22
About Sensitivity Analysis	22
Sensitivity Problem Formulation	22
Theory for Stationary Sensitivity Analysis.	23
Theory for Eigenvalue Sensitivity Analysis	26
Theory for Time-Dependent Sensitivity	27
Specification of the Objective Function	29
Choosing a Sensitivity Method.	30
Postprocessing Sensitivities	31
Issues to Consider Regarding the Control Variables	32
Issues to Consider Regarding the Objective Function	33
Issues to Consider Regarding Constraints	34
 Theory for Shape Optimization	 36
Transformation	36
PDE-Based Regularization	36
Polynomial-Based Regularization	37
Summary	38
References	38

Theory for Topology Optimization	39
The Helmholtz Filter	39
Milling	40
Projection	40
Penalization	41
Summary	42

Chapter 3: The Optimization Interface

Adding an Optimization Interface	44
 The Optimization Interface	 45
Least-Squares Objective	46
Value Column	48
Time Column	49
Parameter Column	49
Coordinate Column.	49
Ignored Column	49
Integral Objective (Point Sum Objective)	50
Probe Objective	50
Integral Inequality Constraint (Point Sum Inequality Constraint)	51
Pointwise Inequality Constraint	51
Control Variable Field	52
Control Variable Bounds	53
Global Objective	54
Global Least-Squares Objective	54
Global Inequality Constraint	55
Global Control Variables	56
 Topology Optimization	 58
Introduction	58
Density Model	58
Prescribed Void	62
Prescribed Material	63
Prescribed Density	63
Prescribed Void Boundary	63

Prescribed Material Boundary	63
Prescribed Density Boundary	64
Mirror Symmetry.	64
Sector Symmetry.	65
Shape Optimization	66
Introduction	66
Free Shape Domain	67
Transformation	68
Free Shape Boundary	69
Free Shape Shell	71
Polynomial Boundary	72
Polynomial Shell	73
Symmetry/Roller	74
Fixed Point	74
Fixed Edge	74
Fixed Boundary	75
Mirror Symmetry.	75
Sector Symmetry.	76
Parameter Estimation	77
Introduction	77
Least-Squares Objective	77
Global Least-Squares Objective	78
Control Variables	79
Introduction	79
Control Function.	79
Control Variable Field	80
References	82

Chapter 4: The Sensitivity Interface

The Sensitivity Interface	84
Integral Objective	85

Probe Objective	86
Control Variable Field	86
Global Objective	87
Global Control Variables	87

Chapter 5: The Optimization Solvers

The Parameter Optimization Study	91
The Parameter Estimation Study	98
The Topology Optimization Study	103
The Shape Optimization Study	109
The General Optimization Study	115
The Curve Fitting Study	129
About the Optimization Solvers	131
About Derivative-Free Solvers.	131
About Gradient-Based Solvers.	132
The Nelder–Mead Solver.	133
The BOBYQA Solver	134
The COBYLA Solver	134
The EGO Solver	134
The IPOPT Solver	135
The MMA Solver	137
The Levenberg–Marquardt Solver	138
About the Convergence Plot and Progress Window	139
About Optimality Tolerances	140
About Constraint Handling	143
References for the Optimization Solvers	146

The Optimization Solver	147
Cluster Computing	156
Distributing the Forward Problem	156
Parametric Sweep and Gradient Free Optimization	156
Load Cases, Multiple Frequencies, and Other Inner Sweeps.	157
Advanced Solver Properties	158
MMA Solver Properties	158

Chapter 6: Glossary

Glossary of Terms	160
Index	163

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 Reference Manual*. This chapter contains an overview of the Optimization Module.

Optimization Module Overview

What Can the Optimization Module Do?

The Optimization Module can be used throughout the COMSOL product family — it provides 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.

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*, where the problem is to find the values of control variables or design variables that yield the best performance of a model, quantified by means of an objective 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, where the problem is to reliably determine the values of a set of parameters that provide simulated data that best matches measured data. Such problems arise in, for example, geophysical imaging, nondestructive testing, and biomedical imaging. *Curve fitting* also belongs to this category.

Problems of the above types can often be formulated more generally as *optimization problems*. The Optimization interface and Optimization study step in COMSOL Multiphysics® are useful for solving design problems as well as inverse problems and parameter estimation.

OPTIMIZATION ALGORITHMS

There are four optimization algorithms for gradient-based optimization available in the Optimization Module.

The first algorithm is the IPOPT solver. When using IPOPT, 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, frequency dependent, or time dependent, analytic

sensitivities of the objective function with respect to the control variables can be used. The main authors are Andreas Waechter and Carl Laird from IBM, but the code is still developed as an open-source project with many contributors.

The second and third algorithms are the MMA and GCMMA solvers, which are based on the method of moving asymptotes by Krister Svanberg of KTH (Royal Institute of Technology, Stockholm). The MMA solver can handle objective functions and constraints of the same very general form as IPOPT. It can solve min-max optimization problems, and it is well suited to handle problems with a large number of control variables, such as topology optimization. The globally convergent version of the solver is often referred to as GCMMA. Both MMA and GCMMA have support for limiting the step length via move limits.

The fourth 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 designed to solve problems of least-squares type, it typically converges faster than IPOPT for such problems.

In addition, the Optimization Module provides a number of gradient-free (derivative-free) optimization algorithms. Currently Nelder–Mead, BOBYQA, COBYLA, EGO, and a coordinate search are supported. These methods can optimize a model with respect to design parameters (model parameters) such as parameters which control the geometry sequence that defines the model’s geometry.

All optimization solvers are accessible from the same General Optimization study step, which contains the ordinary solver sequence over which the optimization method iterates. The gradient-free methods can contain any other study sequence, while the gradient-based methods are limited to optimizing over a single study step of a type supporting computation of analytic sensitivities: currently Stationary, Time Dependent, Frequency Domain, and Eigenfrequency studies.



[The Physics Interfaces](#) and [Building a COMSOL Multiphysics Model](#) in
the *COMSOL Multiphysics Reference Manual*

Where Do I Access the Documentation and Application Libraries?



Optimization model examples that do not require any other COMSOL add-on modules are located in the Optimization Module folder in the Application Libraries window. There are also other optimization models found in other folders for different modules. You can find all related examples by entering **optimization** in the search field at the top of the Application Libraries window.

A number of online resources have more information about COMSOL, including licensing and technical information. The electronic documentation, topic-based (or context-based) help, and the Application Libraries 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 an application or content referenced in a different guide. However, if you are using the Help system in COMSOL Multiphysics, these links work to open other modules, application examples, and documentation sets.

THE DOCUMENTATION AND ONLINE HELP

The *COMSOL Multiphysics Reference Manual* describes the core physics interfaces and functionality included with the COMSOL Multiphysics license. This book also has instructions on how to use COMSOL Multiphysics and how to access the electronic Documentation and Help content.

Opening Topic-Based Help

The Help window is useful as it is connected to the features in the COMSOL Desktop. To learn more about a node in the Model Builder, or a window on the Desktop, click to highlight a node or window, then press F1 to open the Help window, which then

displays information about that feature (or click a node in the Model Builder followed by the **Help** button (). This is called *topic-based* (or *context*) *help*.

Win	To open the Help window: • In the Model Builder, Application Builder, or Physics Builder, click a node or window and then press F1. • On any toolbar (for example, Home, Definitions, or Geometry), hover the mouse over a button (for example, Add Physics or Build All) and then press F1. • From the File menu, click Help (). • In the upper-right corner of the COMSOL Desktop, click the Help () button.
Mac	To open the Help window: • In the Model Builder or Physics Builder, click a node or window and then press F1. • In the main toolbar, click the Help () button. • From the main menu, select Help > Help.
Linux	

Opening the Documentation Window

Mac	To open the Documentation window: • Press Ctrl+F1. • In the main toolbar, click the Documentation () button. • From the main menu, select Help > Documentation.
Linux	

THE APPLICATION LIBRARIES WINDOW

Each model or application includes documentation with the theoretical background and step-by-step instructions to create a model or application. The models and applications are available in COMSOL Multiphysics as MPH files that you can open for further investigation. You can use the step-by-step instructions and the actual models as templates for your own modeling. In most models, SI units are used to describe the relevant properties, parameters, and dimensions, but other unit systems are available.

Once the Application Libraries window is opened, you can search by name or browse under a module folder name. Click to view a summary of the model or application and its properties, including options to open it or its associated PDF document.

Opening the Application Libraries Window

To open the **Application Libraries** window ():

CONTACTING COMSOL BY EMAIL

For general product information, contact COMSOL at info@comsol.com.

COMSOL ACCESS AND TECHNICAL SUPPORT

To receive technical support from COMSOL for the COMSOL products, please contact your local COMSOL representative or send your questions to support@comsol.com. An automatic notification and a case number will be sent to you by email. You can also access technical support, software updates, license information, and other resources by registering for a COMSOL Access account.

COMSOL ONLINE RESOURCES

COMSOL website	www.comsol.com
Contact COMSOL	www.comsol.com/contact
COMSOL Access	www.comsol.com/access
Support Center	www.comsol.com/support
Product Download	www.comsol.com/product-download
Product Updates	www.comsol.com/product-update
COMSOL Blog	www.comsol.com/blogs
Discussion Forum	www.comsol.com/forum
Events	www.comsol.com/events
COMSOL Application Gallery	www.comsol.com/models
COMSOL Video Gallery	www.comsol.com/videos
Learning Center	www.comsol.com/support/learning-center
Support Knowledge Base	www.comsol.com/support/knowledgebase

Optimization and Sensitivity Theory

This chapter discusses the theory for optimization and sensitivity. In this chapter:

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

Optimization Theory

This section contains theory useful for understanding and applying [The Optimization Interface](#) and [The General Optimization Study](#). Topics explained in this section are:

- [Basic Optimization Concepts](#)
- [Optimization Problem Formulation](#)
- [PDE-Constrained Optimization](#)

Basic Optimization Concepts

In general, there are three fundamental parts of an optimization problem — the *control variables*, the *objective function* and, optionally, *constraints*.

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 General Optimization Study](#) together with [The Optimization Interface](#) provide a framework for specifying and solving general optimization problems. The objective function and constraints can depend indirectly on the control variables via the solution of a multiphysics model. See [PDE-Constrained Optimization](#).

Optimization Problem Formulation

The Optimization Module is built around a general single-objective minimization problem formulation. The Optimization Study node transforms maximization as well as multiobjective minimax and maximin problems internally to the canonical minimization form.

THE GENERAL OPTIMIZATION PROBLEM

The most 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 ξ , the scalar-valued objective function by Q , and the feasible set is denoted by C . Assuming sufficient continuity, the feasible set can be expressed as a set of — possibly very nonlinear — 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).



For vectorial quantities, the inequality defining C is to be interpreted component-wise, and lb and 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 ξ . However, design problems and parameter estimation problems often result in objective functions Q and constraints G that are not explicitly expressible as closed-form expressions of the control variables ξ .

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 the 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 ξ the control variables.

The complete PDE-constrained optimization problem to be solved by one of the optimization algorithms in the Optimization Module 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 separate those constraints in G that are defined as explicit expressions of ξ only (*design constraints*) from those that mix u and ξ (*performance constraints*). The former group can further be divided into *simple bounds*, which set

a lower and upper limit directly on the control variables, and constraints on general expressions of the control variables. Hence, the general constraint formulation $\text{lb} \leq G(u(\xi), \xi) \leq \text{ub}$ above is replaced by three classes of constraints:

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

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

$$\left\{ \begin{array}{l} \min_{\xi} \quad Q(u(\xi), \xi) \\ L(u(\xi), \xi) = 0 \\ \text{lb}_P \leq P(u(\xi), \xi) \leq \text{ub}_P \\ \text{lb}_\Psi \leq \Psi(\xi) \leq \text{ub}_\Psi \\ \text{lb}_b \leq \xi \leq \text{ub}_b \end{array} \right. \quad (2-3)$$

This is the general form of the optimization problem considered in the Optimization Module. Control variables ξ can be either global model parameters converted into control variables in the Optimization study step or control variable fields set up in an Optimization interface. You specify the objective function and the constraints in the form of expressions of ξ and u . The relation between u and ξ , which is a system of equations written here compactly as $L(u, \xi) = 0$, is given by the multiphysics model.

SPECIFICATION OF THE OBJECTIVE FUNCTION

The objective function is, in general, 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_{global} is the *global contribution* to the objective function Q . It is given as one or more general global expressions, either in an Optimization study step or in a Global Objective node under an Optimization interface.
- Q_{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. You specify the

probe point used for the point evaluation explicitly in a Probe Objective node under an Optimization interface.

- $Q_{\text{int},k}$ 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. Use an Integral Objective node under an Optimization interface to specify an integrand and a select a set of domains, boundaries, edges, or points over which to integrate. For a point selection, the integration reduces to a summation.

Several global, probe, and integral contributions can be defined in separate nodes under an Optimization interface. In such cases, the total global, probe, and integral contribution is given as the sum of the contributions. If you specify one or more objectives directly in the Optimization study step, these are also added to the sum.

SPECIFICATION OF CONSTRAINTS

The full nonlinear set of constraints $\text{lb} \leq G(u(\xi), \xi) \leq \text{ub}$ in the general PDE-constrained optimization problem, [Equation 2-2](#), are separated into three groups:

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

The first row above contains the general *implicit constraints*, or *performance constraints* in the case of a design problem. These are given in terms of expressions involving both the solution variables u and control variables ξ . The second row constitutes the *explicit constraints* — or *design constraints* — which are those

constraints given by explicit expressions only in the control variables ξ . The last row contains the *control variable bounds*.



The reason for this subdivision is computational. Each evaluation of an implicit constraint requires an up-to-date solution of the multiphysics solution u . The gradient-based optimization methods also require a complete sensitivity evaluation, which is computationally demanding; see [Choosing a Sensitivity Method](#) in the *COMSOL Multiphysics Reference Manual*.

Explicit constraints, in contrast, can be computed without updating the multiphysics solution. They can, however, be nonlinear, making it difficult for the optimization methods to follow an active constraint. Control variable bounds are the least expensive to handle, since when active, the optimization solver can essentially just exclude the corresponding control variable.

The **Optimization** interface differentiates between the following constraints (in the description that follows, n denotes the dimension of the multiphysics model): control variable bounds, pointwise inequality constraints, integral inequality constraints, and global 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 $lb \leq \xi \leq ub$. They are handled efficiently by all solvers that support them and in many cases improve solver stability and efficiency.
- *Pointwise inequality constraints* are inequality constraints involving an explicit expression in terms of the control variables. This type of 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. Apart from the specification of bounds, a global inequality constraint is identical to a Global Objective.
- *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.



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

Theory for the Sensitivity Interface

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 for evaluating the sensitivity of a model with respect to almost any variable. The Sensitivity interface is used together with a Sensitivity study step, which in turn controls the Sensitivity solver extension. Simple cases can be handled directly in the Sensitivity study step, while more advanced cases must be set up in a Sensitivity interface prior to solving.

Simulation is a powerful tool for predicting the behavior of physical systems, particularly those that are governed by partial differential equations. However, a single simulation is often 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 study step and corresponding physics 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 physics 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 set of control variables can be specified. The Sensitivity interface can perform these distinct tasks:

- Select control variables and set their values
- Define scalar objective functions



The control variables are independent variables whose values are 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.

The companion Sensitivity study step is responsible for:

- Choosing which objective functions and control variables to solve for
- Selecting a sensitivity evaluation method
- Selecting which study step to compute sensitivities for
- Setting up the Sensitivity solver extension

Theory for Stationary Sensitivity Analysis

Evaluating the sensitivity of a scalar-valued objective function $Q(\xi)$ with respect to the control variables, ξ , at a specific point, ξ_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 ξ alone. Rather, $Q(u(\xi), \xi)$ is also a function of the solution variables u , which are in turn implicitly functions of ξ . Therefore, by using the chain rule, the sensitivity can be computed as

$$\frac{d}{d\xi} Q(u(\xi), \xi) = \frac{\partial Q}{\partial u} \frac{\partial u}{\partial \xi} + \frac{\partial Q}{\partial \xi} \quad (2-4)$$

The explicit partial derivative of the objective function with respect to the control variables or to the solution u are easy to compute using symbolic differentiation. In contrast, the remaining term, the partial derivative of the solution u with respect to the control variables, is more difficult to obtain. Since the multiphysics problem is a PDE, which after discretization is represented by a system of equations $L(u(\xi), \xi) = 0$, computing the total derivative of this expression with respect to ξ gives

$$0 = \frac{dL}{d\xi} = \frac{\partial L}{\partial u} \frac{\partial u}{\partial \xi} + \frac{\partial L}{\partial \xi}$$

Assuming that $L(u(\xi), \xi) = 0$ has a unique solution, the matrix $\partial L/\partial u$ is nonsingular, which implies

$$\frac{\partial u}{\partial \xi} = -\left[\frac{\partial L}{\partial u}\right]^{-1} \frac{\partial L}{\partial \xi} \quad (2-5)$$

By inserting Equation 2-5 into Equation 2-4, the sensitivity can be rewritten as

$$\frac{d}{d\xi} Q(u(\xi), \xi) = \frac{\partial Q}{\partial u} \frac{\partial u}{\partial \xi} + \frac{\partial Q}{\partial \xi} = -\frac{\partial Q}{\partial u} \left[\frac{\partial L}{\partial u}\right]^{-1} \frac{\partial L}{\partial \xi} + \frac{\partial Q}{\partial \xi} \quad (2-6)$$

Assuming that the PDE solution has N degrees of freedom and that there are n control variables ξ_i , $\partial Q/\partial u$ is an 1-by- N matrix, $\partial L/\partial u$ is an N -by- N matrix, and $\partial L/\partial \xi$ is an N -by- n matrix.



The system of equations, L , is here assumed to include any constraints present in the multiphysics model. The number of degrees of freedom, N , therefore in theory includes also Lagrange multipliers for the constraints. In practice, constraints are usually eliminated, which imposes some restrictions on the sensitivity analysis; see [The Sensitivity Analysis Algorithm](#) in the *COMSOL Multiphysics Reference Manual*.

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 (see [Forward Sensitivity Methods](#) and [Adjoint Sensitivity Method](#) below).

If an incomplete Jacobian has been detected during the sensitivity analysis, an attempt to assemble the complete Jacobian is done. If the assembly succeeds, the complete Jacobian is used in sensitivity computations in the following way:

Assume that the Jacobian $\partial L/\partial u$ in Equation 2-4 above is incomplete and denote it by $(\partial L/\partial u)_{\text{incomplete}}$.

Let the complete Jacobian be $\partial L/\partial u$. Hence, the system to solve is

$$\frac{\partial L}{\partial u} \cdot \frac{\partial u}{\partial \xi_i} = -\frac{\partial L}{\partial \xi_i} \quad (2-7)$$

Using

$$\frac{\partial L}{\partial u} = \left(\frac{\partial L}{\partial u} \right)_{\text{incomplete}} + \left(\frac{\partial L}{\partial u} - \left(\frac{\partial L}{\partial u} \right)_{\text{incomplete}} \right)$$

the previous system becomes

$$\left(\frac{\partial L}{\partial u} \right)_{\text{incomplete}} \cdot \frac{\partial u}{\partial \xi_i} = -\frac{\partial L}{\partial \xi_i} - \left(\left(\frac{\partial L}{\partial u} \right)_{\text{incomplete}} - \frac{\partial L}{\partial u} \right) \cdot \frac{\partial u}{\partial \xi_i}$$

Then, the solution to the system in Equation 2-7 is approximated iteratively by

$$\left(\frac{\partial L}{\partial u} \right)_{\text{incomplete}} \cdot \left(\frac{\partial u}{\partial \xi_i} \right)^n = -\frac{\partial L}{\partial \xi_i} - \left(\left(\frac{\partial L}{\partial u} \right)_{\text{incomplete}} - \frac{\partial L}{\partial u} \right) \cdot \left(\frac{\partial u}{\partial \xi_i} \right)^{n-1}$$

where n is the iteration number.

The iterations are terminated either when the estimated error is less than the relative tolerance used by the current solver (convergence), or when the number of iterations has reached the maximum number of iterations specified in the Fully Coupled or Segregated attribute node (nonconvergence).

If the previous algorithm does not converge (that is, the estimated error is larger than the given tolerance), the sensitivity computations are repeated using the incomplete Jacobian and the warning *Jacobian is incomplete. No convergence when attempting to use the complete Jacobian* is written.

If the assemble of the complete Jacobian fails, the incomplete Jacobian is used and the warning *Unable to assemble the complete Jacobian. Using incomplete Jacobian for sensitivity analysis* is written.

In the Optimization interface, a warning is written only if the optimization problem does not converge.

FORWARD SENSITIVITY METHODS

The *forward sensitivity* method explicitly computes and returns the term $\partial u / \partial \xi$ in Equation 2-5 in order to compute the sensitivity in Equation 2-4 by solving n linear systems of equations

$$\frac{\partial u}{\partial \xi_i} = - \left[\frac{\partial L}{\partial u} \right]^{-1} \cdot \frac{\partial L}{\partial \xi_i}, \quad i = 1, \dots, n$$

When this system is solved, a line is added to the progress window displaying the number (*i*) of the current sensitivity parameter being solved for.

ADJOINT SENSITIVITY METHOD

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

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

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

$$\frac{\partial L}{\partial u}^T u^* = \left[\frac{\partial Q}{\partial u} \right]^T$$

using the transpose of the original PDE Jacobian.

Theory for Eigenvalue Sensitivity Analysis

In simplified terms, the eigenvalue problem solved by COMSOL Multiphysics can be written

$$0 = (E\lambda^2 - D\lambda + K)u \quad (2-8)$$

where λ is the eigenvalue and u is the eigenmode, while E , D , and K are the eliminated mass, damping, and stiffness matrices, respectively. These matrices are also sometimes referred to as E_c , D_c , and K_c . Taking the derivative of Equation 2-8 with respect to a set of controls, ξ , yields

$$0 = (E\lambda^2 - D\lambda + K) \frac{\partial u}{\partial \xi} + (2E\lambda - D)u \frac{\partial \lambda}{\partial \xi} + \frac{\partial J}{\partial \xi} u \quad (2-9)$$

where $\frac{\partial J}{\partial \xi} = \frac{\partial E}{\partial \xi} \lambda^2 - \frac{\partial D}{\partial \xi} \lambda + \frac{\partial K}{\partial \xi}$

Multiplying this equation with the adjoint mode, u^* , and rearranging yields

$$u^*(2E\lambda - D)u \frac{\partial \lambda}{\partial \xi} = u^*(E\lambda^2 - D\lambda + K) \frac{\partial u}{\partial \xi} + u^* \frac{\partial J}{\partial \xi} u \quad (2-10)$$

so the adjoint equation can be identified as

$$0 = u^*(E\lambda^2 - D\lambda + K) \quad (2-11)$$

If the problem is symmetric, it is self-adjoint; that is, $u^* = u$. Otherwise, the left eigenvectors have to be computed. In either case, the sensitivity of the eigenvalue can be computed as

$$\frac{\partial \lambda}{\partial \xi} = \left(u^* \frac{\partial J}{\partial \xi} u \right) / (u^* (2E\lambda - D) u) \quad (2-12)$$

In COMSOL Multiphysics this is referred to as the *adjoint gradient method*. The implementation assumes that the controls do not enter any (nonweak) constraints. Moreover, the eigenvalue problem can be a linearization of a nonlinear problem, and the implementation assumes that the linearization point is independent of the controls.

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

$$Q = Q(u(\xi), \xi) \quad (2-13)$$

with respect to the control variables ξ evaluated at the final time $t=T$. The dependent variables, $u(\xi)$, depend on the controls, ξ , but for notational simplicity we will just write u in the following. Note that the objective cannot depend on time derivatives of the dependent variables, that is

$$\frac{\partial Q}{\partial \dot{u}} = 0 \quad (2-14)$$

This limitation can be circumvented by projecting the time derivative to a dependent variable. The forward solution u is a solution to the parameterized discrete forward problem

$$L(\dot{u}, u, \xi) = N_F \Lambda \quad M(u, \xi) = 0 \quad (2-15)$$

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

To compute the sensitivity of Q with respect to ξ , first apply the chain rule:

$$\frac{dQ}{d\xi} = \frac{\partial Q}{\partial \xi} + \frac{\partial Q}{\partial u} \frac{\partial u}{\partial \xi} \quad (2-16)$$

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 2-15](#), formally with respect to ξ :

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

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 ξ (that is, $\partial N_F/\partial \xi = 0$), you can write the above relations in matrix form as

$$\begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial \dot{u}}{\partial \xi} \\ \frac{\partial \Lambda}{\partial \xi} \end{bmatrix} + J \begin{bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial \Lambda}{\partial \xi} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \xi} \\ \frac{\partial M}{\partial \xi} \end{bmatrix} \quad J = \begin{bmatrix} K & N_F \\ N & 0 \end{bmatrix} \quad (2-17)$$

solve for the sensitivities $\partial u/\partial \xi$ and $\partial \Lambda/\partial \xi$, with initial conditions $\partial u_0/\partial \xi$ and $\partial \Lambda_0/\partial \xi$, respectively, and plug them back for evaluation at $t = T$ into [Equation 2-16](#).

If the number of individual control variables, ξ_j , is small, [Equation 2-17](#) 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 2-16](#). This is the *forward method*, which in addition to the sensitivity $dQ/d\xi$ returns the sensitivity of the solution, $\partial u/\partial \xi$. As an alternative, the right-hand side of [Equation 2-17](#) can be calculated by finite differences using the *forward numeric method*.

If there are many control variables and the sensitivity of the solution itself, $\partial u/\partial \xi$, 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 2-16](#):

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

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 require “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 can 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_{global} is the *global contribution* to the objective function Q . It is given as one or more general global expressions.
- Q_{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 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, using one of these methods:

- Select one of the [Forward Sensitivity](#) methods 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 (or forward numeric) 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 forward numeric method uses numerical perturbation rather than analytical methods

to calculate forward sensitivities, and can be used when the analytical method fails for some reason or as a tool to verify that the analytical method is correct. In addition, the forward numeric method requires two additional residual evaluations. The iterative linear and segregated solvers can reuse preconditioners and other data but must otherwise perform a complete solution each time. Further, the forward numeric method only differentiates the PDE problem numerically (giving a numeric method for the forward sensitivity). The objective sensitivity is still differentiated analytically (both with respect to the controls and with respect to the PDE variables). The functional sensitivity is therefore computed with a hybrid method.

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 [Studies and Solvers](#) in the *COMSOL Multiphysics Reference Manual*.

Postprocessing Sensitivities

When a multiphysics problem is solved using sensitivity analysis, the generated solution contains stored sensitivity data. You can access this data in postprocessing using the **fsens** and **sens** operators:

- **fsens**(<control_variable>) evaluates the sensitivity (derivative) of the objective function with respect to the specified control variable. This result is available for all sensitivity methods. The result of **fsens** can be evaluated on the geometric entities

where the control variable is defined. For a global control variable, `fsens` is available everywhere. In the same way, `fsensimag(<control_variable>)` evaluates the sensitivity (derivative) of the objective function with respect to the imaginary part of the specified control variable.

- `sens(<dependent_variable>, <control_variable>)` or `sens(<dependent_variable>, <control_DOF>)` evaluates the sensitivity (derivative) of the specified dependent variable with respect to the specified control variable degree of freedom. This is only possible when forward sensitivity has been used, which computes and stores derivatives of the entire solution vector with respect to each control variable degree of freedom.

Global control variables can be identified by name. Otherwise, control variable degrees of freedom are identified by their index (starting from 1) among all control variables in the solution vector. The result of `sens` has the same geometric scope as the dependent variable argument; it can be plotted or evaluated wherever the dependent variable itself is available.

Issues to Consider Regarding the Control Variables

THE EFFECT OF DISCRETIZATION

The sensitivity analysis is always performed on the discretized system of equations. As already mentioned, the control variables 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 a 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 can, 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 `dvo1` results in a plot that is proportional to the true pointwise sensitivity.



If the plan is to use the sensitivities in an automatic optimization procedure, as is done through the Optimization interface available with the 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

You can use the control variables directly to parameterize any aspect of the physics that is controlled by an expression. This applies 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 require special attention.

Control variables cannot be used directly in the geometry description. Instead, the model must be set up using a Deformed Geometry or Moving Mesh interface to control the shape of the geometry. Then use control variables to control the mesh movement, effectively parameterizing the geometry.



See [Deformed Geometry and Moving Mesh](#) in the *COMSOL Multiphysics Reference Manual* for details about these interfaces and ALE 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 directly applied only when the objective function is a real differentiable or complex analytic function of the control variables. This is usually not a severe constraint, even for frequency-domain models where the PDE solution

variables are complex valued. One 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.

Some PDE problem or the objective functions are, however, nonanalytic. This is the case, for example, when the equations or the objective function contain `real()`, `imag()`, or `abs()`. One solution in such cases is to enable **Split complex variables in real and imaginary parts** in the **Compile Equations** node corresponding to the study step for which sensitivity is computed. This converts the discretized PDE system from a complex-valued system to a real-valued system of double size, with separate degrees of freedom for the real and imaginary part. For this split system, also the nonanalytic functions are differentiable almost everywhere such that sensitivities can be computed.

One special form of nonanalytic objective functions can be treated more efficiently than splitting the variables: many common quantities of interest are harmonic time averages, which 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 symbolic 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

$$Q(a + \delta a, b + \delta b) \approx Q(a, b) + \text{real}\left(\frac{\partial Q}{\partial a} \delta a + \frac{\partial Q}{\partial b} \delta b\right) \quad (2-18)$$

for any small complex increments δa and δb , the final sensitivities are evaluated correctly. The special function `real_dot(a,b)` is identical to `real(a*conj(b))` when evaluated but implements partial derivatives according to Equation 2-18. For that reason, use it in the definition of any time-average quantity set as objective function in a sensitivity analysis.

Issues to Consider Regarding Constraints

The theory behind sensitivity analysis as presented above (under [Theory for Stationary Sensitivity Analysis](#)) assumes that constraints on the multiphysics problem are handled in the same way as with any other equations. This is indeed the case for *weak constraints*, which are implemented as a part of the main system of equations. Standard pointwise constraints are instead eliminated from the discretized equations at

an early stage in the solution process. This elimination is not visible to the sensitivity solver, which therefore may miss some symbolic derivative terms necessary for computing a correct sensitivity.

In particular, if the mixed second derivative of a standard constraint with respect to both PDE solution and control variables is nonzero, sensitivity will not be correctly computed. For example, for a solution variable u and a control variable p , a constraint:

- $u = p$ will give correct sensitivity.
- $u^2 = p^2$ will give correct sensitivity.
- $u^2 = up$ will give incorrect sensitivity.

If your multiphysics model contains constraints of the problematic type, you can still compute a correct sensitivity, provided that you enable *weak constraints* in the

Constraint Settings section of the corresponding boundary condition node.



For technical details about the solver implementation, see [The Sensitivity Analysis Algorithm](#) in the *COMSOL Multiphysics Reference Manual*.

For more about the standard versus the weak constraints, see [Boundary Conditions](#) in the *COMSOL Multiphysics Reference Manual*.

Theory for Shape Optimization

About Shape Optimization

Shape optimization involves techniques that use mesh deformation to change the geometry. Two kinds of optimization are available:

- Transformation (translation, scaling, and rotation)
- Boundary deformation
 - PDE-based regularization
 - Polynomial-based regularization

Shape optimization of shells is typically not associated with any domains, but in all other situations a PDE-based approach is used to smoothen the deformation in the interior of domains.

Transformation

The most common type of shape optimization involves simple transformation of domains or boundaries:

- Translation
- Scaling
- Rotation

This can also be achieved by changing the geometry directly, but that approach excludes the use of gradient-based optimization and in some cases one might be interested in the optimal translation and/or scaling of many objects, so that gradient based optimization is necessary.

PDE-Based Regularization

PDE-based regularization of shape optimization is inspired by the Helmholtz filter introduced for topology optimization; see [Ref. 1](#) and [Ref. 2](#). The shape deformation of the boundary is thus typically given by

$$\mathbf{d} = R_{\min}^2 \nabla_t^2 \mathbf{d} + \mathbf{c} \ , \quad \|\mathbf{c}\| \leq d_{\max}$$

where the maximum displacement, $d_{\max}$, and filter length, $R_{\min}$, effectively define a maximum slope of deformation. Boundary conditions can be added to fix the boundary deformation or allow sliding along planar boundaries. If the initial normal vector is orthogonal to the boundary, this property can be preserved by imposing $\mathbf{d} \cdot \mathbf{n} = 0$ on the edge. If the orthogonality should not be preserved, one can solve a Helmholtz filter on the edge and impose the result as a Dirichlet boundary condition.

If different values for $d_{\max}$ are present for boundaries that share an edge, the lower value of $d_{\max}$ will be used. The same logic is used when one boundary is regularized with a PDE while another is regularized with a polynomial, except in 3D where the polynomial boundary will always control the edge.

Polynomial-Based Regularization

All edges are parameterized and one can visualize the parameterization using the $\mathbf{s}$ and $\mathbf{s1}$ variables (2D and 3D). In general the edge parameter will go from 0 to 1, and if it does not it can be normalized to do so. Nothing similar is immediately possible for surfaces in 3D, because although a parameterization exists ($\mathbf{s1}$ and $\mathbf{s2}$), it is a trimmed parameterization. The 3D polynomial regularization thus relies on computing an untrimmed parameterization using a PDE, and this is only possible for triangular and quadrilateral boundaries.

Once the parameterization is available, the boundary deformation can be expressed as an explicit expression of a polynomial, the coefficients of which are the control parameters:

$$\mathbf{d} = \sum_{i=0}^n f_i^n(s) \mathbf{c}_i \quad , \quad \|\mathbf{c}_i\| \leq d_{\max}$$

$$f_i^n(s) = \frac{n!}{n!(n-i)!} s^{n-i} (1-s)^i$$

Here n is the order and $d_{\max}$ is the maximum displacement. The equation shown here is a Bernstein polynomial for an edge in 2D. In contrast to Lagrange polynomials, Bernstein polynomials respect the maximum displacement constraint strictly. Similar to the PDE-based regularization, terms can be excluded to prevent deformation of the boundary near fixed boundaries, and continuity of the normal near planar symmetry boundaries can also be controlled by restricting the design freedom, at least in 2D. Finally, it is possible to preserve continuity of the normal between boundaries in 2D,

but this is not possible in 3D and therefore the PDE-based regularization is preferred for most cases of 3D shape optimization.

Summary

The [Free Shape Domain](#) feature adds the variables listed in [Table 2-2](#).

TABLE 2-1: VARIABLES ASSOCIATED WITH THE FREE SHAPE DOMAIN FEATURE.

DESCRIPTION	NAME	AVAILABLE IN
Area of deformed domain	fsd#.area	2D/2Daxi
Relative area of deformed domain	fsd#.relArea	2D/2Daxi
Volume of deformed domain	fsd#.volume	3D
Relative volume of deformed domain	fsd#.relVolume	3D

References

1. K.U. Bletzinger, “A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape,” *Structural and Multidisciplinary*, vol. 49, no. 6, pp. 873–895, 2014.
2. B.S. Lazarov and O. Sigmund, “Filters in topology optimization based on Helmholtz-type differential equations,” *International Journal for Numerical Methods in Engineering*, vol. 86, pp. 765–781, 2011.

Theory for Topology Optimization

About Density Topology

Topology optimization with the density method is one of the oldest and most simple techniques. A domain control variable is discretized on nodes or elements and a fictitious material is introduced to account for the material boundary in an implicit way. An interpolation is then constructed such that the physical governing equation is solved wherever the control variable is equal to one, while an equation associated with the fictitious material is solved where the control variable is equal to zero. The interpolation is specific to the physics and it is constructed such that intermediate value of the control variable are suboptimal, see [Ref. 1](#). Furthermore, the problem might be ill-posed in the absence of a constraint on the design freedom. This constraint is often imposed implicitly by means of a filter that introduces a minimum length scale via its *filter radius*.

The Helmholtz Filter

One can impose a minimum length scale, $R_{\min}$, on a domain control variable, θ_c , using a Helmholtz filter (see [Ref. 2](#)):

$$\theta_f = R_{\min}^2 \nabla^2 \theta_f + \theta_c$$

where θ_f is the filtered material volume factor. The mesh element size is taken as the default filter radius, but the filter radius should not be smaller than mesh element size and one has to pick a fixed value to get mesh independent designs. The filtered material volume factor is continuous and discretized with linear polynomials. The filter can be turned off, in which case $\theta_f = \theta_c$ holds. [Figure 2-1](#) illustrates the filtering process, and

it is taken from the model *Topology Optimization of an MBB Beam* in the Optimization Module Application Library.



Figure 2-1: A symmetric topology optimization problem is shown with the control variable to the left and the filtered variable to the right.

Milling

It is possible to restrict the design freedom using milling constraints. This involves solving convective equation for each of the milling directions:

$$\theta_f/R_{\text{mil}} = \hat{\mathbf{m}}_{\text{mil}}^i \cdot \nabla \theta_m^i$$

The source radius, R_{mil} , is typically taken to be similar to the filtering radius. The automatic default option uses the filter radius, $R_{\text{mil}} = R_{\text{min}}$. Once all the milling fields, θ_m^i , have been computed, they are combined using an approximate maximum operator:

$$\theta_m = \left[\sum_{i=1}^n ((\theta_m^i)^{-p_{\text{mil}}})/n \right]^{-1/p_{\text{mil}}}$$

A large aggregation exponent, p_{mil} , gives a good approximation of a maximum operator. However, this can also cause numerical problems so the default automatic option uses $p_{\text{mil}} = 10$. The final milling material volume factor, θ_m , is often significantly larger than 1 so it is important to use projection with milling constraints.

Projection

The filtered (and milling) material volume factor can have large areas with intermediate values. This tends to make the optimization problem easier to solve, but the design might rely on areas with unphysical properties due to intermediate values. These areas

can then be reduced using a projection operation based on the hyperbolic tangent function (see [Ref. 3](#)):

$$\theta = \frac{\tanh(\beta(\theta_f - \theta_\beta)) + \tanh(\beta\theta_\beta)}{\tanh(\beta(1 - \theta_\beta)) + \tanh(\beta\theta_\beta)}$$

Here θ_β and β are the projection point and slope, respectively, and θ is the output material volume factor. This is plotted in [Figure 2-2](#) for $\theta_\beta = 1/2$ and $\beta = 8$. Projection can slow down the optimization progress. Therefore, it is turned off by default so that $\theta = \theta_f$.

Volume constraints are common in topology optimization and they are simple to implement using the average material volume factor, θ_{avg} :

$$\theta_{\text{avg}} = \int_{\Omega} \theta d\Omega / \int_{\Omega} d\Omega$$



Figure 2-2: A symmetric topology optimization problem is shown with the filtered variable to the left and the projected variable to the right.

Penalization

Intermediate design variables do not have a clear physical interpretation so the optimization problem has to be modified if they appear in the optimal design. In the context of volume constraints/objectives, this is typically achieved using penalization. This means that, while θ is used for the volume computation, the penalized material volume factor, θ_p , is used for the material interpolation:

SIMP	$\theta_p = \theta_{\min} + (1 - \theta_{\min})\theta^p$
RAMP	$\theta_p = \theta_{\min} + \frac{\theta(1 - \theta_{\min})}{1 + q(1 - \theta)}$
Darcy	$\theta_p = \frac{q(1 - \theta)}{q + \theta}$

The SIMP (Solid Isotropic Material with Penalization) and RAMP (Rational Approximation of Material Properties) are used for solid mechanics, while the Darcy interpolation is used in fluid mechanics.

Summary

The [Density Model](#) feature adds the variables listed in [Table 2-2](#).

TABLE 2-2: VARIABLES ASSOCIATED WITH THE DENSITY MODEL FEATURE.

DESCRIPTION	NAME	INPUT TO
Control material volume factor	dtopo#.theta_c	Filter
Filtered material volume factor	dtopo#.theta_f	Projection/milling
Milling material volume factor	dtopo#.theta_m	Projection
Output material volume factor	dtopo#.theta	Penalization
Penalized material volume factor	dtopo#.theta_p	Material properties
Average material volume factor	dtopo#.theta_avg	Volume constraint

References

1. M.P. Bendsøe and O. Sigmund, *Topology Optimization Theory, Methods, and Applications*, Springer, 2004.
2. B.S. Lazarov and O. Sigmund, “Filters in topology optimization based on Helmholtz-type differential equations,” *Int. J. Numer. Methods Eng.*, vol. 86, pp. 765–781, 2011.
3. F. Wang, B.S. Lazarov and O. Sigmund, “On projection methods, convergence and robust formulations in topology optimization,” *Struct. Multidisc. Optim.*, vol. 43, pp. 767–784, 2011.

The Optimization Interface

The Optimization interface, found under the **Mathematics > Optimization and Sensitivity** branch () when adding an interface, is designed to facilitate setting up and solving advanced optimization problems. Problems that do not require least-squares contributions to the objective function, control variable fields, or pointwise constraints are preferably set up directly using only the Optimization study step node.

The optimization interface contains tools that let you set objective function, constraints, and bounds and introduce new control variable fields as well as global control variables.

In this section:

- [Adding an Optimization Interface](#)
- [The Optimization Interface](#)
- [Topology Optimization](#)
- [Shape Optimization](#)
- [Parameter Estimation](#)
- [Control Variables](#)

Adding an Optimization Interface

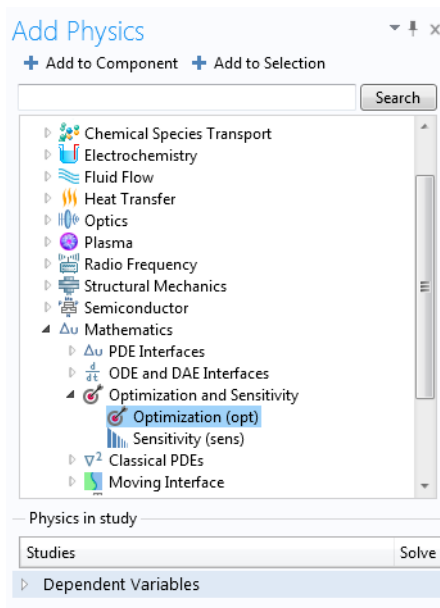
Add an **Optimization** interface when creating a new model or at any time during modeling. For a new model, physics interfaces are selected in the **Model Wizard** (after specifying the space dimension) or from the **Add Physics** window. In any active session, you can also right-click a **Component** node in the Model Builder to open the **Add Physics** window.



To add an **Optimization** interface to a Component using the **Model Wizard**, see [Creating a New Model](#) in the *COMSOL Multiphysics Reference Manual*.

To add an **Optimization** interface to a Component using the **Add Physics** window:

- 1 Under **Mathematics > Optimization and Sensitivity**, select **Optimization** ().



- 2 Click **Add to Component**. **Optimization** is added under the chosen Component in the Model Builder.

The Optimization Interface

The **Optimization (opt)** interface () contains tools for setting up advanced optimization problems. The main purpose of the interface is its ability to set up objective functions, constraint contributions, and control variables that are defined locally only on certain geometric entities, as well as least-squares contribution with a time or parameter dependence.

You can 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 geometric entities, or make them globally available. The Optimization interface itself does not have any geometry selection and is not associated with any particular space dimension. Instead you find the same set of feature nodes for domains, boundaries, edges, and points.



- [Optimization Theory](#)
- [Common Physics Interface and Feature Settings and Nodes](#) in the *COMSOL Multiphysics Reference Manual*

SETTINGS

The **Label** is the default physics interface name.

The **Name** is used primarily as a scope prefix for variables defined by the physics interface. Refer to such physics interface variables in expressions using the pattern <name>.<variable_name>. In order to distinguish between variables belonging to different physics interfaces, the name string must be unique. Only letters, numbers, and underscores (_) are permitted in the **Name** field. The first character must be a letter.

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

OPTIMIZATION INTERFACE

The following feature nodes are available in the optimization interface. Some of the features can have subnodes (available by right-clicking) as indicated by the subbullets.

- Least-Squares Objective
 - Value Column
 - Time Column
 - Parameter Column
 - Coordinate Column
 - Ignored Column
- Integral Objective (Point Sum Objective)
- Probe Objective
- Integral Inequality Constraint (Point Sum Inequality Constraint)
- Pointwise Inequality Constraint
- Control Variable Field (which includes the settings for the associated bound constraints)
 - Control Variable Bounds
- Global Objective
- Global Least-Squares Objective
- Global Inequality Constraint
- Global Control Variables

Least-Squares Objective

Uses a **Least-Squares Objective** feature to create an objective function representing the sum of squared differences between measurements, P_i , and a corresponding expression, $P(u(\xi_i), \xi_i)$, evaluated in the COMSOL Multiphysics model.

$$Q = \sum_{i=1}^N [P(u(\xi_i), \xi_i) - P_i]^2 / 2 ,$$

where ξ_i are the parameters to be estimated. The model expression is evaluated using interpolation on the feature's selection, at measurement locations specified in the data file.

	The number of coordinate columns in the data file must be the same as the dimension of the geometry, even when the selection of the Least-Squares Objective feature is on a lower dimension. In that case, model expressions are evaluated at the nearest points on the given selection.
	• Use a domain-level Least-Squares Objective node unless the model expressions corresponding to the measured data exists only on boundaries, edges, or points. • There is no need to add points to the geometry at the measurement locations specified in the file. • If your experimental data does not contain one or more columns with measurement locations, use a Global Least-Squares Objective feature instead.

To create a least-squares objective, first 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.

	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, semicolon-separated, space-separated, and tab-separated columns of measurements. 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. 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.



If you have LiveLink™ for Excel®, you can also click the **Load from Excel File** () and **Save to Excel File** () buttons.

See the *LiveLink™ for Excel® User's Guide* for more information. Or go to www.comsol.com/livelink-for-excel to learn more about the product.

Value Column

To add a **Value Column** subnode, right-click the **Least-Squares Objective** node. Use a **Value Column** to identify a column in the experimental data file as containing measurement values. Enter a corresponding **Expression**, which must be available for evaluation on the geometric entities selected in the node. Enter a corresponding **Column contribution weight**, which must be strictly positive and be available for evaluation in the global scope in the current model.

Optionally, a **Column name** can be specified to enable access to the data from the file for postprocessing. Note that the variable will be defined in only the data points (parameter value, time, and coordinates) from the file. In postprocessing, it is necessary to evaluate in the data points to ensure an adequate visualization. One option is to use a **Point Evaluation** or **Point Graph** feature with a **Cut Point ID** (2D/3D) dataset. The Cut Point dataset is available as a subnode under the **Datasets** node. In the **Point Data** section in the **Settings** window for the **Cut Point ID** (2D/3D) dataset,

you can use the **From file** option from the **Entry method** list. Other options for postprocessing the experimental data are interpolation functions and tables (use a new **Table** node with data imported from the least-squares objective file in that case). A **Unit** can also be specified if desired, using the syntax for a valid unit definition in the COMSOL Multiphysics software. The difference between the **Expression** and the value from the file is 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

To add a **Time Column** subnode, right-click the **Least-Squares Objective** node. Use a **Time Column** 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

To add a **Parameter Column** subnode, right-click the **Least-Squares Objective** node. Use a **Parameter Column** 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

To add a **Coordinate Column** subnode, right-click the **Least-Squares Objective** node. Use a **Coordinate Column** subnode to identify a column in the experimental data file as containing the global coordinates at which measurements in the value columns were made. Select an applicable coordinate and frame from the **Coordinate** and **Frame** lists. The number of coordinates must correspond to the number of dimensions in the model.

Ignored Column

To add an **Ignored Column** subnode, right-click the **Least-Squares Objective** node. Use an **Ignored Column** subnode to identify a column in the experimental data file that should not be used.

Integral Objective (Point Sum Objective)

An **Integral Objective** (or **Point Sum Objective** on points) 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.

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. For components with axi-symmetry, you can choose to **Multiply with $2\pi r$** to get the volumetric value.

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.

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 (Point Sum Inequality Constraint)

Integral Inequality Constraints (Point Sum Inequality Constraints on points) specify bounds on the value of the integral of an expression $P_{\text{int}}(\xi, u)$ taken over a selected set of geometric entities of the same dimension, Ω :

$$\text{lb}_{\text{int}} \leq \int_{\Omega} P_{\text{int}}(\xi, u) \leq \text{ub}_{\text{int}}$$

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).

For integral inequality constraints on points, the integration reduces to a summation over the selected points:

$$\text{lb}_{\text{int}} \leq \sum P_{\text{int}}(\xi, u) \leq \text{ub}_{\text{int}}$$

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.

For components with axi-symmetry, you can choose to **Multiply with $2\pi r$** to get the volumetric value.

BOUNDS

By default, the **Lower bound** and **Upper bound** checkboxes 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.

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**, **Quintic**, **Sextic**, or **Septic**.

The **Constraint method** setting controls where the constraints are evaluated:

- Choose **Elemental** to make the software assemble the constraint on each node in each element; that is, there are usually several constraints at the same global coordinates because elements in the computational mesh overlap at nodes.
- Choose **Nodal** (the default) to make the software assemble a single constraint for each global node point. The nodal constraint method provides an averaging of the constraints from adjacent elements.

The default is **Nodal** in order to minimize the number of constraints that must be handled by the optimization solvers.



The **Constraint method** setting has no effect for **Discontinuous Lagrange** shape functions whose nodes all lie strictly inside the mesh elements.

BOUNDS

By default, the **Lower bound** and **Upper bound** checkboxes 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

Add a **Control Variable Field** node to define a control variable that varies as function of position within selected geometric entities (domains, boundaries, edges, or points). The control variable field is discretized using shape functions in the same way as other dependent variables in a multiphysics model. The discrete control variable degrees of

freedom, on which the optimization solvers operate, represent values at element nodes. Right-click the node to add a **Control Variable Bounds** subnode.

CONTROL VARIABLE

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

UNIT

Specify the unit of the control variable field.

SCALING

Enter a **Scale** indicating a typical magnitude of the control variable. The relative solver tolerances refer to variables rescaled with respect to this scale, and it may also influences the search pattern of some optimization solvers.

DISCRETIZATION

This section contains settings for the shape functions 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**, **Quintic**, **Sextic**, or **Septic**.

The default choice of **Value type when using splitting of complex variables** is **Real**. This means that if the solver is set up to split complex variables in real and imaginary parts, no imaginary part is allocated for the control variable field, which is therefore guaranteed to be real. Choose **Complex** to allocate both real and imaginary parts.

Control Variable Bounds

The **Control Variable Bounds** node specifies simple bounds for its parent **Control Variable Field** node. You can only add one single **Control Variable Bounds** node for each **Control Variable Field**.

BOUNDS

By default, both lower and upper bounds are active and set to 0, which constraints the control variable field to be identically zero everywhere. Enter new **Lower bound** and

Upper bound values to specify an allowed range for the control variable, or deactivate one of the bounds to specify a one-sided bound.



Bound expressions must be parameter expressions, meaning that they must only contain numbers, model parameters, and physical constants. Such expressions can be evaluated to a number independently of the geometry and the multiphysics model solution.

Global Objective

Specify the **Global Objective** contribution to the function. To add this feature, either right-click the **Optimization** interface node and select it from the context menu, or in the **Physics** toolbar, click **Global Objective** (). In some cases, select it from the **Global** submenu.

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 compares measured data to a globally available model expression. Therefore it does not require any selection, and does not allow any **Coordinate Column** subnodes.

In addition, the **Experimental Data** section contains a **Data source** choice:

- Select **File** (the default) to take the experimental data directly from a file. Click **Browse** to locate and select the file. Add column nodes below the **Least-Squares Objective** to specify the contents of each column in the file. See [Least-Squares Objective](#), [Value Column](#), [Time Column](#), [Parameter Column](#), and [Ignored Column](#) for details.
- Select **Result table** to use experimental data from a **Table** node under **Results**. The data may have been generated by another feature under **Results**, or imported into

the table feature. In either case, the experimental data will be stored in the COMSOL model file.

- Select **Local table** to enter experimental data directly into a local table in this **Settings** window. Click the **Add** button () below the table to add another column. Rows are added automatically as you fill in the first column. To remove a column of data, select some cell in that column and click the **Delete Column** button (). Similarly, click the **Delete** button () to delete the current row. You can also save the definitions of the experimental data to a text file by clicking the **Save to file** button (). To load a text file with experimental data, use the **Load from file** button (). The file dialog allows a number of different file types, both for import and export.

When using experimental data in a **Local table** or from a **Results table**, you must choose from the **Parameter type** list whether each data row in the table corresponds to a **Time** or a **Parameter** value, and in which **Time column** or **Parameter column** the corresponding value is stored. Also specify the **Time unit**, or the **Parameter name** and **Parameter unit**, as appropriate. Finally, fill in the **Model expression** corresponding to the experimental values in each **Data column**, as well as the column **Unit** and **Weight**.

When the data source is **Results table** or **Local table**, you can also select which items in the **Data column** by clearing or selecting the corresponding checkbox in the **Use** column. You can then, for example, use the same results table to contain both experimental input and output data when performing a time-dependent optimization using a time-dependent load.

To add this feature, either right-click the **Optimization** interface node and select it from the context menu, or in the **Physics** toolbar, click **Global Least-Squares Objective** (). In some cases, select it from the **Global** submenu.



See *Mooney–Rivlin Curve Fit* for an example of fitting unknown function parameters to measured data using a global least-squares objective: Application Library path **Optimization_Module/Parameter_Estimation/curve_fit_mooney_rivlin**.

Global Inequality Constraint

Specify a **Global Inequality Constraint** that may involve both control variables and solution variables, as long as the expression is available for evaluation on the global level. To add this feature, either right-click the **Optimization** interface node and select

it from the context menu, or in the **Physics** toolbar, click **Global Inequality Constraint** (). In some cases, select it from the **Global** submenu.

CONSTRAINT

Enter a globally defined **Constraint expression** whose value is to be constrained.

BOUNDS

By default, the **Lower bound** and **Upper bound** checkboxes 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 Variables** that are globally available. To add this feature, either right-click the **Optimization** interface node and select it from the context menu, or in the **Physics** toolbar, click **Global Control Variables** (). In some cases, select it from the **Global** submenu.

CONTROL VARIABLES

In the table, enter **Variable** names, **Initial values**, **Lower bound**, and **Upper bound** of the 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 (). You can 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 that appears. To load a text file with global control variables, use the **Load from file** button () and using the **Load from File** dialog that appears. Data must be separated by spaces or tabs.



If you have LiveLink™ for Excel®, you can also click the **Load from Excel File** () and **Save to Excel File** () buttons.

See the *LiveLink™ for Excel® User's Guide* for more information. Or go to www.comsol.com/livelink-for-excel to learn more about the product.

UNIT

Specify the unit of the control variables.

SCALING

Enter a **Scale** indicating a typical magnitude of the control variables. The relative solver tolerances refer to variables rescaled with respect to this scale, and it may also influence the search pattern of some optimization solvers.

Topology Optimization

Introduction

Topology optimization problems can be solved by relaxing the discrete optimization problem to a continuous optimization problem using the density method; see [Theory for Topology Optimization](#). The [Density Model](#) feature adds a material volume factor as a control variable in a custom selection. A minimum length scale can be imposed together with common options for projection and interpolation. It is up to you as the user to use the penalized material volume factor to interpolate material properties; see the *Topology Optimization of an MBB Beam* or *Minimizing the Flow Velocity in a Microchannel* models for examples of topology optimization in structural and fluid mechanics, respectively. The following is a list of topology optimization features

- [Density Model](#)
- [Prescribed Density](#)
- [Prescribed Void](#)
- [Prescribed Material](#)
- [Prescribed Density Boundary](#)
- [Prescribed Void Boundary](#)
- [Prescribed Material Boundary](#)
- [Mirror Symmetry](#)
- [Sector Symmetry](#)

The Density Model feature uses a Helmholtz filter, its behavior can be modified using the [Prescribed Void](#), [Prescribed Material](#), [Prescribed Density](#), [Prescribed Void Boundary](#), [Prescribed Material Boundary](#) and [Prescribed Density Boundary](#). All of these features can be added via the graphics context menu.

Density Model

The **Density Model** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. It includes settings for

- [Filtering](#)
- [Coordinate System](#)
- [Manufacturing Constraints](#)

- [Extruded Boundary](#)
- [Projection](#)
- [Interpolation](#)
- [Discretization](#)

VARIABLES

The **Density Model** feature adds

- A domain control variable field (`dtopo#.theta_c`) (*control material volume factor*)
- A domain dependent variable (`dtopo#.theta_f`) (*filtered material volume factor*)
- One or several domain milling dependent variables (`dtopo#.theta_m#`) (*milling material volume factor*)
- A domain variable (`dtopo#.theta_m`) (*milling material volume factor*)
- A domain variable (`dtopo#.theta`) (*projected material volume factor*)
- A domain variable (`dtopo#.theta_p`) (*penalized material volume factor*)
- A global variable (`dtopo#.theta_avg`) (*average material volume factor*). Note that the average is computed for the selection of the **Density Model** feature only, so it is unaffected by any **Prescribed Material** or **Prescribed Void** features.

The control variable field is bounded to the interval $[0;1]$. It is used as input to a Helmholtz filter, which introduces a minimum length scale. The projected material volume factor is fixed to 1 outside the selection for the Density Model feature.



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

FILTERING

From the *filter type* list, choose between no filtering (**None**) and a Helmholtz filter (**Helmholtz**; the default), which can impose a minimum length scale by means of a *filter radius* $R_{\min}$ (SI unit: m). The local mesh element size h is the default value because this radius should not be smaller than the mesh element size. However, a fixed length scale must be used to get mesh-independent results.

COORDINATE SYSTEM

This section only appears when [Manufacturing Constraints](#) are enabled, and it allows selection of a coordinate system of [Base Vector System](#) type, so that it is easy to specify a milling direction that is not aligned with the coordinate system.

MANUFACTURING CONSTRAINTS

By default manufacturing constraints are disabled, but one can choose to enable **Extrusion** or **Milling** constraints.

Extrusion

When **Extrusion** is selected, the [Extruded Boundary](#) section appears which allows defining the extrusion direction automatically based on a boundary. Alternatively the extrusion direction can be set to a **User-defined** value.

Milling

When **Milling** is selected, the milling directions can be entered in the table. Use the [+](#) button to add more rows in the table. The *source radius* can be **User defined** or it can be **Automatic**, in which case the *filter radius* is used assuming the *filter type* is **Helmholtz**. Alternatively the local mesh element size is used.

The *aggregation exponent* can either be **User defined** or it can be **Automatic**. If it is **Automatic**, a value of 10 will be used. The *Discretization* can be **Linear** or **Constant**, and if it is linear it is possible to set a *Diffusion length*. This can either be **User defined**, or it can be **From mesh** in which case 10% or 50% of the mesh size is used for 2D and 3D, respectively.

The implementation of the milling constraints are based on [Ref. 1](#).

EXTRUDED BOUNDARY

This section is only visible with the extrusion manufacturing constraint. The selection is used to discretize the Control Variable field and the Helmholtz filter, so it is important that it is large enough that the Geometric Entity Selection does maps to an interior point. The extrusion direction is taken to be the normal, if the boundary is flat.

PROJECTION

From the **Projection type** list, specify no projection (**None**; the default) or projection based on the hyperbolic tangent function (**Hyperbolic tangent projection**). When using projection, you can choose the *projection point* and the *projection slope*. Projection with a large slope produces designs almost free of intermediate values, but the optimization problem will be difficult to solve if the slope is too large.

Milling constraints rely on projection, and therefore projection cannot be disabled when milling constraints are enabled.

INTERPOLATION

Using the **Interpolation type** list, the feature supports **RAMP** and **SIMP** interpolation for solid mechanics, while **Darcy** interpolation can be used for fluid mechanics. Finally, you can also choose a **Linear** or a **User defined** interpolation. For solid mechanics it is common to interpolate Young’s modulus, and the relative void stiffness can be bounded using the *minimum penalized volume fraction*, $\theta_{\min}$. The **SIMP exponent** p_{SIMP} and the q_{RAMP} parameters determine the stiffness for intermediate design variables.

TABLE 3-1: THE PARAMETERS ASSOCIATED WITH THE BUILT-IN INTERPOLATION TYPES OF THE DENSITY MODEL FEATURE DETERMINE THE MATERIAL PROPERTIES FOR INTERMEDIATE VALUES OF THE CONTROL VARIABLES, SO THEY CAN BE CRITICAL FOR AVOIDING UNPHYSICAL RESULTS.

TYPE	SIMP	RAMP	DARCY
Common use cases	Many, including structural mechanics	Many, including structural mechanics	Fluid mechanics
Common parameter values	$p_{\text{SIMP}} = 1, 2, 3$ Automatic= 3 $\theta_{\min} = 10^{-6}, 10^{-3}$ Automatic = 10^{-3}	$Q_{\text{RAMP}} = 2, 4, 6$ $\theta_{\min} = 10^{-6}, 10^{-3}$	$q_{\text{Darcy}} = 10^{-2}, 1$

Values in the interval [2;4] produce well-defined topologies in the context of volume-constrained compliance minimization. For fluid mechanics it is common to introduce a volume damping force, which should be large in the solid regions. The maximum value, however, depends on the fluid viscosity as well as the mesh size, so it is up to you as the user to specify this value. The **Darcy penalization** parameter controls the damping for intermediate design variables, and a value of 1 works well in the context of volume constrained dissipation maximization, but sometimes a continuation starting from a lower value can be required to avoid bad local minima.



Note that you can use the `dtopo#.theta_avg` variable to impose a volume constraint and the `dtopo#.theta` variable to write a custom material interpolation.

DISCRETIZATION

The **Linear** discretization gives a continuous representation by associating the design variables with the mesh nodes, while an elementwise **Constant** discretization gives a discontinuous representation, which generally gives rise to more variables than the **Linear** discretization.

The **Initial value** θ_0 should be set such that the starting design does not violate any optimization constraints.



Topology Optimization of an MBB Beam: Application Library path
Optimization_Module/Topology_Optimization/mbb_beam_optimization

Topology Optimization of a Beam with Milling Constraints:
Application Library path **Optimization_Module/Topology_Optimization/beam_optimization_milling**

Optimization of an Extruded MBB Beam: Application Library path
Optimization_Module/Design_Optimization/mbb_beam_extruded_optimization

Topology Optimization of a Magnetic Circuit: Application Library path
Optimization_Module/Topology_Optimization/magnetic_circuit_topology_optimization

Bracket — Topology Optimization: Application Library path
Structural_Mechanics_Module/Tutorials/bracket_topology_optimization_stl

Optimization of a Tesla Microvalve: Application Library path
Microfluidics_Module/Fluid_Flow/tesla_microvalve_optimization

Prescribed Void

The **Prescribed Void** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. The feature will cause any **Density Model** to have the selection of its filtered, projected, and penalized variables extended (or overwritten) by the **Prescribed Void** selection. If the **Density Model** uses Helmholtz filtering, the filter will be extended and use zero as input to the filter.

Prescribed Material

The **Prescribed Material** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. The feature will cause any **Density Model** to have the selection of its filtered, projected, and penalized variables extended (or overwritten) by the **Prescribed Material** selection. If the **Density Model** uses Helmholtz filtering, the filter will be extended and use one as input to the filter.

Prescribed Density

The **Prescribed Density** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. Any **Density Model** will have the selection of its filtered, projected, and penalized variables extended (or overwritten) by the **Fixed Topology Domain** selection. If the **Density Model** uses Helmholtz filtering, the filter will be extended and use 0, 1, or θ_{fix} as input to the filter, depending on the **Fixed value** setting.

FIXED VALUE

From the **Fixed value** list, choose **User-defined**, **Void**, or **Material**. The **Void** and **Material** options corresponds to the [Prescribed Void](#) and [Prescribed Material](#) features, respectively, while the **User-defined** option allows setting an arbitrary value, θ_{fix} , to be used in place of the control variable.

Prescribed Void Boundary

The **Prescribed Void Boundary** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. It acts on boundaries of [Density Model](#) features, which use Helmholtz filtering. The feature constrains the filtered field to zero on its selection.

Prescribed Material Boundary

The **Prescribed Material Boundary** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. It acts on boundaries of [Density Model](#) features, which use Helmholtz filtering. The feature constrains the filtered field to one on its selection.

Prescribed Density Boundary

The **Prescribed Density Boundary** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. It acts on boundaries of **Density Model** features, which use Helmholtz filtering. The feature constrains the filtered field to 0, 1, or θ_{fix} on its selection.

FIXED VALUE

From the **Fixed value** list, choose **User-defined**, **Void**, or **Material**. The **Void** and **Material** options corresponds to the **Prescribed Void Boundary** and **Prescribed Material Boundary** features, respectively, while the **User-defined** option allows setting an arbitrary value, θ_{fix} , to be used for the constraint.

Mirror Symmetry

The **Mirror Symmetry** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. The feature works on domains and boundaries such that any **Density Model** will have the selection of its filtered, projected, and penalized variables extended (or overwritten) by the **Mirror Symmetry** selection. The feature can thus be used to reduce design freedom relative to have a larger selection for the **Density Model** feature.

The **Mirror Symmetry** feature will only work in conjunction with **Density Model** features that have the same selection dimension.



The mapping associated with the feature can fail, if the mesh is not mirrored on curved boundaries, and this failure can manifest itself as a convergence issue when solving.

PLANE

Choose a whether the **Point on plane of reflection** should be **Automatic** or **User defined**. The **Automatic** option will use a point on a flat plane that is adjacent to the feature as well as a **Density Model**. Similarly, the **Normal vector to plane of reflection** can either be **Automatic** or **User defined**. The **Automatic** option will use the normal vector of a flat plane that is adjacent to both the feature and a **Density Model**. When both the **Mirror Symmetry** feature and the **Density Model** have a 3D boundary selection, the **Automatic** option will look for planer edges adjacent to both features and use the cross product between the vector between the end points of the edge and the normal vector of the

Mirror Symmetry selection. If the latter is not planar, a **User defined** value must be provided.



Maximizing the Eigenfrequency of a Beam: Application Library path
**Optimization_Module/Topology_Optimization/
beam_eigenfrequency_topology_optimization**

Sector Symmetry

The **Sector Symmetry** feature () can be added from the model tree under **Component** using the **Topology Optimization** context menu. Any **Density Model** will have the selection of its filtered, projected, and penalized variables extended (or overwritten) by the **Sector Symmetry** selection.

The **Sector Symmetry** feature will only work in conjunction with **Density Model** features that have the same selection dimension.

SECTOR

Provide a **Point on the revolution axis** and a **Direction of the revolution axis**. For the latter can choose between **x-axis**, **y-axis**, **z-axis**, and **User defined**. The **Transformation** can either be **Rotation and reflection** or **Rotation**. The **Rotation and reflection** option only make sense with an even number of sector. The number of sectors as well as the sector angle is computed automatically based on the selection of the **Density Model** feature.



The mapping associated with the feature can fail, if the mesh is not mirrored on curved boundaries, and this failure can manifest itself as a convergence issue when solving.

Shape Optimization

Introduction

Shape optimization can be used to deform an existing geometry to improve its performance, see [Theory for Shape Optimization](#). The technique works by deforming the mesh, so the mesh quality will often decrease, and this effect is normally what limits the magnitude of the shape deformation. Ultimately, the elements can become inverted, and if this happens, the error message in [Figure 3-1](#) is typically displayed. The following is a list of the shape optimization features

- [Free Shape Domain](#)
- [Transformation](#)
- [Free Shape Boundary](#)
- [Free Shape Shell](#)
- [Polynomial Boundary](#)
- [Polynomial Shell](#)
- [Symmetry/Roller](#)
- [Fixed Boundary](#)
- [Fixed Edge](#)
- [Fixed Point](#)
- [Mirror Symmetry](#)
- [Sector Symmetry](#)

The [Free Shape Boundary](#), [Free Shape Boundary](#), [Free Shape Shell](#), and [Polynomial Shell](#) features set up control variables for all dimensions, so a point on a boundary associated with one of these features can move within a box with side lengths equal to double the **Maximum displacement** $d_{\max}$. The features are thus free to move the boundaries in the tangential directions.

The features can be combined arbitrarily. The displacement on shared entities will be controlled by the feature with the smaller **Maximum displacement** $d_{\max}$.

It is only possible to add [Free Shape Boundary](#) or [Free Shape Boundary](#) features on the boundary of [Free Shape Domain](#) features, because these control the smoothing of domain meshes. The behavior of the features can be modified using the [Symmetry/](#)

[Roller](#), [Fixed Point](#), [Fixed Edge](#) and [Fixed Boundary](#) features. All of the shape optimization features can be added via the graphics context menu.

Free Shape Domain

The **Free Shape Domain** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. It sets up a smoothing equation for the domains in its selection, but it does not add any control variables, so unless a [Free Shape Boundary](#) or a [Polynomial Boundary](#) are added, the shape of the geometry will not change.

Boundaries will be considered fixed, if they are not part of a [Free Shape Boundary](#), [Polynomial Boundary](#), or [Symmetry/Roller](#) boundary. The interior boundaries of the **Free Shape Domain** feature are never fixed.

VARIABLES

The **Free Shape Domain** feature adds

- Area and volume variables for imposing constraints (`fsd#.area`, `fsd#.volume`)
- Area in material relative to geometry frame, and similar for the volume in three dimensions (`fsd#.relArea`, `fsd#.relVolume`)
- Post-processing variable to assess how close the controls are to the bounds (`fsd#.rel_disp`)

SMOOTHING

For the mesh smoothing, choose smoothing type from the **Mesh smoothing type** list: **Laplace**, **Winslow**, **Hyperelastic**, or **Yeoh** (the default). For **Yeoh**, also enter a **Stiffening factor** C_2 (default: 10). See [Smoothing Methods](#) in the *COMSOL Multiphysics Reference Manual* for more information.

INITIAL DEFORMATION

Enter the initial deformation in the **dx** fields (default: 0 for all components).



Optimizing a Flywheel Profile: Application Library path

Optimization_Module/Shape_Optimization/flywheel_profile

Shape Optimization of a Wrench: Application Library path

Optimization_Module/Shape_Optimization/wrench_shape_optimization

Optimization of a Waveguide Iris Bandpass Filter — Polynomial

Version: Application Library path **Optimization_Module/**

Shape_Optimization/waveguide_filter_optimization_polynomial

Optimizing the Shape of a Horn: Application Library path

Acoustics_Module/Optimization/horn_shape_optimization

Shape Optimization of a Tesla Microvalve: Application Library path

Optimization_Module/Shape_Optimization/

tesla_microvalve_shape_optimization

Transformation

The **Transformation** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. It allows connected components to be moved, scaled, and rotated. A connected component is a list of domains or boundaries sharing points. If a connected component is next to fixed points, it will not move, so all **Transformation** features have to be surrounded by void or **Free Shape Domain** features. Use **Symmetry/Roller** features to fix internal straight lines that have to remain straight.

The **Transformation** feature includes the following settings:

COORDINATE SYSTEMS

It is possible to choose coordinate systems for the translation as well as the scaling. By default the **Global Coordinate System** is used, but any (space-invariant) **Base Vector System** can be chosen.

TRANSLATION

For the translation, choose translation from the **Translate type** list: **Free** (the default), **Distance**, or **Fixed**. **Free** is associated with a table that has check columns for allowing

translation along different directions. The table also has columns for the translation bounds. By default box directions are locked, so the domain groups are fixed.

SCALING

For the scaling, choose from the **Scaling type** list: **No scaling**, **Isotropic** (the default), **Anisotropic**, **Isotropic (XY)** (3D only), **Isotropic (YZ)** (3D only), or **Isotropic (ZX)** (3D only). All options except **No scaling** are associated with a table with check columns for allowing scaling along different directions as well as columns for the scaling bounds.

ROTATION

For the rotation, choose rotation type from the **Rotation type** list: **Prescribed axis** or **Fixed** (the default). In 3D, the options **Axis in plane** and **Free rotation** are also available. The **Prescribed axis** option is associated with a setting for the upper and lower rotation bounds, while these are combined to a single option for **Axis in plane** and **Free rotation**. In 3D the **Prescribed axis** is associated with an **Axis** list: **x-axis**, **y-axis**, **z-axis** (the default), or **User defined**. The **User defined** option allows specifying the axis **x**, **y**, and **z** components. Similarly, the **Axis in plane** is associated with a **Plane** list: **xy-plane**, **yz-plane**, **zx-plane**, or **Normal vector**. The **Normal vector** option allows specifying the **x**, **y**, and **z** components of a vector defining the plane.

CENTER OF SCALING AND ROTATION

Scaling and rotation are associated with a point that you choose from the **Center type** list: **Average** (the default), **Minimum**, **Maximum**, or **User defined**.

	<i>Shape Optimization of Coils:</i> Application Library path ACDC_Module/Electromagnetics_and_Optimization/coil_shape_optimization <i>Optimization of a Waveguide Iris Bandpass Filter — Transformation Version:</i> Application Library path RF_Module/Filters/waveguide_filter_optimization_transformation <i>Shape Optimization of a Rectangular Loudspeaker Horn in 3D:</i> Application Library path Acoustics_Module/Optimization/rectangular_horn_shape_optimization
--	--

Free Shape Boundary

The **Free Shape Boundary** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. The selection of the feature has

to be adjacent to a **Free Shape Domain**. The **Free Shape Boundary** feature includes the following settings:

CONTINUITY

When the **Preserve continuity of normals at symmetry boundaries** is **Enabled**, the normal vector will be fixed at edges (points in 2D) between the **Free Shape Boundary** feature and the **Symmetry/Roller** feature. The **User-defined** option allows only fixing the normal at some of the relevant edges.

Similarly, when the **Preserve continuity of normals between free shape boundary features** is **Enabled**, no kinks will be introduced on edges (points in 2D) between different **Free Shape Boundary** features, assuming that the boundaries are initially joined continuously.

PRESERVE CONTINUITY OF NORMALS OVER SYMMETRY BOUNDARIES

This section only appears when **Preserve continuity of normals at symmetry boundaries** is **User-defined** (see previous section).

COORDINATE SYSTEM

Select the **Coordinate system for controls** among the **Base Vector System** features in the component. This section is only visible when the **Maximum displacement** is **User-defined**.

CONTROL VARIABLE SETTINGS

Choose between the **Maximum displacement** types

- **Distance**, which enforces the maximum displacement strictly
- **Box**, which confines the displacement to a box with side length equal to twice the maximum displacement.
- **User-defined**, which confines the displacement to a box with custom dimensions.

The default value for the **Maximum displacement** field is 5% of the geometry bounding box.



A large value can give rise to error messages such as that in [Figure 3-1](#).

FILTERING

Pick a **Small**, **Medium**, or **Large** filter radius $R_{\min}$, corresponding to 1, 1.5, and 2 times the **Maximum displacement** $d_{\max}$. Alternatively, you can set a **User defined** filter radius

and enter a custom value in the associated test field (default: 10% of the geometry bounding box). A small value can give rise to error messages such as that in [Figure 3-1](#).

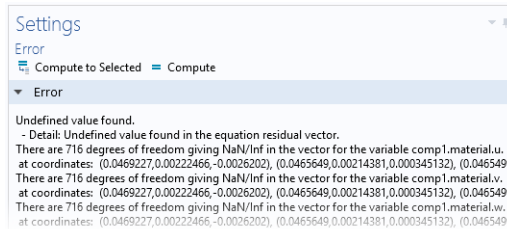


Figure 3-1: This error message is a sign that the $d_{\max}$ parameter is too large or the $R_{\min}$ variable is too small.

CONTROL VARIABLE INITIAL VALUE

You can set an initial value $\mathbf{d}_0$ for the control variable field that is different from zero, but oftentimes it is better to modify the initial geometry directly instead.

Free Shape Shell

The **Free Shape Shell** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. The selection of the feature cannot be adjacent to a **Free Shape Domain**. It is recommended not to use this feature on boundaries adjacent to meshed domains. The **Free Shape Shell** feature includes the following settings:

CONTINUITY

See the documentation for the [Continuity](#) settings of the [Free Shape Boundary](#) feature.

PRESERVE CONTINUITY OF NORMALS OVER SYMMETRY BOUNDARIES

See the documentation for the [Preserve Continuity of Normals over Symmetry Boundaries](#) settings of the [Free Shape Boundary](#) feature.

COORDINATE SYSTEM

See the documentation for the [Coordinate System](#) settings of the [Free Shape Boundary](#) feature.

CONTROL VARIABLE SETTINGS

See the documentation for the [Control Variable Settings](#) settings of the [Free Shape Boundary](#) feature.

FILTERING

See the documentation for the [Filtering](#) settings of the [Free Shape Boundary](#) feature.

	<i>Shape Optimization of a Shell:</i> Application Library path Optimization_Module/Shape_Optimization/shell_shape_optimization
	<i>Maximizing the Eigenfrequency of a Shell:</i> Application Library path Optimization_Module/Shape_Optimization/shell_eigenfrequency_shape_optimization

CONTROL VARIABLE INITIAL VALUE

See the documentation for the [Control Variable Initial Value](#) settings of the [Free Shape Boundary](#) feature.

Polynomial Boundary

The **Polynomial Boundary** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. The selection has to be adjacent to a **Free Shape Domain**, but the **Polynomial Shell** feature does not have this requirement. In 3D, the feature is only applicable to triangular and quadrilateral boundaries. The **Polynomial Boundary** feature includes the following settings:

CONTINUITY

This section is only visible when the polynomial **Type** is set to **Bernstein**. In order for the settings to have an effect, the **Order** has to be larger than 1. The first two settings are only available in 2D.

When the **Preserve continuity of normals at symmetry boundaries** is **Enabled**, the normal vector will be fixed at points between the **Polynomial Shell** feature and the **Symmetry/Roller** feature.

Similarly, when the **Preserve continuity of normals between polynomial boundaries** is **Enabled**, no kinks will be introduced on points between different **Polynomial Boundary** features, assuming that the boundaries are initially joined continuously. This also applies to points that are internal to the selection of the **Polynomial Boundary**.

Finally, when the **Preserve angles at fixed boundaries** is **Enabled**, no kinks will be introduced on points (or edges in 3D) between **Polynomial Boundary** and **Fixed Boundary** features, assuming that the boundaries are initially joined continuously.

COORDINATE SYSTEM

See the documentation for the [Coordinate System](#) settings of the [Free Shape Boundary](#) feature.

CONTROL VARIABLE SETTINGS

See the documentation for the [Coordinate System](#) settings of the [Free Shape Boundary](#) feature.

POLYNOMIAL

Choose the type of polynomial from the **Type** list: **Bernstein** (the default) or **Lagrange**. Then enter an **Order** n for the polynomial (default: 2).

Bernstein polynomials satisfy the maximum displacement bound everywhere, while **Lagrange** polynomials only satisfy it in the interpolation points and therefore **Bernstein** polynomials can be more robust.

Polynomial Shell

The **Polynomial Shell** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. The selection of the feature cannot be adjacent to a **Free Shape Domain**, but the **Polynomial Boundary** can. In 3D, the feature is only applicable to triangular and quadrilateral boundaries. It is recommended not to use this the **Polynomial Shell** feature on boundaries adjacent to meshed domains. The feature includes the following settings:

CONTINUITY

See the documentation for the [Continuity](#) settings of the [Polynomial Boundary](#) feature.

COORDINATE SYSTEM

See the documentation for the [Coordinate System](#) settings of the [Free Shape Boundary](#) feature.

CONTROL VARIABLE SETTINGS

See the documentation for the [Coordinate System](#) settings of the [Free Shape Boundary](#) feature.

POLYNOMIAL

See the documentation for the [Polynomial](#) settings of the [Polynomial Boundary](#) feature.

Symmetry/Roller

The **Symmetry/Roller** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu.

When the **Geometric entity level** is set to **Boundary**, it includes no specific settings so you only need to specify the selection of boundaries where it should be valid in the **Boundary Selection** section. Only straight edges (in 2D) and flat faces (in 3D) can be selected.

For 3D components, the **Geometric entity level** can be set to **Edge**, and then you can restrict the deformation of the edge to a plane defined by a normal vector. Only edges spanning less than 3 dimensions can be selected, and the edges have to be adjacent to a **Free Shape Shell** feature.

Edges and points shared between the feature and **Free Shape Boundary**, **Free Shape Shell**, or **Polynomial Boundary** features will be free to move tangentially to the boundaries that are part of the **Symmetry/Roller** selection.

Fixed Point

The **Fixed Point** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. It is only available in 2D and includes no specific settings so you only need to specify the selection of points where it should be valid in the **Point Selection** section.

The feature fixes points in place, but keep in mind that boundaries are automatically fixed, if they are not associated with a **Polynomial Boundary**, **Free Shape Boundary**, **Free Shape Shell**, or **Symmetry/Roller** feature.

Fixed Edge

The **Fixed Edge** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. It is only available in 3D and includes no specific settings so you only need to specify the selection of points where it should be valid in the **Edge Selection** section.

The feature fixes edges in place, but keep in mind that boundaries are automatically fixed, if they are not associated with a **Polynomial Boundary**, **Free Shape Boundary**, **Free Shape Shell**, or **Symmetry/Roller** feature.

Fixed Boundary

The **Fixed Boundary** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu.

Boundaries not associated with a shape optimization feature are automatically fixed unless they are internal to a **Free Shape Domain**. Therefore the feature mainly serves to temporarily overwrite other shape optimization features or fix boundaries internal to **Free Shape Domain** features.

Mirror Symmetry

The **Mirror Symmetry** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. The feature works on domains and boundaries such that any **Free Shape Domain**, **Transformation**, **Free Shape Shell**, or **Polynomial Shell** feature will have its displacement variables extended (or overwritten) by the **Mirror Symmetry** selection. The feature can thus be used to reduce design freedom relative to have a larger selection for (in example) a **Free Shape Domain** feature. The **Mirror Symmetry** feature will only work in conjunction with features that have the same selection dimension.

	The mapping associated with the feature can fail, if the mesh is not mirrored on curved boundaries, and this failure can manifest itself as a convergence issue when solving.
---	---

PLANE

Choose whether the **Point on plane of reflection** should be **Automatic** or **User defined**. The **Automatic** option will use a point on a flat plane that is adjacent to the feature as well as a **Density Model**. Similarly, the **Normal vector to plane of reflection** can either be **Automatic** or **User defined**. The **Automatic** option will use the normal vector of a flat plane that is adjacent to both the **Mirror Symmetry** and the source feature. When both the **Mirror Symmetry** feature and the source feature have a 3D boundary selection, the **Automatic** option will look for planer edges adjacent to both features and use the cross product between the vector between the end points of the edge and the normal vector

of the **Mirror Symmetry** selection. If the latter is not planar, a **User defined** value must be provided.

	<i>Maximizing the Eigenfrequency of a Shell:</i> Application Library path Optimization_Module/Shape_Optimization/shell_eigenfrequency_shape_optimization
	<i>Bracket — Eigenfrequency Shape Optimization:</i> Application Library path Structural_Mechanics_Module/Tutorials/bracket_eigenfrequency_shape_optimization

Sector Symmetry

The **Sector Symmetry** feature () can be added from the model tree under **Component** using the **Shape Optimization** context menu. The feature works on domains and boundaries such that any **Free Shape Domain**, **Transformation**, **Free Shape Shell**, or **Polynomial Shell** feature will have its displacement variables extended (or overwritten) by the **Sector Symmetry** selection. The feature can thus be used to reduce design freedom relative to have a larger selection for (in example) a **Free Shape Domain** feature.

The **Sector Symmetry** feature will only work in conjunction with features that have the same selection dimension.

	The mapping associated with the feature can fail, if the mesh is not mirrored on curved boundaries, and this failure can manifest itself as a convergence issue when solving.
---	---

SECTOR

Provide a **Point on the revolution axis** and the **Direction of revolution axis**. For the latter you can choose between **x-axis**, **y-axis**, **z-axis**, and **User defined**. The **Transformation** can either be **Rotation and reflection** or **Rotation**. The **Rotation and reflection** option only make sense with an even number of sectors. The number of sectors as well as the sector angle is computed automatically based on the union selection of the source features.

Parameter Estimation

Parameter Estimation can be performed using

- [Least-Squares Objective](#)
- [Global Least-Squares Objective](#)
- [The Parameter Estimation Study](#)

Introduction

The Parameter Estimation features can be used to set up objectives for parameter estimation representing the sum of squared differences between measurements, P_i , and a corresponding expression, $P(u(\xi_i), \xi_i)$, evaluated in the COMSOL Multiphysics model.

$$Q = \sum_{i=1}^N [P(u(\xi_i), \xi_i) - P_i]^2 / 2,$$

where ξ_i are the parameters to be estimated. The model expression is evaluated using interpolation on the feature's selection, at measurement locations specified in the data file.



The **Least-Squares Objective** and **Global Least-Squares Objective** features and the **Parameter Estimation** study step are available for all licenses.

Least-Squares Objective

The **Least-Squares Objective** feature () can be added from the model tree under **Component** using the **Parameter Estimation** context menu.

The **Least-Squares Objective** feature includes the following settings:

EXPERIMENTAL DATA

The settings in the **Experimental Data** section are identical to the corresponding settings in the Parameter Estimation study step, except the Data source setting does not have the **Selected Least-Squares objectives** or **All Least-Squares objectives** options. The rest of

the settings are identical, see [Experimental Data](#) under [The Parameter Estimation Study](#).

DATA COLUMN SETTINGS

The settings in the **Data Column Settings** section are identical to the [Data Column Settings](#) in [The Parameter Estimation Study](#), except the **Least-Squares Objective** supports selections and thus also coordinate columns.

EXPERIMENTAL CONDITIONS

The settings in the **Experimental Conditions** section are identical to the corresponding settings in the Parameter Estimation study step. See [Estimated Parameters](#) under [The Parameter Estimation Study](#).

Global Least-Squares Objective

The **Global Least-Squares Objective** feature () can be added from the model tree under **Global Definitions** using the **Parameter Estimation** context menu. The **Global Least-Squares Objective** feature is identical to the [Least-Squares Objective](#), except the **Global Least-Squares Objective** feature does not have a geometry or a selection and therefore coordinate columns are unsupported.

The **Global Least-Squares Objective** feature includes the following settings:

EXPERIMENTAL DATA

The settings in the **Experimental Data** section are identical to the corresponding settings in the Parameter Estimation study step, except the Data source setting does not have the **Selected Least-Squares objectives** or **All Least-Squares objectives** options. The rest of the settings are identical, see [Experimental Data](#) under [The Parameter Estimation Study](#).

DATA COLUMN SETTINGS

The settings in the **Data Column Settings** section are identical to the [Data Column Settings](#) in [The Parameter Estimation Study](#).

EXPERIMENTAL CONDITIONS

The settings in the **Experimental Conditions** section are identical to the corresponding settings in the Parameter Estimation study step. See [Estimated Parameters](#) under [The Parameter Estimation Study](#).

Control Variables

Introduction

The **Control Function** can be used to set up controls that vary depending on time or another scalar variable, while the **Control Variable Field** is used to define a control that varies spatially.

Control Function

Add a **Control Function** feature () by right-clicking a **Definitions** node in a **Component** and choosing it from the **Control Variables** submenu. The function can be accessed using its **Name**, and it defines variable equal to the function average. The **Control Function** feature includes the following settings:

INPUT

Set the **Minimum argument** and **Maximum argument** as well as the **Extrapolation**. Choose between **None** (throws error outside input bounds), **Linear**, **Constant**, **NaN**, and **Periodic**. If **Piecewise Bernstein polynomial** or **Helmholtz filter** discretization is used, the **Periodic** option causes the function to be continuous across the endpoints.

OUTPUT

Set the **Minimum value** and **Maximum value** of the function as well as the boundary conditions. Unless **Periodic** extrapolation is used, you can set the **Start boundary condition** and **End boundary condition**. The options are **None**, **Dirichlet**, **Neumann**, or **Dirichlet and Neumann**, but the latter option is not available with the **Helmholtz filter**. Finally, the **Initial value** can be set, and one can check **Add analytic function with optimized control function**, so that the result can be exported or used for further modeling.



The behavior of the **Initial value** depends on the **Control type** in the sense that the polynomial discretization will only satisfy the condition in Lagrange points, while the Helmholtz discretization just initializes the controls and thus the initial value is only approximately satisfied.

CONTROL VARIABLE DISCRETIZATION

Specify the **Control type** as **Bernstein polynomial**, **Piecewise Bernstein polynomial**, or **Helmholtz filter**. The **Helmholtz filter** can be regularized with the **Maximum slope** setting, while the **Order** can be specified for the polynomials. In the case of the **Piecewise Bernstein polynomial**, the **Number of segments** can also be defined. When the control type is a **Helmholtz filter**, one can also set the **Number of elements** to use for the control variable field. Note that there will be continuity of the slope between segments if the polynomial order is higher than 1.

UNITS

Set the units of the **Argument** and **Function**.



Shape Optimization of a Rectangular Loudspeaker Horn in 3D:
Application Library path **Acoustics_Module/Optimization/**
rectangular_horn_shape_optimization

Control Variable Field

Add a **Control Variable Field** node () by right-clicking a **Definitions** node in a **Component** and choosing it from the **Control Variables** submenu. It is also available on the **Definitions** ribbon toolbar, from the **Optimization** menu. Specify the **Control Variable Field** to add a control variable specific to the geometric entity level (domain, edge, boundary, or point) in question. The field can be accessed via its **Name** throughout the model.

GEOMETRIC ENTITY SELECTION

Here you specify the geometric entity level and the selection of geometric entities on that level for which the control variable should be defined.

INITIAL VALUE

Specify an initial value for the controls.

UNIT

Specify a unit for the controls.

BOUNDS

Specify bounds for the controls.

SCALING

The scale for the controls can affect the initial step taken by optimization solvers as well as the termination criteria. Good values could be the distance between the bounds or the approximate change between the initial and optimized values.

DISCRETIZATION

Choose **Lagrange**, **Discontinuous Lagrange**, or **Constant over geometric entities** for the shape function type. In the latter case, you can select to group adjacent entities. Otherwise, you can select the element order of the shape functions.

References

1. L.C. Høghøj and E.A. Träff, “An advection-diffusion based filter for machinable designs in topology optimization,” *Comp. Meth. Appl. Mech. Eng.*, vol. 391, p. 114488, 2022.

The Sensitivity Interface

The Sensitivity interface, found under the **Mathematics > Optimization and Sensitivity** branch () when adding an interface, 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 optimization interface lets you set objective function and to introduce the sensitivity parameters.

In this section:

- [The Sensitivity Interface](#)

The Sensitivity Interface

The **Sensitivity (sens)** interface (), found under the **Mathematics > Optimization and Sensitivity** () branch when adding a physics interface, provides tools for adding advanced sensitivity evaluation to a stationary or time-dependent model. Basic problems defined only in terms of global scalar objective functions and model parameters can be set up directly in a Sensitivity study step and therefore do not require the use of a Sensitivity interface.



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

The objective functions 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), which can be fields dependent on position in space or scalar quantities defined globally. This flexibility is reflected in the physics 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 which each corresponds a separate feature and its Settings window:

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



Note that adding a Sensitivity study step to a study makes it possible to perform a sensitivity analysis directly at the study level. See [Sensitivity](#) in the *COMSOL Multiphysics Reference Manual*.

SENSITIVITY TOOLBAR

The following nodes are available from the **Sensitivity** ribbon toolbar (Windows users), **Sensitivity** context menu (Mac or Linux users), or by right-clicking to access the context menu (all users).



For step-by-step instructions and general documentation descriptions, this is the **Sensitivity** toolbar.

TABLE 4-1: THE SENSITIVITY TOOLBAR.

BUTTON OR MENU	NAME
Physics 	
	Add Physics
Global	
	Global Objective
	Global Control Variables

The main **Settings** window for the **Sensitivity** node contains the following section:

SETTINGS

The **Label** is the default physics interface name.

The **Name** is used primarily as a scope prefix for variables defined by the physics interface. Refer to such physics interface variables in expressions using the pattern `<name>.<variable_name>`. In order to distinguish between variables belonging to different physics interfaces, the name string must be unique. Only letters, numbers, and underscores (`_`) are permitted in the **Name** field. The first character must be a letter.

The default **Name** (for the first physics interface in the model) is `sens`.

	• Common Physics Interface and Feature Settings and Nodes in the <i>COMSOL Multiphysics Reference Manual</i> • Global Objective • Global Control Variables
	<i>Sensitivity Analysis of a Communication Mast Detail</i> : Application Library path COMSOL_Multiphysics/Structural_Mechanics/mast_diagonal_mounting_sensitivity

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.

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.

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

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

CONTROL VARIABLE

The field can be accessed via its **Control variable name** throughout the model, and the **Initial value** can also be specified.

UNIT

Here you specify the unit of the field.

SCALING

If desired, you can scale the control variable using a scalar value that you enter in the **Scale** field (default: 1; that is, no scaling). A large scale will generally cause the optimization solvers to take longer steps (MMA without move limits is the exception).

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**, **Quintic**, **Sextic**, or **Septic**. For **Discontinuous Lagrange**, one can also pick **Constant** shape functions. The value type (complex or real) for all the variables defined by this **Global Equations** node is selected in the **Value type when using splitting of complex variables** selection. The default value type is **Complex**.



Common Physics Interface and Feature Settings and Nodes in the
COMSOL Multiphysics Reference Manual

Global Objective

Specify the **Global Objective** contribution to the function by entering an objective expression. To add this feature, either right-click the **Sensitivity** interface node and select it from the context menu, or in the **Physics** toolbar, click **Global Objective** ().

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

Use the **Global Control Variables** node to specify any globally available control variables. To add this feature, either right-click the **Sensitivity** interface node and select it from the context menu, or in the **Physics** toolbar, click **Global 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 ().

The Optimization Solvers

The parameter optimization study step makes it easy to get started with parameter optimization. There you can specify which objective functions, control variables, and constraints that are included in the optimization problem as well as select an optimization solver and set its most important parameters. The Parameter Estimation study step provides a simplified interface for standard parameter estimation tasks. There are also dedicated Topology Optimization and Shape Optimization study steps. Finally, the General Optimization study step is the hub of all optimization tasks.

This chapter describes the Optimization study steps, as well as the theory and detailed settings applying to the individual solvers:

- [The Parameter Optimization Study](#)
- [The Parameter Estimation Study](#)
- [The Topology Optimization Study](#)
- [The Shape Optimization Study](#)
- [The General Optimization Study](#)
- [The Curve Fitting Study](#)
- [About the Optimization Solvers](#)

- The Optimization Solver
- Cluster Computing
- Advanced Solver Properties

The Parameter Optimization Study

The **Parameter Optimization** study (p_i^o) node collects all settings necessary for solving parameter optimization problems. It serves the dual purpose of defining the optimization problem to be solved and choosing an optimization solver, as well as controlling important solver properties and solver output.

The complete optimization problem can be set up directly in the **Parameter Optimization** study node when the objective function to be minimized or maximized is a global scalar expression and the only control variables to be varied are already defined as model parameters. The study step is incompatible with the Optimization and Parameter Estimation interfaces.

When you add an **Parameter Optimization** study node to a study, it is always inserted as the first node, at the top of the study sequence (below any parametric sweeps). The study only works with derivative-free optimization; see [About Derivative-Free Solvers](#) for more information.



This section describes the Parameter Optimization study node available with the Optimization Module. See also [Studies and Solvers](#) in the *COMSOL Multiphysics Reference Manual* for more information about solvers in general. In this guide, see [About the Optimization Solvers](#) for details on the capability and settings of the individual optimization solvers.



Expressions for user-defined objective and constraint functions are evaluated in the global namespace, while the expressions for physics-defined objective and constraint expressions are evaluated in the component namespace. If there are a global parameter or function and a component variable or function with the same name (for example, `par` or `func`: `root.par` and `compi.par`, `root.func`, and `compi.func`) and a user-defined objective function or constraint with expression `f(par)` or `func`, it is evaluated as `f(root.par)` and `root.func`, while a physics-defined objective or constraint with expression `f(par)` or `func` is evaluated as `f(compi.par)` or `compi.func`. Note that the evaluation of the expressions of the user-defined objective and constraint functions in the Parameter Optimization study differs from the evaluation in Results (where the expressions are evaluated in the component namespace).

The **Settings** window has the following sections:

- [Optimization Solver](#)
- [Solver Settings](#)
- [Gaussian Process Function](#)
- [Objective Function](#)
- [Control Parameters](#)
- [Constraints](#)
- [Output](#)

OPTIMIZATION SOLVER

Select an optimization solver and specify its most important parameters.

Method

Choose an optimization solver method. The **Method** list of available solvers contains the following derivative-free methods: **Nelder–Mead**, **BOBYQA**, **COBYLA**, and **EGO**. Of these options, Nelder–Mead is the most robust, whereas EGO is good for problems with many local optima. However, both of these solvers are quite slow, particularly

EGO. The fastest option is BOBYQA, but sometimes COBYLA works better if there are constraints. BOBYQA can perform better if all constraints are scaled.



- It is better to use [The General Optimization Study](#) study step with a gradient-based solver if all of the following conditions are met
- the objectives and constraints are differentiable (no min or max operators)
 - the control parameters do not influence the geometry sequence or the mesh
 - there is only a single study step and it supports gradient-based optimization

Of these options, **Nelder–Mead** is the most robust, whereas **EGO** is good for problems with many local optima. However, both of these solvers are quite slow, particularly **EGO**. The fastest option is **BOBYQA**, but sometimes **COBYLA** works better if there are constraints.

Optimality Tolerance

Specify the relative **Optimality tolerance**. The value is applied relative to each control variable after scaling with its corresponding specified scale.

Maximum Number of Model Evaluations

Specify the **Maximum number of model evaluations**. The default is 1000. This number limits the computational time, but bounding the number of model evaluations. Note that the study will throw a warning, if the solver terminates on the maximum number of model evaluations.

SOLVER SETTINGS

See [Solver Settings](#) for [The General Optimization Study](#).

GAUSSIAN PROCESS FUNCTION

This section is visible for the EGO solver, see [Gaussian Process Function](#) for [The General Optimization Study](#).

OBJECTIVE FUNCTION

You specify the objective function for the optimization problem in the table’s **Expression** column. Enter any globally available expression that evaluates to a real number. Optionally, you can add a description in the **Description** column. Click the **Add**

Expression (+) and **Replace Expression** () buttons to search through a list of predefined expressions.



For an optimization objective that is expressed in terms of the solution u of a PDE, **Integration** (described in *COMSOL Multiphysics Reference Manual*) is one example of how you can define a scalar objective as required by the optimization solver. The evaluation of the objective function is similar to **Global Variable Probe** (described in *COMSOL Multiphysics Reference Manual*), so any variable that can be represented by a global variable probe is suitable as an objective.

The **Evaluate for** column specifies the study step for which each objective contribution will be evaluated. Available options are the last study step in the sequence plus all Study Reference nodes.

Type

Select whether to perform a **Minimization** or a **Maximization** of the objective function. The default is to minimize the objective function.

Solution

Here you select which solution or solutions to use for evaluating the objective function when several solutions are present, like for Time Dependent or Eigenvalue studies. The possible choices are **Auto** (the default), **Use first**, **Use last**, **Sum of objectives**, **Minimum of objectives**, and **Maximum of objectives**. Note that the last three options first evaluate multiple objectives according to the **Multiple objectives** setting for each solution individually, and then compute the sum, minimum, or maximum of the individual results. The **Multiple objectives** setting only appears if there are multiple objectives.

With **Auto** selected, the solver chooses the evaluation method based on the innermost study. For studies with parametric solutions (for example, studies in the **Frequency Domain**), the contributions from all solutions are summed (equivalent to the **Sum of objectives** option). For all other study types, the optimization solver selects the last solution, like the solution at the final time for a **Time Dependent** problem.

CONTROL PARAMETERS

The first table under **Control Parameters** is used to select model parameters for use as control variables. Click the **Add** (+) button to add one of the parameters defined in the **Settings** window for **Parameters** under **Global Definitions** to the set of control variables.

From a list in the **Parameter name** column, select the parameter to redefine as a control variable. Specify an **Initial value** for the control variables you add. The initial value is used as initial guess in the optimization solver and the objective function is explored around this point.

Use the **Lower bound** and **Upper bound** columns to add lower and upper bounds to the control variables. The Optimization solver only evaluates the objective function within these bounds. If you do not want to set bounds on a variable, leave the cell in the table empty.

	The Parameter Optimization study scales the control parameters based on the bounds, so bounds have to be specified. Use the The General Optimization Study to specify scales manually.
---	---

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

	If you have the LiveLink™ for Excel®, you can also save and load control parameters to and from Microsoft Excel Workbook (*.xlsx) files.
--	--

CONSTRAINTS

The first table under **Constraints** lets you specify additional constraints to be imposed on the optimum solution. The **Expression** column accepts any globally available expression which evaluates to a real number. Constraints can be functions of the control variables both directly and indirectly via PDE solution variables. The **Lower bound** and **Upper bound** columns can only contain *parameter expressions*; they must not depend on control variables, PDE solution variables, or any other user-defined variables, but can include model parameters, physical constants and units. One bound column can be left blank to indicate that no upper or lower bound is required.

If there are multiple solutions, the constraints are imposed for all them, except if the [Solution](#) setting is set to **First** or **Last** under the [Objective Function](#) section. In which

case the constraints are imposed only for the solution used to evaluate the objective function.

The **Evaluate for** column specifies the study step for which each constraint will be evaluated. Available options are the last study step in the sequence plus all Study Reference nodes.

OUTPUT

Copy Optimized Parameters to a Parameter Case

This checkbox can be used to generate a parameter case where the optimized values of the parameters are stored.

Keep Solutions

Use the **Keep solutions** list to control how to store the solutions from the individual optimization solutions.

- Select **All** to store all the optimizations solutions in memory.

Select **Only last** or **Automatic** (the default) to store only the last solution from the optimization job. If there are **Parametric Sweep** study steps in an **Parameter Optimization** study, the default option (**Automatic**) will save the last solution from the optimization job and all solutions from the parametric jobs.

Plot

Select **Plot** and choose a **Plot group** to update after each major iteration of the optimization algorithm.

Probes

Select which **Probes** to evaluate and plot in each iteration.

Keep Objective Values in Table

Select the **Keep objective values in table** checkbox to retain the table containing control variable and objective function values after the solver completes. Choose an existing **Output table**, or select **New** to create a new table. After computing the study, the **Output table** setting will be changed to the table actually being used.

Objective values displayed on the last line of the table are the converged result of the optimization when the **MMA**, **Levenberg-Marquardt**, or **IPOPT** solver is used.

Select the **Show individual objective values** checkbox to include one table column for each contribution to the objective. Otherwise, only the total objective is displayed.

Select the **Table graph** checkbox to plot the objective function values displayed in the objective table. Choose an existing **Plot window**, including the standard **Graphics** window, or select **New window**. After computing the study, the **Plot window** setting will be changed to the window actually being used.

Keep Constraint Values in Table

Only available for derivative-free methods. Select the **Keep constraint values in table** checkbox to retain the table containing global constraint values after the solver completes. Choose an existing **Constraint table**, or select **New** to create a new table. After computing the study, the **Constraint table** setting will be changed to the table actually being used.

The Parameter Estimation Study

The **Parameter Estimation** () study node provides a simplified interface for performing least-squares parameter estimation. Note that the Parameter Estimation study step can be used to fit multiple curves and it is compatible with all other study steps. This is opposed to [The Curve Fitting Study](#), which only works for time-dependent problems, and only allows fitting of a single curve. It has the following sections

- [Experimental Data](#)
- [Data Column Settings](#)
- [Objective Function](#)
- [Estimated Parameters](#)
- [Experimental Conditions](#)
- [Parameter Estimation Method](#)
- [Confidence Intervals](#)
- [Output](#)

EXPERIMENTAL DATA

When the **Data source** is a **File** (the default), you can specify a **Filename**. The data files can contain comma-separated, semicolon-separated, space-separated, and tab-separated data. You can also click the downward arrow beside the **Browse** button and choose **Browse From** () to open the fullscreen **Select File** window. Click the downward arrow for the **Location** menu () to choose **Show in Auxiliary Data** () to move to the row for this file in the **Auxiliary Data** window, **Copy File Location** (), and (if you have copied a file location) **Paste File Location** (). Click the **Import** button () to import the data from the source. Click the **Refresh** button () to update the data. You can also choose a **Decimal separator**. The separator can be either **Point** (the default) or **Comma**.

Choosing a **Local table** as the data source makes it possible to specify the table in the feature itself. You can also take the data from a **Result table**. Finally, you can use **All Least-Squares objectives**, or **Selected Least-Squares objectives**, if the objective has already

been defined in a **Global Least-Squares objective** via a **Parameter Estimation** interface under one of the components.



Note that importing a file into a **Local table** is only supported for space-delimited *.txt or comma-delimited or semicolon-delimited *.csv files.

DATA COLUMN SETTINGS

In this section there is a list with the following options for each column:

- **Time** (first default), the only setting for the time column is **Time unit**. Only one of the columns can be assigned as time column.
- **Value** (second default), this has settings for the **Model expression**, which is the value of the data according to the model in terms of global variables. For post-processing, the values in the file can be referenced using the **Column name**. It is important to specify the **Unit** of the data, if this differs from the unit system of the model. Both the data and model expression are normalized with the **Scale**.

$$\text{scale} = \sqrt{\sum_{i=1}^n (y^i - y_{\text{mean}})^2 / n}$$

(5-1)

This can be provided with the **Manual** option, it can be computed **From values** according to [Equation 5-1](#), or it can be computed using the **Automatic** setting, but then the behavior depends on the setting for the variance according to [Table 5-1](#).

TABLE 5-1: THE SCALE USED TO NORMALIZE THE LEAST-SQUARES OBJECTIVE DEPENDS ON THE SETTINGS FOR VARIANCE, IF THE SETTING FOR THE SCALE IS SET TO AUTOMATIC.

Variance \ Scale	Automatic	From values	Manual
Automatic (posterior)	Equation 5-1	Equation 5-1	Provided scale
Manual	$\sqrt{\text{variance}}$	Equation 5-1	Provided scale
Column	$\sqrt{\text{variance}}$	Equation 5-1	Provided scale

- The **Variance** can be specified for the column as a **Manual** value or another column in the data source. In the latter case, it is assumed that the unit of the variance is equal to the square of the data **Unit**. The **Variance** can also be computed based on the data using the **Automatic (posterior)** option. This setting is particularly important for the computation of confidence intervals and the covariance matrix.
- **Frequency**, the only setting for the frequency column is the **Frequency unit**. Only one of the columns can be assigned as frequency column.

- **Parameter**, this has a list of parameters in the model as well as an option to specify the **Unit**. If the parameter is constant, it is better specify the value in the section below (Experimental Parameters).
- **Ignored**, does not do anything, so there are no settings, but it can reduce the need for manipulation of the data source.

It is not possible to have both a time and a frequency column.

OBJECTIVE FUNCTION

This section includes a list of **Global Least-Squares Objective** features defined in the **Parameter Estimation** interfaces of the various components in the model.

ESTIMATED PARAMETERS

Click the **Add** () button to add one of the parameters defined in the **Settings** window for **Parameters** under **Global Definitions** to the set of parameters to be estimated. Use the **Move Up** (), **Move Down** (), and **Delete** () buttons under the table to organize the data.

From a list in the **Parameter name** column, select one of the available global parameters. Specify an **Initial value** which is used as initial guess when estimating the parameter.

The **Scale** column is important. Each control variable is rescaled with its specified scale. This means, in practice, that the solvers only get to see the parameters divided by their corresponding scale, and it is on these rescaled variables that all tolerances are applied — both user-defined and internal tolerances intended to ensure the stability of the optimization methods. The default value is **1**, which makes the solver work with the original, unscaled, parameters.

Use the **Lower bound** and **Upper bound** columns to set lower and upper bounds on the parameters. The Optimization solver only evaluates the objective function within these bounds. When doing parameter estimation, bounds can be used to avoid negative parameters or other unphysical behavior.



If you choose to add bounds to help the solver, make sure to check the solution afterward. If any of the estimated parameters has reached its bound value, then the bound should be relaxed or the parameter should be eliminated from the problem. In both cases, it is necessary to solve the modified problem one more time.

An alternative to specifying parameter names and values directly in the table is to specify them in a text file. Use the **Load from File** button () to browse to such a text file. You can also click the downward arrow beside the **Load from File** button and choose **Load From** () to open the fullscreen **Select File** window. The program appends the read names and values to the current table. The format of the text file must be such that the parameter names appear in the first column and the values for each parameter appear row-wise with a space separating the name and values, and a space separating the values. Click the **Save to File** button () to save the contents of the table to a text file (or to a Microsoft Excel Workbook spreadsheet if the license includes LiveLink™ for Excel®).

Randomize Initial Values of Bounded Global Controls

It is possible to randomize the initial values of bounded global controls. The functionality is intended to be used together with a **Parametric Sweep** over a parameter that is set as the **Random seed**, so that different optima can be identified and compared.

EXPERIMENTAL CONDITIONS

This section allows you to specify values of parameters that are constant for the data source. It is not visible, if the **Data source** is **Least-Squares objectives** or **Selected Least-Squares objectives**.

PARAMETER ESTIMATION METHOD

Select an optimization **Method** — **BOBYQA**, **Levenberg–Marquardt** (the default), or **IPOPT** — to solve the parameter estimation problem. Choose **BOBYQA** when the parameters to be estimated control the geometry, mesh, or any other aspect of the model that is not represented as a term in the model equations. Otherwise, try **Levenberg–Marquardt**, which is generally faster than IPOPT.

Specify the relative **Optimality tolerance**. The value is applied on rescaled variables, using the scale specified for each parameter. The default value of this setting varies depending on the selected optimization **Method**. The **Maximum number of model evaluations** limits the number of times the objective function is evaluated, which is related to the number of times the multiphysics system is simulated for different attempted values of the parameters.

The **Study step** to optimization over can be selected when gradient based optimization is used (not BOBYQA) and there is more than one compatible study step in the study.



The **Least-squares time/parameter list method** list only appears in a model where there is a least-squares objective.

Set the **Least-squares time/parameter list method** to choose how the parameter estimation defined times or parameters should be used. The default option is **Merge all** and this will merge all parameter estimation-defined time or parameter values with the time or parameter values defined in general parameter value lists. Alternatively, some of the data points can be excluded using the **Merge within manual range** option. If **Use only least-squares data points** is chosen, only time or parameter values defined by the parameter estimation are used and all other time or parameter values are disregarded. This also means that is impossible to add an Auxiliary sweep in the Stationary or Time Dependent study step.

For **BOBYQA**, select the **Stop if error** checkbox to stop the optimization job at once if an error occurs.

CONFIDENCE INTERVALS

The settings in the Output While Solving section are identical to the corresponding settings in the General Optimization study step. See [Confidence Intervals](#) under [The General Optimization Study](#).

OUTPUT

The settings in the Output While Solving section are identical to the corresponding settings in the General Optimization study step. See [Output](#) under [The General Optimization Study](#).

The Topology Optimization Study

The **Topology Optimization** study () node collects all settings necessary for solving topology optimization problems. It serves the dual purpose of defining the optimization problem to be solved and choosing an optimization solver, as well as controlling important solver properties and solver output.

The complete optimization problem can be set up directly in the **General Optimization** study node when the objective function to be minimized or maximized is a global scalar expression and the only control variables to be varied are already defined as model parameters. The study step is incompatible with the Optimization interface.

When you add an **Topology Optimization** study node to a study, it is always inserted as the first node, at the top of the study sequence (below any parametric sweeps). The study only works with gradient-based optimization; see [About Gradient-Based Solvers](#) for more information.



This section describes the Topology Optimization study node available with the Optimization Module. See also [Studies and Solvers](#) in the *COMSOL Multiphysics Reference Manual* for more information about solvers in general. In this guide, see [About the Optimization Solvers](#) for details on the capability and settings of the individual optimization solvers.



Expressions for user-defined objective and constraint functions are evaluated in the global namespace, while the expressions for physics-defined objective and constraint expressions are evaluated in the component namespace. If there are a global parameter or function and a component variable or function with the same name (for example, `par` or `func`: `root.par` and `compi.par`, `root.func`, and `compi.func`) and a user-defined objective function or constraint with expression `f(par)` or `func`, it is evaluated as `f(root.par)` and `root.func`, while a physics-defined objective or constraint with expression `f(par)` or `func` is evaluated as `f(compi.par)` or `compi.func`. Note that the evaluation of the expressions of the user-defined objective and constraint functions in the Topology Optimization study differs from the evaluation in Results (where the expressions are evaluated in the component namespace).

The **Settings** window has the following sections:

- [Optimization Solver](#)
- [Objective Function](#)
- [Control Variables](#)
- [Constraints](#)
- [Output](#)

OPTIMIZATION SOLVER

Select an optimization solver and specify its most important parameters.

Method

Choose an optimization solver method. The **Method** list of available solvers contains the following gradient-based methods: **GCMMA**, **MMA**, and **IPOPT**.

The different solvers are more or less suitable for different types of topology optimization problems.

Study Step

Select the **Study step** to optimize over. The setting is hidden if the study only contains a single compatible study step.

Optimality Tolerance

Specify the relative **Optimality tolerance**. The value is applied relative to each control variable after scaling with its corresponding specified scale.

Maximum Number of Iterations

Specify the **Maximum number of iterations**. The default is 100. This number limits the number of iterations. Use [The General Optimization Study](#) to constrain the number of model evaluations directly. Note that the study will not throw a warning, if the solver terminates on the maximum number of iterations.

Move Limits

Select the **Move limits** checkbox to constrain the maximum absolute change of any (scaled) control variable. Sometimes this option can prevent optimization solver from returning bad local minima. The setting is only available with the **MMA** optimization solver.

OBJECTIVE FUNCTION

You specify the objective function for the optimization problem in the table's **Expression** column. Enter any globally available expression that evaluates to a real

number. Optionally, you can add a description in the **Description** column. Click the **Add Expression** (+) and **Replace Expression** () buttons to search through a list of predefined expressions.

	For an optimization objective that is expressed in terms of the solution u of a PDE, Integration (described in <i>COMSOL Multiphysics Reference Manual</i>) is one example of how you can define a scalar objective as required by the optimization solver. The evaluation of the objective function is similar to Global Variable Probe (described in <i>COMSOL Multiphysics Reference Manual</i>), so any variable that can be represented by a global variable probe is suitable as an objective.
---	--

Type

Select whether to perform a **Minimization** or a **Maximization** of the objective function. The default is to minimize the objective function.

Solution

Here you select which solution or solutions to use for evaluating the objective function when several solutions are present, like for Time Dependent or Eigenvalue studies. The possible choices are **Auto** (the default), **Use first**, **Use last**, **Sum of objectives**, **Minimum of objectives**, and **Maximum of objectives**. Note that the last three options first evaluate multiple objectives according to the **Multiple objectives** setting for each solution individually, and then compute the sum, minimum, or maximum of the individual results.

When using **IPOPT**, **Auto** is the only choice. **MMA** supports all options except when optimizing over a **Time Dependent** study step.

With **Auto** selected, the solver chooses the evaluation method based on the innermost study.

For the studies of [Eigenvalue](#), [Eigenfrequency](#), or [Linear Buckling](#) type (all described in *COMSOL Multiphysics Reference Manual*), the first solution corresponding to the smallest eigenvalue is chosen. For studies with parametric solutions (for example, studies in the [Frequency Domain](#)), the contributions from all solutions are summed (equivalent to the **Sum of objectives** option). For all other study types, the optimization solver selects the last solution, like the solution at the final time for a [Time Dependent](#) problem.

Condition-based Final Time

For time-dependent optimization, the objective is evaluated at the final time, but it is possible to let the optimization solver control this time by specifying a **Stop expression**. Note that this will be satisfied to the accuracy of the event tolerance on the **Time-Dependent Solver**.

Objective Scaling

Most optimization solvers benefit from a well-scaled objective function, so that the actual change in the objective is neither extremely small nor extremely large. The default value is **None** and corresponds to a scale of 1, while **Manual** allows specifying a custom scale. If **Initial solution based** scaling is used, the setting interacts with the solution setting, so that the initial sum, minimum, maximum, first, or last value is chosen as the scale.

Alternatively, the scale can be incorporated directly in the definition of the objective, but that would affect the output in the log and in the objective table, while the Objective scaling setting only causes the objective to be divided by the scale internally in the solver.

CONTROL VARIABLES

If there is a **Optimization** or **Sensitivity** interface in the model, containing control variable nodes, these show up in a separate table under **Control Variables**. Use the **Solve for** column to deactivate individual control variables as needed.

CONSTRAINTS

The first table under **Constraints** lets you specify additional constraints to be imposed on the optimum solution. The **Expression** column accepts any globally available expression which evaluates to a real number. Constraints can be functions of the control variables both directly and indirectly via PDE solution variables. The **Lower bound** and **Upper bound** columns can only contain *parameter expressions*; they must not depend on control variables, PDE solution variables, or any other user-defined variables, but can include model parameters, physical constants, and units. One bound column can be left blank to indicate that no upper or lower bound is required.

If there are multiple solutions, the constraints are imposed for all them, except if the **Solution** setting is set to **First** or **Last** under the **Objective Function** section. In which case the constraints are imposed only for the solution used to evaluate the objective function.

OUTPUT

Keep Solutions

Use the **Keep solutions** list to control how to store the solutions from the individual optimization solutions:

- Select **Only last** (the default) if you want to save only the last solution.
- Select **All** if you want to save all solutions.
- Select **Last N** if you want to save the last N solutions. Then specify a positive integer in the **Number of solutions to save** field to define N (2 by default).
- Select **Every Nth** if you want to save every N^{th} solution. Specify a positive integer in the **Save every Nth** field to define N (10 by default to save every 10th solution).

Select **First and last** if you want to save the first and last solution.

Plot

Select **Plot** and choose a **Plot group** to update after each major iteration of the optimization algorithm.

Probes

Select which **Probes** to evaluate and plot in each iteration.

Keep Objective Values in Table

Select the **Keep objective values in table** checkbox to retain the table containing control variable and objective function values after the solver completes. Choose an existing **Output table**, or select **New** to create a new table. After computing the study, the **Output table** setting will be changed to the table actually being used.

Objective values displayed on the last line of the table are the converged result of the optimization when the **MMA**, **Levenberg-Marquardt**, or **IPOPT** solver is used.

Select the **Show individual objective values** checkbox to include one table column for each contribution to the objective. Otherwise, only the total objective is displayed.

Select the **Table graph** checkbox to plot the objective function values displayed in the objective table. Choose an existing **Plot window**, including the standard **Graphics** window, or select **New window**. After computing the study, the **Plot window** setting will be changed to the window actually being used.

Keep Constraint Values in Table

Only available for derivative-free methods. Select the **Keep constraint values in table** checkbox to retain the table containing global constraint values after the solver completes. Choose an existing **Constraint table**, or select **New** to create a new table.

After computing the study, the **Constraint table** setting will be changed to the table actually being used.

The Shape Optimization Study

The **Shape Optimization** study () node collects all settings necessary for solving shape optimization problems. It serves the dual purpose of defining the optimization problem to be solved and choosing an optimization solver, as well as controlling important solver properties and solver output.

The complete optimization problem can be set up directly in the **General Optimization** study node when the objective function to be minimized or maximized is a global scalar expression and the only control variables to be varied are already defined as model parameters. The study step is incompatible with the Optimization interface.

When you add an **Shape Optimization** study node to a study, it is always inserted as the first node, at the top of the study sequence (below any parametric sweeps). The study only works with gradient-based optimization; see [About Gradient-Based Solvers](#) for more information.



This section describes the Shape Optimization study node available with the Optimization Module. See also [Studies and Solvers](#) in the *COMSOL Multiphysics Reference Manual* for more information about solvers in general. In this guide, see [About the Optimization Solvers](#) for details on the capability and settings of the individual optimization solvers.



Expressions for user-defined objective and constraint functions are evaluated in the global namespace, while the expressions for physics-defined objective and constraint expressions are evaluated in the component namespace. If there are a global parameter or function and a component variable or function with the same name (for example, `par` or `func`: `root.par` and `compi.par`, `root.func`, and `compi.func`) and a user-defined objective function or constraint with expression `f(par)` or `func`, it is evaluated as `f(root.par)` and `root.func`, while a physics-defined objective or constraint with expression `f(par)` or `func` is evaluated as `f(compi.par)` or `compi.func`. Note that the evaluation of the expressions of the user-defined objective and constraint functions in the Shape Optimization study differs from the evaluation in Results (where the expressions are evaluated in the component namespace).

The **Settings** window has the following sections:

- [Optimization Solver](#)
- [Objective Function](#)
- [Control Variables](#)
- [Constraints](#)
- [Output](#)

OPTIMIZATION SOLVER

Select an optimization solver and specify its most important parameters.

Method

Choose an optimization solver method. The **Method** list of available solvers contains the following gradient-based methods: **GCMMA**, **MMA**, or **IPOPT**.

The different solvers are more or less suitable for different types of topology optimization problems.

Study Step

Select the **Study step** to optimize over. The setting is hidden if the study only contains a single compatible study step.

Optimality Tolerance

Specify the relative **Optimality tolerance**. The value is applied relative to each control variable after scaling with its corresponding specified scale.

Maximum Number of Iterations

Specify the **Maximum number of iterations**. The default is 100. This number limits the number of iterations. Use [The General Optimization Study](#) to constrain the number of model evaluations directly. Note that the study will not throw a warning, if the solver terminates on the maximum number of iterations.

Move Limits

Select the **Move limits** checkbox to constrain the maximum absolute change of any (scaled) control variable. This option can be useful in preventing some inverted elements during shape optimization. The setting is only available with the **MMA** optimization solver.

OBJECTIVE FUNCTION

You specify the objective function for the optimization problem in the table's **Expression** column. Enter any globally available expression that evaluates to a real

number. Optionally, you can add a description in the **Description** column. Click the **Add Expression** (+) and **Replace Expression** () buttons to search through a list of predefined expressions.

	For an optimization objective that is expressed in terms of the solution u of a PDE, Integration (described in <i>COMSOL Multiphysics Reference Manual</i>) is one example of how you can define a scalar objective as required by the optimization solver. The evaluation of the objective function is similar to Global Variable Probe (described in <i>COMSOL Multiphysics Reference Manual</i>), so any variable that can be represented by a global variable probe is suitable as an objective.
---	--

Type

Select whether to perform a **Minimization** or a **Maximization** of the objective function. The default is to minimize the objective function.

Solution

Here you select which solution or solutions to use for evaluating the objective function when several solutions are present, like for Time Dependent or Eigenvalue studies. The possible choices are **Auto** (the default), **Use first**, **Use last**, **Sum of objectives**, **Minimum of objectives**, and **Maximum of objectives**. Note that the last three options first evaluate multiple objectives according to the **Multiple objectives** setting for each solution individually, and then compute the sum, minimum, or maximum of the individual results.

When using **IPOPT**, **Auto** is the only choice. **MMA** supports all options except when optimizing over a **Time Dependent** study step.

With **Auto** selected, the solver chooses the evaluation method based on the innermost study.

For the studies of [Eigenvalue](#), [Eigenfrequency](#), or [Linear Buckling](#) type (all described in *COMSOL Multiphysics Reference Manual*), the first solution corresponding to the smallest eigenvalue is chosen. For studies with parametric solutions (for example, studies in the [Frequency Domain](#)), the contributions from all solutions are summed (equivalent to the **Sum of objectives** option). For all other study types, the optimization solver selects the last solution, like the solution at the final time for a [Time Dependent](#) problem.

Condition-based Final Time

For time-dependent optimization, the objective is evaluated at the final time, but it is possible to let the optimization solver control this time by specifying a **Stop expression**. Note that this will be satisfied to the accuracy of the event tolerance on the **Time-Dependent Solver**.

Objective Scaling

Most optimization solvers benefit from a well-scaled objective function, so that the actual change in the objective is neither extremely small nor extremely large. The default value is **None** and corresponds to a scale of 1, while **Manual** allows specifying a custom scale. If **Initial solution based** scaling is used, the setting interacts with the solution setting, so that the initial sum, minimum, maximum, first, or last value is chosen as the scale.

Alternatively, the scale can be incorporated directly in the definition of the objective, but that would affect the output in the log and in the objective table, while the Objective scaling setting only causes the objective to be divided by the scale internally in the solver.

CONTROL VARIABLES

If there is a **Optimization** or **Sensitivity** interface in the model, containing control variable nodes, these show up in a separate table under **Control Variables**. Use the **Solve for** column to deactivate individual control variables as needed.

CONSTRAINTS

The first table under **Constraints** lets you specify additional constraints to be imposed on the optimum solution. The **Expression** column accepts any globally available expression which evaluates to a real number. Constraints can be functions of the control variables both directly and indirectly via PDE solution variables. The **Lower bound** and **Upper bound** columns can only contain *parameter expressions*; they must not depend on control variables, PDE solution variables, or any other user-defined variables, but can include model parameters, physical constants and units. One bound column can be left blank to indicate that no upper or lower bound is required.

If there are multiple solutions, the constraints are imposed for all them, except if the **Solution** setting is set to **First** or **Last** under the **Objective Function** section. In which case the constraints are imposed only for the solution used to evaluate the objective function.

OUTPUT

Keep Solutions

Use the **Keep solutions** list to control how to store the solutions from the individual optimization solutions:

- Select **Only last** (the default) if you want to save only the last solution.
- Select **All** if you want to save all solutions.
- Select **Last N** if you want to save the last N solutions. Then specify a positive integer in the **Number of solutions to save** field to define N (2 by default).
- Select **Every Nth** if you want to save every N^{th} solution. Specify a positive integer in the **Save every Nth** field to define N (10 by default to save every 10th solution).

Select **First and last** if you want to save the first and last solution.

Plot

Select **Plot** and choose a **Plot group** to update after each major iteration of the optimization algorithm.

Probes

Select which **Probes** to evaluate and plot in each iteration.

Keep Objective Values in Table

Select the **Keep objective values in table** checkbox to retain the table containing control variable and objective function values after the solver completes. Choose an existing **Output table**, or select **New** to create a new table. After computing the study, the **Output table** setting will be changed to the table actually being used.

Objective values displayed on the last line of the table are the converged result of the optimization when the **MMA**, **Levenberg-Marquardt**, or, **IPOPT** solver is used.

Select the **Show individual objective values** checkbox to include one table column for each contribution to the objective. Otherwise, only the total objective is displayed.

Select the **Table graph** checkbox to plot the objective function values displayed in the objective table. Choose an existing **Plot window**, including the standard **Graphics** window, or select **New window**. After computing the study, the **Plot window** setting will be changed to the window actually being used.

Keep Constraint Values in Table

Only available for derivative-free methods. Select the **Keep constraint values in table** checkbox to retain the table containing global constraint values after the solver completes. Choose an existing **Constraint table**, or select **New** to create a new table.

After computing the study, the **Constraint table** setting will be changed to the table actually being used.

The General Optimization Study

The **Optimization** study () node collects all settings necessary for solving optimization problems:

- Optimization Solver
- Solver Settings
- Gaussian Process Function
- Confidence Intervals
- Objective Function
- Control Variables and Parameters
- Constraints
- Output

The study serves the dual purpose of defining the optimization problem to be solved and choosing an optimization solver, as well as controlling important solver properties and solver output.

The complete optimization problem can be set up directly in the **General Optimization** study node when the objective function to be minimized or maximized is a global scalar expression, and the only control variables to be varied are already defined as model parameters. If the model requires control variables or objective functions that depend on position in the geometry, general least-squares objective contributions, or pointwise constraints, these must be set up separately using an Optimization interface. Such contributions are displayed in the **General Optimization** study node settings where they can be individually disabled or enabled.

When you add a **General Optimization** study node to a study, it is always inserted as the first node, at the top of the study sequence (below any parametric sweeps). Remaining nodes in the sequence define the multiphysics problem on which the optimization process will act. If this sequence contains **Study Reference** nodes and you are using a derivative-free solver, you can choose for each objective function contribution and

constraint whether to compute its value for the main sequence or for the sequence pointed to by one of the study references.

	A derivative-free optimization solver can be combined with a parametric sweep, but in general it is only possible to use one Sensitivity, Optimization, Parameter Estimation, or Parametric Sweep study step in each study. These study nodes tend to control the same solver settings and are therefore incompatible with each other. To perform parametric or nested optimization, you can call a study containing an Optimization node from inside another study, via a Study Reference node.
	This section describes the General Optimization study node available with the Optimization Module. See also Studies and Solvers in the <i>COMSOL Multiphysics Reference Manual</i> for more information about solvers in general. In this guide, see About the Optimization Solvers for details on the capability and settings of the individual optimization solvers.
	Expressions for user-defined objective and constraint functions are evaluated in the global namespace, while the expressions for physics-defined objective and constraint expressions are evaluated in the component namespace. If there are a global parameter or function and a component variable or function with the same name (for example, <code>par</code> or <code>func</code> : <code>root.par</code> and <code>comp.i.par</code> , <code>root.func</code> , and <code>comp.i.func</code>) and a user-defined objective function or constraint with expression <code>f(par)</code> or <code>f(func)</code> , it is evaluated as <code>f(root.par)</code> and <code>root.func</code> , while a physics-defined objective or constraint with expression <code>f(par)</code> or <code>f(func)</code> is evaluated as <code>f(comp.i.par)</code> or <code>comp.i.func</code> . Note that the evaluation of the expressions of the user-defined objective and constraint functions in the General Optimization study differs from the evaluation in Results (where the expressions are evaluated in the component namespace).

The **Settings** window has the following sections:

OPTIMIZATION SOLVER

Select an optimization solver and specify its most important parameters.

Method

Choose an optimization solver method. The **Method** list of available solvers contains both gradient-based and derivative-free methods.

- Derivative-free (gradient-free) optimization options: **Nelder–Mead** (the default), **BOBYQA**, **COBYLA**, and **EGO**.
- Gradient-based optimization options: **IPOPT**, **GCMMA**, **MMA**, and **Levenberg–Marquardt**.

The different solvers are more or less suitable for different types of optimization problems. There are also differences in which problem features they can handle. Objective contributions, control variables, and constraints that are not compatible with the selected solver are marked by a warning sign (⚠) in the first column of the corresponding table.



The result of a derivative-free optimization solver can be continued in a new study by right-clicking the General Optimization study node and choosing **Add New Optimization Study with Current Optimization Solution**.

Optimality Tolerance

Specify the relative **Optimality tolerance**. The value is applied relative to each control variable after scaling with its corresponding specified scale. The default value of this setting varies depending on the selected optimization **Method**.

Study Step

The gradient-based solvers can optimize only over the output from a single study step, and not for all study step types. Allowed types are those for which the default solver is a Stationary Solver, Eigenvalue Solver, or a Time-Dependent Solver. Note that this includes the Frequency Domain study step, which uses a stationary solver. The Eigenvalue Solver only supports objectives and constraints expressed using the eigenvalue (optimization of the eigenvectors is unsupported). Finally, the Time-Dependent Solver evaluates the objective at the end time.

Choose any of the available study steps from the **Study step** list. Click the **Go to Source** button (🔗) to move to the **Settings** window for the selected study node. Note that this setting is only visible when the study contains more than one compatible study step.

Solver Settings

Under **Solver settings** you can specify the following optimization solver settings:

Specify the **Maximum number of model evaluations**. The default is 1000. This number limits the number of times the objective function is evaluated, which is related to the number of times the multiphysics system is simulated for different values of the optimization control parameter.



The number of objective evaluations is not equal to the number of iterations taken by the optimizer because each iteration can invoke more than a single objective function evaluation.

Least-Squares Time/Parameter List Method

If least-squares objectives are defined, you can specify the **Least-squares time/parameter list method**. The default is **Merge all**. In that case, all least-squares-defined time or parameter values are merged with the time or parameter values defined in general parameter value lists. It is possible to exclude some of the times/parameters using the **Merge within manual range** option, see also [Least-Squares Data](#). The last option is **Use only least-squares data points**. In that case, only least-squares defined time or parameter values are used and all other time or parameter values are disregarded.

SOLVER SETTINGS

Distribute Parametric Sweep

If you are using a solver that supports batch evaluation of objective values, you can choose to distribute a parametric sweep by sending one parameter value to each compute node over the nodes in a cluster. To enable this functionality, first click the **Show More Options** button () and select **Batch and Cluster** in the **Show More Options** dialog, then select the **Distribute parametric sweep** checkbox. If desired, adjust the value in the **Maximum number of model evaluations in each Parametric Sweep** field (default: 1). This setting is available with the **Nelder–Mead** and **EGO** optimization solver.

Stop if error

For derivative free optimization, select the **Stop if error** checkbox to stop the optimization job at once if an error occurs.

Objective Scaling

Most optimization solvers benefit from a well-scaled objective function, so that the actual change in the objective is neither extremely small nor extremely large. The setting is available for all optimization solvers, except Levenberg–Marquardt. The default value is **None** and corresponds to a scale of 1, while **Manual** allows specifying a custom scale. If **Initial solution based** scaling is used, the setting interacts with the

solution (and multiple objectives) setting, so that the initial sum, minimum, maximum, first, or last value is chosen as the scale. Note that the setting is not used when the MMA optimization solver is used with least-squares objectives.

Alternatively, the scale can be incorporated directly in the definition of the objective, but that would affect the output the in the log and in the objective table, while the Objective scaling setting only causes the objective to be divided by the scale internally in the solver.

Randomize Initial Values of Bounded Global Controls

If the optimization solver is not EGO, it is possible to randomize the initial values of bounded global controls using a Sobol sequence. The functionality is intended to be used together with a **Parametric Sweep** over a parameter that is set as the **Quasi-random index shift**, so that different optima can be identified and compared.

	<i>Optimization of a Waveguide Iris Bandpass Filter — Transformation</i> Version: Application Library path Optimization_Module/ Shape_Optimization/waveguide_filter_optimization_transformation
	Note that BOBYQA does not respect initial values of controls variables close to the bounds. This can somewhat limit the exploration of the design space using randomization of initial control values.

Constraint Settings

For some optimization solvers, you can select a **Constraint handling method**. The possible options are **Penalty** and **Augmented Lagrangian**. The former is the default when available and requires no further settings. Choosing the **Augmented Lagrangian** method activates additional options to control its behavior and accuracy. See [Constraint Handling for Derivative-Free Methods](#) for further details.

For derivative-free optimization solvers that support the **Penalty** constraint handling method (Nelder–Mead) and for COBYLA you can select the **Enforce design constraints strictly** checkbox to evaluate design constraints before the forward problem is solved and avoid running the forward problem if there are infeasible design constraints. All constraints that do not depend on the solution of the forward problem are considered to be design constraints. With the **Enforce design constraints strictly** checkbox selected (the default), the solver evaluates all design constraints before the forward problem is

run. If infeasible constraints are found, the forward problem is not run, NaN is reported as objective, and the optimization solver proceeds to the next iteration.

Gaussian Process Function Settings

The Gaussian process function settings are available with the **EGO** solver. See [Surrogate Models — Gaussian Process](#) in the *Uncertainty Quantification User's Guide* for more information.

GAUSSIAN PROCESS FUNCTION

This section is visible for the EGO solver.

Specify the **Usage**. The default is **Restart**. In this case, the EGO solver starts from sampling new points from the control variables and iteratively improves the objective value. If you select **Improve**, the solver will reuse the previously trained Gaussian process function (specified under the **Gaussian process function** and **Constraint Gaussian process function** if constraints are present) to refine the solution.

Specify the **Initial sample per control variable**. The default is 10. This parameter defines the number of initial evaluation for the objective function, which influences the accuracy of the initial Gaussian process function.

From the **Covariance** list, choose **Squared exponential**, **Matérn 3/2** (the default), **Matérn 5/2**, or **Single-layer neural network**. From the **Mean** list, choose **Constant** (the default), **Linear**, or **Quadratic**.

From the **Gaussian process function** list, choose **New** to create a new **Gaussian Process** function for the objective value or choose any existing Gaussian process function.

From the **Constraint Gaussian process function** list, choose **New** to create a new **Gaussian Process** function for the constraint value or choose any existing constraint Gaussian process function.

See [Surrogate Models — Gaussian Process](#) in the *Uncertainty Quantification User's Guide* for more information.

An optimization approach is used to maximize the acquisition function to find the next sample point. Use the **Optimization method for acquisition function** list: **Direct** (the default) or **Monte Carlo**. The **Direct** method is normally preferred because it is faster and gives a better estimate. In each iteration of this method, the number of Gaussian process evaluations can change. Initially, only a few evaluations are done, but the number can quickly grow. Therefore, you can set a bound, both on the number of iterations and the total number of Gaussian process evaluations. Enter values in the

Maximum surrogate evaluations for optimization (default: 10,000) and **Maximum number of optimization iterations** fields (default: 500).

You can specify the **Resolution tolerance** to prevent the selection of new sample points that are too close to existing ones.

CONFIDENCE INTERVALS

This section is visible for the Levenberg–Marquardt solver. Select the **Compute confidence intervals** checkbox to retain

- the table containing individual confidence intervals for every global control parameter.
- the table containing the covariance matrix (unit system of the model).

Specify the **Confidence level** (0.95 by default, meaning the 95% confidence interval). Choose an existing **Confidence interval table** and **Covariance matrix table**, or select **New** to create new tables. If an existing table has been chosen, the data in the table will be overridden. Each row in the confidence interval table corresponds to one global control parameter in the order they are given in tables in the **Control Variables and Parameters** section, assuming that the confidence interval for each parameter has been computed. When the confidence interval for some parameter cannot be computed a warning is given. The confidence intervals table is also printed out in the log. Confidence intervals for individual parameters are computed using the t -test statistic and the linear approximation method (Ref. 2). The confidence interval for an individual parameter θ_j for a confidence level $1 - \alpha$ is given by

$$\left\{ \theta_j \pm t_{\left(1 - \frac{\alpha}{2}, n - p\right)} \sigma_j \right\}$$

where σ_j is the standard deviation of the parameter θ_j , n is the number of data points, and p is the number of parameters to identify (Ref. 2). Here,

$$t_{\left(1 - \frac{\alpha}{2}, n - p\right)}$$

is the percentage point of a t -variable with $n - p$ degrees of freedom that leaves a probability of $\alpha/2$ in the upper tail and $1 - \alpha/2$ in the lower tail. The standard deviation σ_j is computed using a linear approximation of the Hessian, as given in Ref. 1. Moreover, it is assumed that the observations on the independent variable are measured without error and that the random errors in the dependent variable are normally and independently distributed. If no variance column is present, the variance

of the random errors is estimated from the data. If a variance column is present, this is used as the error for computation of the (parameter) covariance matrix, D . This can be computed using a posteriori estimator for the nondimensional bulk variance, $\hat{\sigma}_y^2$,

$$D = \hat{\sigma}_y^2 (J^T W J)^{-1}, \quad (5-2)$$

where W are the weights taken as the inverse of the squared scales and

$$J = \partial \mathbf{y} / \partial \mathbf{p}$$

is the sensitivity of the simulated measurements, $\mathbf{y}$, to the control parameters, $\mathbf{p}$. Alternatively, the variance can be provided manually and then the computation depends on the settings for the scale. If the scale is also provided manually, the covariance matrix is computed according to

$$D = W_J J^T W \Sigma W J W_J, \quad (5-3)$$

where Σ is the (measurement) covariance matrix, and

$$W_J = (J^T W J)^{-1}. \quad (5-4)$$

Finally, the scale can be computed automatically, in which case a simplified formula is used

$$D = (J^T \Sigma J)^{-1} \quad (5-5)$$

This formula requires that the inverse variance is used as weights, so the scale has to be **Automatic** and the variance has to be specified manually or using a data column. [Table 5-2](#) summarizes the relationship between the computation of the covariance matrix and the settings for the scale and variance.

TABLE 5-2: THE EQUATION USED FOR COMPUTATION OF THE COVARIANCE MATRIX DEPENDS ON THE SETTINGS FOR THE SCALE AND THE VARIANCE.

Variance \ Scale	Automatic	From Values	Manual
Automatic (posterior)	Equation 5-2	Equation 5-2	Equation 5-2
Manual	Equation 5-5	Equation 5-3	Equation 5-3
Column	Equation 5-5	Equation 5-3	Equation 5-3

No confidence intervals or covariance matrix are computed for control variable fields or when value columns with different settings for the scale or variance are combined.

OBJECTIVE FUNCTION

You specify the objective function for the optimization problem in the table’s **Expression** column. Eigenvalue problems are limited to objectives expressed in terms of the eigenvalue. Otherwise, you can enter any globally available expression that evaluates to a real number. Optionally, you can add a description in the **Description** column. Click the **Add Expression** (+) and **Replace Expression** () buttons to search through a list of predefined expressions.

	For an optimization objective that is expressed in terms of the solution u of a PDE, Integration (described in <i>COMSOL Multiphysics Reference Manual</i>) is one example of how you can define a scalar objective as required by the optimization solver. The evaluation of the objective function is similar to Global Variable Probe (described in <i>COMSOL Multiphysics Reference Manual</i>), so any variable that can be represented by a global variable probe is suitable as an objective.
---	--

The **Evaluate for** column specifies the study step for which each objective contribution will be evaluated. Available options are the last study step in the sequence plus all Study Reference nodes. When a gradient-based solver is used, the column is not active: all contributions are then evaluated for the same study step.

If there is an **Optimization** or **Sensitivity** interface in the model, containing objective function nodes, these show up in a separate table under **Objective Function**. Use the **Active** column to deactivate individual contributions as needed. Contributions not supported by the currently selected solver are marked by a warning sign () in the first column, which disappears if you manually deactivate the objective.

Type

Select whether to perform a **Minimization** or a **Maximization** of the objective function. The default is to minimize the objective function.

Multiple Objectives

If you have defined more than one objective function, choose how to evaluate the overall objective: Choose from **Sum of objectives** (the default), **Minimum of objectives**, or **Maximum of objectives**. Note that not all options are available together with all solvers.

Solution

Here you select which solution or solutions to use for evaluating the objective function when several solutions are present, like for Time Dependent or Eigenvalue studies. The

possible choices are **Auto** (the default), **Use first**, **Use last**, **Sum of objectives**, **Minimum of objectives**, and **Maximum of objectives**. Note that the last three options first evaluate multiple objectives according to the **Multiple objectives** setting for each solution individually, and then compute the sum, minimum, or maximum of the individual results.

All options are available only with the derivative-free solvers. When using **IPOPT**, or **Levenberg–Marquardt**, **Auto** is the only choice. All options are only supported with **MMA**, but not when optimizing over a **Time Dependent** study step.

With **Auto** selected, the solver chooses the evaluation method based on the innermost study. For the studies of **Eigenvalue**, **Eigenfrequency**, or **Linear Buckling** type (all described in *COMSOL Multiphysics Reference Manual*), the first solution corresponding to the smallest eigenvalue is chosen. For studies with parametric solutions (for example, studies in the **Frequency Domain**), the contributions from all solutions are summed (equivalent to the **Sum of objectives** option). For all other study types, the optimization solver selects the last solution, like the solution at the final time for a **Time Dependent** problem, but note that least-squares objectives often have contributions for multiple values of the time parameter.

Condition-based Final Time

When gradient based optimization solvers are combined with a time-dependent study step, the objective is evaluated at the final time, but it is possible to let the optimization solver control this time by specifying a **Stop expression**. Note that this will be satisfied to the accuracy of the event tolerance on the **Time-Dependent Solver**.

Outer Solution

When there is a parametric sweep or a material sweep below the optimizations study step, the objective is evaluated using the **Outer solutions** from the sweep. The options are between the **Minimum of objectives**, **Maximum of objectives**, or **Sum of objectives** (the default).

CONTROL VARIABLES AND PARAMETERS

The first table under **Control Variables and Parameters** is used to select model parameters for use as control variables. Click the **Add** (**+**) button to add one of the parameters defined in the **Settings** window for **Parameters** under **Global Definitions** to the set of control variables.

From a list in the **Parameter name** column, select the parameter to redefine as a control variable. Specify an **Initial value** for the control variables you add. The initial value is

used as initial guess in the optimization solver and the objective function is explored around this point.

The **Scale** column is important. Each control variable is rescaled with its specified scale. This means, in practice, that the solvers only get to see the control variables divided by their corresponding scale, and it is on these rescaled variables that all tolerances are applied — both user-defined and internal tolerances intended to ensure the stability of the optimization methods. The default value is 1, which makes the solver work with the original, unscaled, variables.

Use the **Lower bound** and **Upper bound** columns to add lower and upper bounds to the control variables. The Optimization solver only evaluates the objective function within these bounds. If you do not want to set bounds on a variable, leave the cell in the table empty.



The Optimization solver determines whether bounds are allowed or not; the MMA solver also needs bounds but can estimate them automatically — but at a cost.

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



If you have the LiveLink™ for Excel®, you can also save and load control parameters to and from Microsoft Excel Workbook (*.xlsx) files.

If there is an **Optimization** or **Sensitivity** interface in the model, containing control variable nodes, these show up in a separate table under **Control Variables and Parameters**. Use the **Solve for** column to deactivate individual control variables as needed. Variables not supported by the currently selected solver are marked by a warning sign () in the first column, which disappears if you manually choose not to include it in the solution.

CONSTRAINTS

The first table under **Constraints** lets you specify additional constraints to be imposed on the optimum solution. For eigenvalue problems the **Expression** column only accepts the eigenvalue. Otherwise, you can enter any globally available expression which evaluates to a real number. Constraints can be functions of the control variables both directly and indirectly via PDE solution variables. The **Lower bound** and **Upper bound** columns can only contain *parameter expressions*; they must not depend on control variables, PDE solution variables, or any other user-defined variables, but can include model parameters, physical constants and units. One bound column can be left blank to indicate that no upper or lower bound is required. Derivative-free optimization solvers will output orange rows in the objective probe table, if any constraints are violated.

If there are multiple solutions, the constraints are imposed for all them, except if the **Solution** setting is set to **First** or **Last** under the **Objective Function** section. In which case the constraints are imposed only for the solution used to evaluate the objective function.

The **Evaluate for** column specifies the study step for which each constraint will be evaluated. Available options are the last study step in the sequence plus all Study Reference nodes. When a gradient-based solver is used, the column is not active: all contributions are then evaluated for the same study step.

If there is an **Optimization** interface in the model, containing inequality constraint nodes, these also show up in a separate table under **Constraints**. Use the **Active** column to deactivate individual constraints as needed. Constraints not supported by the currently selected solver are marked by a warning sign (▲) in the first column, which disappears if you manually choose to deactivate the constraint.

OUTPUT

Copy Optimized Parameters to a Parameter Case

This checkbox can be used to generate a parameter case where the optimized values of the parameters are stored.

Keep Solutions (Derivative Free)

For the derivative-free optimization methods — **Nelder–Mead**, **BOBYQA**, **COBYLA**, and **EGO** — use the **Keep solutions** list to control how to store the solutions from the individual optimization solutions.

- Select **All** to store all the optimizations solutions in memory.
- Select **Only last** or **Automatic** (the default) to store only the last solution from the optimization job. If there are **Parametric Sweep** study steps in a **General Optimization** study, the default option (**Automatic**) will save the last solution from the optimization job and all solutions from the parametric jobs.

Keep Solutions (Gradient Based)

For the gradient-based optimization methods — **IPOPT**, **MMA**, and **Levenberg–Marquardt** — use the **Keep solutions** list to control how to store the solutions from the individual optimization solutions:

- Select **Only last** (the default) if you want to save only the last solution.
- Select **All** if you want to save all solutions.
- Select **Last N** if you want to save the last N solutions. Then specify a positive integer in the **Number of solutions to save** field to define N (2 by default).
- Select **Every Nth** if you want to save every N^{th} solution. Specify a positive integer in the **Save every Nth** field to define N (10 by default to save every 10th solution).
- Select **First and last** if you want to save the first and last solution.

For **All**, **Last N**, **Every Nth**, and **First and last**, the `optSolInd` parameter is created. This parameter can be used to access individual solutions with the `withSol` operator (for advanced postprocessing).

Optimization Log

From the **Optimization log** list, choose **Minimal**, **Normal** (the default), or **Detailed** to control the amount of information output from the optimization solver and the inner solvers it calls.

Probes

Select which **Probes** to evaluate and plot in each iteration.

Keep Objective Values in Table

This setting is available with the **Nelder–Mead**, **BOBYQA**, **COBYLA**, and **EGO** solvers.

Select the **Keep objective values in table** checkbox to retain the table containing control variable and objective function values after the solver completes. Choose an existing

Output table, or select **New** to create a new table. After computing the study, the **Output table** setting will be changed to the table actually being used.

The derivative-free solvers add a new line to this table for each evaluation of the objective function, while the gradient-based solvers add a new line for every accepted step. When using derivative-free methods, the control variable values are also displayed in the table. Objective values displayed on the last line of the table are the converged result of the optimization when the **Nelder–Mead**, **BOBYQA**, **EGO**, **MMA**, **Levenberg–Marquardt**, or **IPOPT** solver is used.

Select the **Show individual objective values** checkbox to include one table column for each contribution to the objective. Otherwise, only the total objective is displayed.

Select the **Table graph** checkbox to plot the objective function values displayed in the objective table. Choose an existing **Plot window**, including the standard **Graphics** window, or select **New window**. After computing the study, the **Plot window** setting will be changed to the window actually being used.

Keep Constraint Values in Table

Only available for derivative-free methods. Select the **Keep constraint values in table** checkbox to retain the table containing global constraint values after the solver completes. Choose an existing **Constraint table**, or select **New** to create a new table. After computing the study, the **Constraint table** setting will be changed to the table actually being used.

The derivative-free solvers add a new line to this table for each evaluation of the objective function. When using the **Nelder–Mead**, **BOBYQA**, or **EGO** solver, the control variable and constraint values displayed on the last line correspond to the converged result of the optimization.

REFERENCES FOR THE CONFIDENCE INTERVAL

1. N. Börlin, “Nonlinear Optimization. Least Squares Problems — The Gauss-Newton Method,” Umeå University, 2007.
2. J.O. Rawlings, S.G. Pantula, and D.A. Dickey, *Applied Regression Analysis: A Research Tool*, 2nd Edition, Springer Texts in Statistics, Springer-Verlag, New York, 1998.

The Curve Fitting Study

The **Curve Fitting** () study node provides a simplified interface for performing least-squares parameter estimation. It can be used when the reference data is a function of time or a single parameter, and the multiphysics model result expected to match the data is a single global expression evaluated for a selected study step in the same study.

When the reference data consists of measured values, you typically first import it as an interpolation function which you can easily plot and visually compare to the result of your multiphysics model. The Curve Fitting study node can refer directly to the interpolation function and is independent of whether the interpolation function was specified directly in the user interface or imported from a file, and is also independent of the file format used.

Alternately, the reference data can be given as a user-defined expression which is evaluated at the time steps or parameter values specified in the corresponding study step. This is useful, for example, when estimating coefficients of a polynomial expected to match the model output, and in general when estimating parameters in a mathematical model intended to replicate the output of the full multiphysics model.

The parameter estimation problem is implemented as an optimization problem minimizing a sum of squared differences between model and reference data. The sum is computed over time or parameter steps as specified either in the argument column of an interpolation function used as reference data, or in the study step selected for evaluation of the model data.

MODEL DATA

Choose one of the allowed study steps from the **Study step** list. Only time-dependent study steps are supported. Note that the setting is only visible when the study contains more than one time-dependent study step.

Click the **Go to Source** button () to move to the **Settings** window for the selected study node. Enter a global **Model expression**, which is evaluated at each time or parameter step where it is compared to the corresponding reference data. You can also click the **Insert Expression** () and **Replace Expression** () buttons to search through a list of predefined expressions.

REFERENCE DATA

Select a **Reference data source**: **Interpolation function** or **User defined**. If you choose an interpolation function as reference data source, you can pick an interpolation function in the **Reference function** list. The argument column of the interpolation function will decide at which time or parameter values the difference between model and reference data is evaluated. When the data source is set to **User defined**, enter a **Reference expression** or click the **Replace Expression** () buttons to search through a list of predefined expressions. You can also press Ctrl+Space to add a predefined expression to the text field. The difference between the **Model expression** and the **Reference expression** is then evaluated at output time steps or parameter values as specified in the selected **Study step**.

ESTIMATED PARAMETERS

The settings in the Parameters section are identical to the corresponding settings in the Parameter Estimation study step. See [Estimated Parameters](#) under [The Parameter Estimation Study](#).

PARAMETER ESTIMATION METHOD

The settings in the Parameter Estimation Method section are identical to the corresponding settings in the Parameter Estimation study step. See [Parameter Estimation Method](#) under [The Parameter Estimation Study](#).

CONFIDENCE INTERVALS

The settings in the Output While Solving section are identical to the corresponding settings in the General Optimization study step. See [Confidence Intervals](#) under [The General Optimization Study](#).

OUTPUT

3. The settings in the Output While Solving section are identical to the corresponding settings in the General Optimization study step. See [Output](#) under [The General Optimization Study](#).

About the Optimization Solvers

The Optimization Module provides a selection of optimization solver algorithms which can be divided into two main groups: on one hand *gradient-based* solvers and on the other hand *derivative-free* solvers. The two groups are suitable for different types of problems and have different performance characteristics.

In this section:

- [About Derivative-Free Solvers](#)
- [About Gradient-Based Solvers](#)
- [The Nelder–Mead Solver](#)
- [The BOBYQA Solver](#)
- [The COBYLA Solver](#)
- [The EGO Solver](#)
- [The IPOPT Solver](#)
- [The MMA Solver](#)
- [The Levenberg–Marquardt Solver](#)
- [About the Convergence Plot and Progress Window](#)
- [About Optimality Tolerances](#)
- [About Constraint Handling](#)
- [References for the Optimization Solvers](#)

About Derivative-Free Solvers

The defining characteristic of the derivative-free solvers is that they do not need to compute derivatives of the objective function with respect to the control variables. They do not even require the objective function to be differentiable in principle. This characteristic makes them suitable for problems where the objective function is nonsmooth or contains noise.

One typical example of a noisy objective function is when the control variables define geometry dimensions. The geometry changes induced by modifying the control variables then lead to different finite element meshes, superimposing different discretization errors on the objective function when evaluated for different control variable values.

Since the derivative-free solvers do not trust the pointwise behavior of the objective function to be a good indicator of where to search for the next, improved, update to the control variables, they must rely on sampling the objective function at different positions in the control variable space. This is more expensive than following a single path toward the optimum but also more robust. Some of the performance penalty is offset by the fact that evaluations that do not depend on one another can be done in parallel — for example, in a cluster environment.

Finally, derivative-free solvers can be further subdivided into *local*, “hill-climbing,” methods and *global*, evolutionary or statistical, methods. The former type starts from an initial guess and strives to improve the objective function in a stepwise manner. Imagine a group of people trying to climb a hill together in dense fog; as long as they stay together and move upward, they find a top but not necessarily the highest one. Global methods, in contrast, try to produce a map of the entire design space, refining it iteratively in areas that appear to be good candidates for containing the global optimum.

The Optimization Module provides five different derivative-free algorithms:

- The **Nelder–Mead** solver walks toward improved objective function values by iteratively replacing the worst corner of a simplex in the control variable space. See [The Nelder–Mead Solver](#).
- The **BOBYQA** solver walks toward improved objective function values by using an iteratively constructed quadratic approximation of the objective. See [The BOBYQA Solver](#).
- The **COBYLA** solver solves a sequence of linear approximations constructed from objective and constraint values sampled at the corners of a simplex in control variable space. See [The COBYLA Solver](#).
- The **EGO** solver iteratively searches for improved objective function values by constructing a Gaussian process surrogate model to approximate the objective function. See [The EGO Solver](#).

These methods are each described in more details below.

About Gradient-Based Solvers

The defining characteristic of a gradient-based solver is that it follows a path in the control variable space where each new iterate is based on local derivative information evaluated at previously visited points. The methods implemented in the Optimization Module require the complete vector of first-order derivatives of the objective function

with respect to the discrete vector of control variable degrees of freedom, which is referred to as the discrete *gradient* of the objective function in the control variable space.

The gradient can be computed in different ways. In general, the **Adjoint** method is the most efficient (and also the default), followed by the **Forward** method. The pure **Numeric** method is the most expensive as it is based on repeated solution of the multiphysics problem, while the **Forward numeric** method requires only repeated assembly of the problem residual. The **Adjoint** method is required for problems with many controls, but it is not supported in all situations, and it can be slower than the **Numeric** method for transient problems with few controls.

The Optimization module provides three different gradient-based algorithms:

- The **IPOPT** solver is a general purpose solver suitable for dealing with large-scale problems with many or difficult constraints. See [The IPOPT Solver](#).
- The **MMA** solver can handle problems of any form and is especially suitable for problems with a large number of control variables, such as topology optimization. See [The MMA Solver](#). It is a first-order method, so termination on the iteration count is often needed.
- The **Levenberg–Marquardt** solver is specifically designed for solving least-squares problems. See [The Levenberg–Marquardt Solver](#).

These methods are each described in more details below.

The Nelder–Mead Solver

The Nelder–Mead solver relies on a simplex of $N + 1$ points, where N is the number of control variables. The solver does not use derivatives of the objective function. In a Nelder–Mead iteration, the solver uses reflections, expansions, and contractions to improve the worst point in the simplex.

The implementation in COMSOL Multiphysics includes a restart procedure for the case when the simplex shape degenerates (that is, the simplex collapses along a direction) and follows the discussion of the Nelder–Mead method in Ch. 8 in [Ref. 1](#) employing the parallelization strategy described in [Ref. 2](#). The sufficient-decrease condition used is a combination of the one used in Algorithm 8.2 and the one described on p. 160 (due to Kelley) with the regeneration. This implementation has been found to give somewhat better performance than the plain Algorithm 8.2 on a few test examples, in particular because the regeneration of a new simplex with equal lengths on edges gives better search directions. Moreover, the solver respects lower

and upper bounds in the control variable space by suitably restricting the length of reflections.

The BOBYQA Solver

The name BOBYQA is an acronym for *bound optimization by quadratic approximation*. The basic idea of the method is to iteratively approximate the objective function by a quadratic model which is valid in a region around the current iterate, the so-called *trust region*. The quadratic model is updated by minimizing the Frobenius norm of the difference in the Hessians of the two consecutive quadratic approximations. The implementation in COMSOL is based on [Ref. 3](#). There are, however, some modifications:

- The number of interpolation conditions is fixed to $2n+1$ where n is the number of control variables, since the updating of the quadratic approximation requires only $O(n^2)$ operations in that case.
- Since the COMSOL implementation works in scaled control variables, the initial trust region radius is fixed at 0.2 in relative terms.
- The RESCUE subroutine is not included due to the unavailability of test problems that would invoke that procedure. Instead, an error message is given.

The COBYLA Solver

The name COBYLA is an acronym for *constrained optimization by linear approximation*. It is an iterative method for derivative-free constrained optimization. The method maintains and updates linear approximations to both the objective function and to each constraint. The approximations are based on objective and constraint values computed at the vertices of a well-formed simplex. Each iteration solves a linear programming problem inside a *trust region* whose radius decreases as the method progresses toward a constrained optimum.

Note that COBYLA treats simple bounds as constraints, which can cause bound violations.

Further details can be found in [Ref. 4](#).

The EGO Solver

The name EGO is an acronym for *efficient global optimization*. The core concept of this method is to iteratively approximate the objective function using a Gaussian

process surrogate model, which predicts both the objective function value and the associated uncertainty at un-sampled points. The Gaussian process is refined by sequentially selecting new sampling points based on an Expected Improvement acquisition function, which balances exploration (sampling where the model is uncertain) and exploitation (sampling where the model predicts improve objective values).

Further details can be found in [Ref. 5](#).

The IPOPT Solver

The IPOPT solver uses a gradient-based optimization technique to find optimal solutions to a very general class of optimization problems. It requires external computation (analytically or seminumerically) for the gradients of both the objective function and all constraints.

The underlying algorithm is an implementation of an interior point line search filter for general nonlinear programming problems:

$$\begin{aligned}
 & \min_{\mathbf{x} \in R^m} && f(\mathbf{x}) \\
 & \text{s. t.} && \mathbf{g}^L \leq \mathbf{g}(\mathbf{x}) \leq \mathbf{g}^U \\
 & && \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \quad \text{for } i \in \mathbf{t}
 \end{aligned} \tag{5-6}$$

where f is the objective, $\mathbf{x}$ are the controls, $\mathbf{g}$ is the constraint function. IPOPT reformulates the problem to eliminate inequality constraints by introducing the slack variables, $\mathbf{s}$, with bounds, $\mathbf{g}^L \leq \mathbf{s} \leq \mathbf{g}^U$, and the auxiliary equation, $\mathbf{g}(\mathbf{x}) = \mathbf{s}$. The equivalent reformulation is

$$\begin{aligned}
 & \min_{\mathbf{x} \in R^m, \mathbf{s} \in R^n} && f(\mathbf{x}) \\
 & && \mathbf{g}(\mathbf{x}) = \mathbf{s} \\
 & \text{s. t.} && \mathbf{g}^L \leq \mathbf{s} \leq \mathbf{g}^U \\
 & && \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U
 \end{aligned}$$

which in terms of the augmented design variable, $\tilde{\mathbf{x}} = [\mathbf{x}, \mathbf{s}]^T$, and the equality constraints, $\mathbf{c}(\tilde{\mathbf{x}}) = \mathbf{g}(\mathbf{x}) - \mathbf{s}$, can be written as

$$\begin{aligned}
& \min_{\tilde{\mathbf{x}} \in R^{m+n}} f(\tilde{\mathbf{x}}) \\
& \text{s. t.} \quad \mathbf{c}(\tilde{\mathbf{x}}) = \mathbf{0} \\
& \quad \tilde{\mathbf{x}}^L \leq \tilde{\mathbf{x}} \leq \tilde{\mathbf{x}}^U
\end{aligned}$$

where $\tilde{\mathbf{x}}^L = [\mathbf{x}^L, \mathbf{g}^L]^T$ and $\tilde{\mathbf{x}}^U = [\mathbf{x}^U, \mathbf{g}^U]^T$. The first-order optimality conditions for the problem (Karush–Kuhn–Tucker, KKT) become

$$\nabla f + \nabla \mathbf{c} \cdot \lambda + \mu^U - \mu^L = \mathbf{0} \quad (5-7)$$

$$\mathbf{c}(\tilde{\mathbf{x}}) = \mathbf{0} \quad (5-8)$$

$$\begin{aligned}
& 0 \leq \mu_j \\
& x_j \mu_j = 0
\end{aligned} \quad (5-9)$$

where λ , μ^U and μ^L are Lagrange variables associated with the equality and inequality constraints, respectively.

TERMINATION

The error used for convergence depends on [Equation 5-7](#), [Equation 5-8](#) and [Equation 5-9](#).

$$E = \max \left\{ \frac{\|\nabla f + \nabla \mathbf{c} \cdot \lambda + \mu^U - \mu^L\|_\infty}{s_d}, \|\mathbf{c}\|_\infty, \frac{\max\{x_j \mu_j\}}{s_c} \right\} \quad (5-10)$$

where s_d and s_c are internal scales that serve to avoid numerical problems

$$s_d = \max \left\{ 1, \frac{\|\lambda\|_1 + \|\mu\|_1}{(m+n)s_{\max}} \right\} \quad s_c = \max \left\{ 1, \frac{\|\mu\|_1}{ns_{\max}} \right\}$$

where $s_{\max} = 100$. The convergence checks required for termination are

- $E \leq \text{tol}$
- $\|\nabla f + \nabla \mathbf{c} \cdot \lambda + \mu^U - \mu^L\|_\infty \leq \text{dual_inf_tol}$

- $\|c\|_{\infty} \leq \text{constr_viol_tol}$
- $\max\{x_j, \mu_j\} \leq \text{compl_inf_tol}$

The IPOPT log contains the error, E , and the maximum infeasibility (MaxInfeas), $\|c\|_{\infty}$.

The MMA Solver

The MMA implementation in the Optimization Module is the globally convergent version of the method of moving asymptotes, referred to as GCMMA in [Ref. 7.](#), but it is also possible to use the classical implementation (without an inner loop) described in [Ref. 8.](#)



MMA Method of Moving Asymptotes, GCMMA, Globally Convergent MMA, and Globally Convergent Method of Moving Asymptotes authored by Krister Svanberg. Copyright © 2013 Krister Svanberg.

This is a three-level algorithm:

- Outer iteration k uses the current control variable estimate, x_k , to evaluate objective function, constraints, and their gradients, which are used together with current asymptote estimates, l_k and u_k , to construct an approximating subproblem. This subproblem, which is guaranteed to be convex and feasible, is passed to the inner iterations.
- Each inner iteration m solves an approximating subproblem for its unique optimum x_{km} and then evaluates the true objective function and constraints at this point. If the approximating subproblem is found to be conservative compared to the true function values, the inner iteration is terminated and the point is accepted as the next outer estimate x_{k+1} . Otherwise, the approximating subproblem is modified to make it more conservative and then passed to the next inner iteration.
- The subproblem in each inner iteration is solved using a dual active set strategy. The approximating subproblems are nonlinear and inequality-constrained, but have a special structure which makes solving the primal problem for fixed dual variables very fast. From the solution to the primal problem, a gradient and full Hessian can be computed for the dual problem, which is solved using a modified Newton active set algorithm.

Note that function (objective and constraints) gradients are computed strictly once in each outer iteration, while function values must be computed once for each extra inner

iteration required. The innermost level sees only an analytical approximating form of the subproblem where current function and gradient estimates appear in various coefficients.

The special structure of the generated approximating subproblems influences the global behavior of the algorithm. In contrast to the IPOPT and Levenberg–Marquardt solvers, which rely on approximating second-order information about the objective function, MMA is essentially a linear method. Its subproblems are linear approximations to the original problem but with barrier-like rational function contributions controlled by the moving asymptotes. No information about the problem is retained between outer iterations except the current position of the asymptotes.

In practice, this means that MMA does not show the quadratic convergence close to the optimum associated with Newton-like methods. In fact, there are very simple problems dominated by a quadratic term in the objective function for which MMA converges very slowly or not at all. In particular, in order for MMA to work efficiently, least-squares problems must be formulated using Least Squares Objective features in an Optimization interface. These features trigger a reformulation of the problem to a form that is more suitable for MMA. Moreover, if the defect is complex valued, only the real part is used by MMA internally.

Because of the linear approximation of the objective function, the first inner iteration in each outer MMA iteration effectively steps into a corner of the feasible set, where it is completely bound by constraints and simple bounds. If this point is found to be nonconservative, as is the case if the objective function is convex with an optimum in the interior of the feasible set, the inner iteration generates a series of iterates gradually moving away from the constraints until a conservative point is found. This behavior favors points close to the constraints, in contrast to the line search used in IPOPT and the trust region in Levenberg–Marquardt which favor points close to the previous iterate. If the objective function has multiple local minima, the different methods can therefore be expected to find different local solutions.

For further details, see [Ref. 7](#), which you can find under `<COMSOL_root>/doc/pdf/Optimization_Module/gcmmma.pdf`, where `<COMSOL_root>` is the root folder of your COMSOL installation.

The Levenberg–Marquardt Solver

The Levenberg–Marquardt solver works exclusively with objective functions of least-squares type. Parameter bounds are supported, but general constraints are not.

Because this method is designed specifically for solving problems of least-squares type, it typically converges faster than IPOPT for such problems. Note, however, that the sensitivity computation is relatively cheaper for IPOPT with time-dependent least-squares objectives, and therefore IPOPT is often preferred for those problems.

The objective function is

$$V(\eta) = \frac{1}{2} \sum_{m=1}^M \sum_{j=1}^{J_m} \sum_{k=1}^{K_{jm}} \frac{f_{jm}^2(x_{jmk}, u_m(x, p_{jm}, \eta), \eta, C_m)}{s_{jm}^2} \quad (5-11)$$

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 x is the space coordinates, η are the parameters for which the cost function should be minimized and $u_m(x, p, \eta)$ solves a given PDE or ODE. The variable p is time if the PDE or ODE is time dependent, but it can also represent any parameter when the forward problem is stationary. Each measurement series can be associated with a set of parameters, C_m . The functions s_{jm} are the scales, 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

$$f_{jm}(x_{jmk}, u_m(x, p_{jm}, \eta), \eta, C_m) = g_{jm}(x_{jmk}, u_m(x, p_{jm}, \eta), \eta, C_m) - \hat{g}_{jmk} \quad (5-12)$$

The Levenberg–Marquardt algorithm as implemented in the Optimization Module relies on two fundamental ideas: evaluation of an approximate Hessian and regularization of the Hessian approximation. The special structure of least-squares objective functions allows cheap evaluation of an approximate Hessian (matrix of second derivatives), which can in principle be used directly in a Newton iteration. However, least-squares problems are also often ill-conditioned, making the full Newton process unstable. Therefore, the Hessian is modified using a regularization parameter to guarantee its positive definiteness. This parameter is updated between iterations, based on the success or failure of the previous step. For further details, see [Ref. 9](#).

About the Convergence Plot and Progress Window

The gradient-based solvers create a **Convergence Plot** that displays the number used for the **Convergence** criteria as a function of the iteration numbers; see [About Optimality Tolerances](#).

An extra line is also added to the **Progress** window, which shows the number used for the **Convergence** criteria. This line also shows the objective, and next to it there is a **Stop**

button that allows premature termination of the optimization algorithm. Finally, the forward sensitivity method is associated with an extra line in the **Progress** window where the control number is displayed; see [Forward Sensitivity](#).

About Optimality Tolerances

The optimality tolerance is an important setting for all optimization solvers. It is intended to represent the relative accuracy in the final scaled control variable values, but because of the wide differences between different solver implementations, uniform behavior cannot be guaranteed.

In particular, the optimality tolerance can play tricks on you if your objective function or your optimization variables are badly scaled. Therefore, take care to specify correct scales for your control variables and make sure that objective functions and constraints are of order 1 — or at least not too far from — for reasonable values of the control variables.

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.

OPTIMALITY TOLERANCE FOR DERIVATIVE-FREE METHODS

For the derivative-free optimization methods, the optimization tolerance, with a default value of 0.01, is used to determine whether a stationary point has been reached. The **BOBYQA**, **COBYLA**, and **Nelder–Mead** methods stop iterating as soon as no improvement over the current best estimate can be found with steps in the scaled control variables of relative size larger than or equal to the optimality tolerance. The **EGO** method stops iterating as soon as no improvement over the weighted average of the ten most recent scaled estimates larger than the optimality tolerance can be found.

Compared to gradient-based optimization methods, which improve based upon the gradient of the objective function with respect to control variables, derivative-free

methods explore the region around the current point by function evaluations only and use that information for determining convergence.



The returned point is not necessarily located close to a stationary point to within the optimality tolerance. When problems are badly scaled or functions are nonsmooth (for example, because of noise in the objective evaluation), the algorithms might miss an opportunity for improvement that requires an absolute step length larger than the optimality tolerance times the specified scale. In case of convergence problems, try to reduce the optimality tolerance, or choose a different initial condition.

OPTIMALITY TOLERANCE FOR IPOPT

There are three tolerances for IPOPT:

- *Dual infeasibility absolute tolerance* measures the sensitivity of the objective for unconstrained problems. For constrained problems, the sensitivity of the objective can be arbitrarily large at optimality, so an equation involving the constraints is used instead; see [The IPOPT Solver](#).
- *Constraint violation absolute tolerance*, which can be decreased to reduce constraint violations. However, note that scaling of the objective and constraint functions can often achieve the same effect without introducing other numerical issues.
- *Complementary conditions absolute tolerance* is similar to the dual infeasibility except it also considers the bounds on the controls, because the sensitivity with respect to controls with active bounds can be arbitrarily large at optimality.

See also [IPOPT-Specific Settings](#) and [Termination](#) under the [The IPOPT Solver](#).

OPTIMALITY TOLERANCE FOR MMA

The MMA solver terminates when the relative change in all scaled control variables is less than the specified optimality tolerance parameter, with a default of $1e-3$. The relative change is defined as the change in the variable since the last outer iteration divided by the range of the variable. The range of the variable is the upper bound minus the lower bound. For unbounded variables, the MMA solver internally estimates bounds based on the previous iteration points.

OPTIMALITY TOLERANCE FOR LEVENBERG–MARQUARDT

Let tol be the specified optimality tolerance. Define $\text{tol}_d = \gamma_d \cdot \text{tol}$, where γ_d is the defect reduction tolerance factor, and $\text{tol}_x = \gamma_x \cdot \text{tol}$, where γ_x is the *control variable tolerance factor*. Moreover, let the defect vector be defined by

$$(d_l)_{l=1}^L = \sqrt{\omega_l} f_l$$

where ω_l and f_l are defined in [Equation 5-11](#) and [Equation 5-12](#), and L is the total number of the measurement evaluations. Then, when the Levenberg–Marquardt solver is used, the following conditions are used to determine when optimality has been reached:

- Terminate when the defect has been reduced enough; that is,

$$\frac{\|d_j\|_2}{\|d_0\|_2} \leq \text{tol}_d$$

where d_0 is the initial defect vector, and d_j is the current defect vector.

- Terminate when the relative increment of the scaled control variable x is below the control variable tolerance; that is,

$$\|x_j - x_{j-1}\|_2 \leq \text{tol}_x$$

This is referred to as the **stepsize** in the **Convergence Plot** and in the **Log Window**

- Terminate when the maximum cosine between the defect and the Jacobian columns is below the optimality tolerance; that is

$$\max \left(\frac{(J^T \cdot d_j)_i}{\|J(:, i)\|_2 \|d_j\|_2} \right) \leq \text{tol}$$

where d_j is the current defect vector and J is the Jacobian. This is referred to as the **errJ** in the **Convergence Plot** and in the **Log Window**.

The default values of the optimality tolerance, defect reduction tolerance factor, and control variable tolerance factor are $1.0 \cdot 10^{-3}$, 1, and 1, respectively. The termination condition defined as the first condition above is not used by default and should be enabled in order to be included. The minimum of the later two, $\min(\text{errJ}, \text{stepsize})$, is shown in the **Convergence Plot**.

About Constraint Handling

An important difference between the available optimization solvers is the types of constraints they can handle, and how they do it. Constraints specified in the **Optimization** interface and optimization study steps can be divided into three categories:

- *Simple bounds* are upper and lower bounds prescribed directly on the individual control variables, for example in the **General Optimization** study step.
- *Pointwise constraints* specify limits on an expression to be enforced at every node point in some region in space. In order to avoid excessively expensive gradient computations, such constraints are required to only depend on control variables directly and not indirectly via PDE solution variables. The constraint expression can, however, be a nonlinear expression in the control variables.
- *General constraints* specify limits on global scalar expressions, typically evaluated as integrals over some domain. This generates a single constraint, as compared to one for each node point for a pointwise constraint. Therefore, the solvers can afford to compute a complete gradient also when the constraint is a, possibly nonlinear, function of the PDE solution.

Note that all constraints are treated as inequalities. An equality constraint can be implemented by specifying the same upper and lower bound for an expression. However, not all constraint handling methods are able to deal with the reduction in control variable space dimension which this implies. Therefore, when possible, it is better to perform a manual change of variables, eliminating a control variable dimension.

CONSTRAINT HANDLING FOR DERIVATIVE-FREE METHODS

The derivative-free methods **Nelder–Mead**, **COBYLA**, and **EGO** can internally handle simple bounds and general constraints on global scalar expressions. In practice this means that in addition to simple bounds and constraints defined in the optimization study steps, **Integral Inequality Constraint** nodes and **Global Inequality Constraint** nodes in Optimization interfaces are accounted for.

The constraint handling algorithm used in **Nelder–Mead** is in principle based on filtering out candidate points in the control variable space which fall outside the feasible region, and to some extent adjust search directions accordingly. The procedure is not guaranteed to find a constrained local minimum fulfilling the KKT conditions.

COBYLA, in contrast, approximates objective function and constraints in a uniform way. Therefore, provided all functions are sufficiently smooth, it will in general find an approximate constrained local minimum. The returned solution may, however, lie slightly outside the feasible set. This can happen, in particular, if the constraints are nonlinear.

The constraint handling algorithm in **EGO** is based on incorporating the constraints directly into the surrogate model using Gaussian process functions. The algorithm models both the objective function and constraints, balancing their improvement when selecting new sample points using the acquisition function. While the procedure improves constraint satisfaction as the model evolves, the returned solution may still lie slightly outside the feasible region, especially in the early stages or if the constraints are highly nonlinear.

The Augmented Lagrangian Method

BOBYQA handles simple bounds internally, but general constraints only via an external iterative procedure based on repeated minimization of an augmented Lagrangian. This augmented Lagrangian method can also be used as an alternative to the internal penalty method in the **Nelder–Mead** solver, but is not selected by default.

The basic principle behind the augmented Lagrangian method is to include the Lagrange multipliers of general constraints as control variables in an augmented problem. In the first iteration, the Lagrange multipliers are set to zero and a modified objective function including a quadratic penalty for constraint violation is minimized. This gives a solution which is in general outside the feasible set, that is, it violates the constraints. In each subsequent iteration, the Lagrange multipliers — as well as a number of penalty parameters — are updated based on the current constraint violation, and a new subproblem is solved. The sequence of subproblems, which converges toward the feasible set from the outside, is terminated once a specified tolerance for constraint violation is reached.

The augmented Lagrangian method as such is very general and quite robust, but to be efficient, it requires balancing the effort spent on each subproblem against the improvement in each outer iteration. It also requires selection of an initial penalty factor. A higher penalty factor generally leads to faster convergence of the augmented Lagrangian algorithm, but subproblems become more ill-conditioned and there is a threshold beyond which the method may become unstable; conversely, a lower penalty factor makes the algorithm more robust but required more iterations.

In the **Settings** window for the optimization study steps, you can choose to use an **Automatic** or a **Manual** definition of the initial **Penalty parameter** p . The automatic

setting is default and computes an initial value based on the objective and constraint function values at the initial point. You can also limit the **Maximum number of augmented iterations** and select a strategy for updating δ (**tolerance for the subsolver**). The options **Dynamic I** and **Dynamic II** both tighten the subsolver tolerance from iteration to iteration, the latter providing some additional control. There is also a **Manual** option. Finally, specify the **Constraint tolerance**, that is, the maximum allowable constraint violation in the final solution.

Since the augmented Lagrangian method computes Lagrange multipliers for each constraint explicitly, these are also available for postprocessing. Their values represent the sensitivity (derivative) of the objective function with respect to changes in a constraint bound. In the **Insert Expression** and **Add Expression** menus, available for most postprocessing features, you will find the Lagrange multipliers under **Model > Solver > Lagrange multipliers**.

CONSTRAINT HANDLING FOR GRADIENT-BASED METHODS

The **IPOPT** algorithm supports constraints and bounds by using an interior point line search method, see [IPOPT-Specific Settings](#).

The **MMA** algorithm accepts constraints of the same general type as IPOPT, requiring an accurate constraint Jacobian, but handles them differently. In each outer, major, iteration, linear and linearized constraints are combined with a linearized objective function into a convex smooth approximation whose unique optimum is always feasible unless the feasible set is empty. The globally convergent version of MMA implemented in the Optimization module is conservative in a way which ensures that each major iterate is feasible not only with respect to the linearized constraints, but with respect to the fully nonlinear constraints.



In the MMA implementation, simple bounds are much less expensive memory-wise than global or pointwise constraints. In particular for control variable fields, simple control variable bounds (added by right-clicking on the **Control Variable Field** feature) are more efficient than enforcing the same bounds using a **Pointwise Inequality Constraint** feature.

The **Levenberg–Marquardt** solver supports bounds but not constraints.

References for the Optimization Solvers

1. A.R. Conn, K. Scheinberg, and L.N. Vicente, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, 2009.
2. D. Lee, and M. Wiswall, “A Parallel Implementation of the Simplex Function Minimization Routine,” *Comput. Econ.*, vol. 30, pp. 171–187, 2007.
3. M.J.D. Powell: *The BOBYQA algorithm for bound constraint optimization without derivatives*, Report DAMTP 2009/NA06, University of Cambridge, UK, 2009.
4. M.J.D. Powell: “A direct search optimization method that models the objective and constraint functions by linear interpolation,” *Proc. Sixth Workshop on Optimization and Numerical Analysis*, vol. 275, pp. 51–67, Kluwer Academic Publishers, Dordrecht, NL, 1994.
5. D.R. Jones, M. Schonlau, and W.J. Welch, “Efficient global optimization of expensive black-box functions,” *J.Glob. Optim.*, vol. 13, pp. 455–492, 1998.
6. A. Wächter and T.B. Lorenz, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
7. K. Svanberg, *MMA and GCMMA – Fortran versions March 2013*, KTH, Royal Institute of Technology, Stockholm, 2013.
8. K.Svanberg, “The Method of Moving Asymptotes – a new Method for Structural Optimization,” *Int. J. Numer. Methods Eng.*, vol. 24, no. 2, pp. 369–373, 1987.
9. K. Madsen, H.B. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems*, 2nd ed., 2004.
10. R. Andreani, E.G. Birgin, J.M. Martinez, and M.L. Schuverdt, “On Augmented Lagrangian Methods with general lower-level constraints,” *SIAM J. Optim.*, vol. 18, pp. 1286–1309, 2007.



Ref. 7 is available in `<COMSOL_root>/doc/pdf/Optimization_Module/gcmmma.pdf`, where `<COMSOL_root>` is the root folder of your COMSOL installation.

The Optimization Solver

The **Optimization Solver** node () contains settings for selecting a gradient-based optimization method and specifying related solver properties:

- [General](#)
- [Optimization Solver](#)
- [Transient Adjoint](#)
- [Output](#)
- [Results While Solving](#)
- [Advanced](#)
- [Constants](#)
- [Log](#)



This section describes **Solver** features available with the Optimization Module. See also [Studies and Solvers](#) in the *COMSOL Multiphysics Reference Manual* for more information about solvers in general.



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

For a more extensive treatment of the gradient-based solvers available in this node, see [About Gradient-Based Solvers](#).

GENERAL

The **Optimality tolerance**, **Maximum number of model evaluations**, and **Method** settings are fundamental and can be controlled from an optimization study step.

Defined by Study Step

Choose to let an optimization study step control the fundamental optimization method settings (the default). For **User Defined** specify the settings directly in this node.

Optimality Tolerance

Specify the **Optimality tolerance**, which has default value **1e-3**. See [About Gradient-Based Solvers](#). Note that this can be too strict, in particular if the forward multiphysics model is not solved accurately enough. See [About Optimality Tolerances](#).

Maximum Number of Model Evaluations

Specify the **Maximum number of model evaluations**, which defaults to 1000. 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 model 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 can 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.

Condition-based Final Time

For time-dependent optimization, the objective is evaluated at the final time, but it is possible to let the optimization solver control this time by specifying a **Stop expression**.

OPTIMIZATION SOLVER

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

- [Method](#)
- [Solution](#)
- [Objective Contributions](#)
- [Gradient Method](#)
- [Gradient Method Parameters for Time-Dependent Problems](#)
- [Numeric Gradient Method Parameters](#)
- [Store Functional Sensitivity](#)
- [IPOPT-Specific Settings](#)
- [Settings specific to MMA and GCMMA](#)
- [Levenberg–Marquardt-Specific Settings](#)

Method

The four available choices are **IPOPT** (the default), **GCMMA**, **MMA**, and **Levenberg–Marquardt**. The Levenberg–Marquardt method can only be used for problems of least-squares type without constraints, and it is not supported for eigenvalue problems. IPOPT and MMA can solve any type of optimization problem. See [About Gradient-Based Solvers](#).

Solution

This setting controls the behavior when the solver node under the Optimization solver node returns a solution containing more than one solution vector (for example, a frequency response). If the gradient step is an eigenvalue study step, the **Auto** setting correspond to **Use first** — namely, only the first eigenvalue and eigenfunction are used. The IPOPT and Levenberg–Marquardt solvers, except for the eigenvalue study step, only support the **Auto** setting, meaning in practice the sum over frequencies and parameters or the last time step. For MMA, the options are as for the derivative-free solvers: **Auto**, **Use first**, **Use last**, **Sum of objectives**, **Minimum of objectives**, and **Maximum of objectives**. The last two settings make the MMA algorithms handle maximin and minimax problems efficiently.

Objective Contributions

When IPOPT or MMA 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 not deactivated in an optimization study step are used as objective function.

By selecting **Manual**, you can enter an expression that is used as the objective function in the **Objective expression** field. The expression `all_obj_contrib` represents the sum of all objective contributions not deactivated in a controlling optimization study step. Hence, this expressions leads to the same optimization problem as selecting **All**. Note, however, that MMA treats least-squares objective contributions in a more efficient way when **All** is selected.

When you use Levenberg–Marquardt, the objective function is always the sum of all active least-squares objective contributions present in the model.

Gradient Method

IPOPT, GCMMA/MMA, and Levenberg–Marquardt are gradient-based methods. The gradient can be computed according to the choices **Automatic, analytic** (the default); **Forward**; **Adjoint**; **Forward numeric**; and **Numeric**. Moreover, only the **Forward** and **Numeric** methods are supported for eigenvalue solvers. When **Automatic, analytic** is chosen, either the *adjoint method* or the *forward method* is used to compute the gradient analytically. The automatically selected method leads to fewer linear solves and thus a lower computational time. The number of linear solves depends on the number of control variables, number of objective functions, and the number of constraints.

It is also possible to explicitly choose to use either the adjoint or forward method using the corresponding alternatives from the menu. With the option **Forward numeric**, a

semianalytic approach is available where the gradient of the PDE residual with respect to control variables is computed by numerical perturbation and then substituted into the forward analytic method. When **Numeric** is chosen, finite differences are used to compute the gradient numerically.



The computational time required to calculate the gradient depends on the method: The cost of the numeric and forward methods scales with the number of controls, and may therefore be time consuming, while the cost of the adjoint method does not depend on the number of controls.

Gradient Method Parameters for Time-Dependent Problems

For time-dependent problems, all analytic gradient methods have options to adjust the default integration tolerances for the sensitivity solver. A separate section titled [Transient Adjoint](#) appears, when using the **Adjoint** gradient method.

For the **Forward** and **Forward numeric** gradient methods a **Forward sensitivity rtol factor** can be specified. This factor multiplied by the forward problem relative tolerance to calculate the relative tolerance for the sensitivity solver. You can also specify a **Forward sensitivity scaled atol**, which is a global absolute tolerance that is scaled with the initial conditions. The absolute tolerance for the sensitivity solution is updated if scaled absolute tolerances are updated for the forward problem.

Numeric Gradient Method Parameters

When the **Numeric** gradient method is selected, you can further specify a **Difference interval** (default $1.5E-8$). This is the relative magnitude of the numerical perturbations to use for all numeric differentiation.

You can choose the **Gradient approximation order** explicitly. 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.

Store Functional Sensitivity

The sensitivity of the objective function is by default stored in the solution object such that it can be postprocessed after the solver has completed. To save memory by discarding this information, change **Store functional sensitivity** to **Off**. If you instead choose **On for results while solving**, sensitivity information is also computed continuously during solution and made available for probing and plotting while solving. This is the most expensive option.

IPOPT-Specific Settings

Absolute tolerance on the dual infeasibility, see [The IPOPT Solver](#). Successful termination requires that the max-norm of the dual infeasibility is less than the **Dual infeasibility absolute tolerance** (the default value is 1).

Absolute tolerance on the constraint violation, see [The IPOPT Solver](#). Successful termination requires that the max-norm of the constraint violation is less than the **Constraint violation absolute tolerance** (The default value is 0.1).

Absolute tolerance on the complementarity, see [The IPOPT Solver](#). Successful termination requires that the max-norm of the complementarity is less than the **Complementary conditions absolute tolerance** (the default value is 0.1).

You can choose a **Linear Solver** for the step computations. The options are **MUMPS** and **PARDISO**. The default value is **MUMPS**.

If **MUMPS** is selected, you can choose **Percentage increase in the estimated working space**. The default value is 1000. A small value can reduce memory requirements at the expense of computational time.

Settings specific to MMA and GCMMA

The **Move limits** option makes it possible to bound the maximum absolute change for any (scaled) control variable between two outer iterations. This is particularly relevant, when MMA is used, because this does not have an inner loop to ensure improvement of the objective and satisfaction of constraints.

By default, the GCMMA and MMA solvers continue to iterate until the relative change in any control variable is less than the optimality tolerance. If the **Maximum outer iterations** option is enabled, the solver stops either on the tolerance criterion or when the number of iterations is more than the maximum specified.

The GCMMA solver has an inner loop which ensures that each new outer iteration point is feasible and improves on the objective function value. By default, the **Maximum number of inner iterations per outer iteration** is 10. When the maximum number of inner iterations is reached, the solver continues with the next outer iteration.

The **Internal tolerance factor** is multiplied by the optimality tolerance to provide an internal tolerance number that is used in the GCMMA/MMA algorithm to determine if the approximations done in the inner loop are feasible and improve on the objective function value. The default is 0.1. Decrease the factor to get stricter tolerances and a more conservative solver behavior.

Both MMA and GCMMA penalize violations of the constraints by a number that is calculated as the specified **Constraint penalty factor** times $1e-4$ divided by the optimality tolerance. Increasing this factor for a given optimality tolerance forces the solver to better respect constraints, while relatively decreasing the influence of the objective function.

Levenberg–Marquardt-Specific Settings

By default the solver terminates on the tolerance, but it is possible to terminate prematurely (without triggering a warning) by using setting the **Maximum number of outer iterations**.

The Levenberg–Marquardt method controls the step length and direction through a positive scalar regularization parameter. 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. See [The Levenberg–Marquardt Solver](#).

The Levenberg–Marquardt method controls this penalty factor internally and tries to have as small penalty as possible in order to approach second-order Newton convergence. Therefore, a small value of the **Initial damping factor** means that the solver tries to be aggressive initially, while a large value means that the solver is more cautious.

By default, the Levenberg–Marquardt solver terminates on the controls or the angle between the defect and the Jacobian, but if **Terminate also for defect reduction** is enabled, the solver will also terminate, if the sum of squares has been reduced by a factor equal to the product of the optimality tolerance and the **Defect reduction tolerance factor**.

TRANSIENT ADJOINT

This section only appears, if the **Adjoint** gradient method is chosen in the [Optimization Solver](#) section. You can choose between two solver types:

- **Time discrete** (the default) forces the adjoint time integration to use the same time steps and the accuracy is comparable to stationary sensitivity, but there can still be a need for recomputation of the forward solution.
- **Time continuous** allows the forward and adjoint time integration to use different time steps.

Unless all forward solutions are stored in memory, the transient adjoint will involve recomputation from the forward solutions that are saved — the checkpoints. This can

be avoided by setting **Out-of-core** for the **Forward solution handling**. The number of checkpoints is determined the setting for **Forward solutions stored between checkpoints**:

- **Automatic** (the default) the number of checkpoints is controlled by specifying the **Maximum in-core memory (KB)**. Note that specifying a large value can result in poor performance, unless manual time stepping is used.

Manual the number of checkpoints is controlled by specifying the maximum number of **Forward solutions between checkpoints** and in practice this means that the number of checkpoints is identical to the number of steps taken by the solver divided by the number given in this setting. The maximum number of stored forward solutions at anytime will then be the number of checkpoints plus the value given in the setting. Recomputation of the forward solution can thus be avoided by setting a value larger than the number of time steps used for the forward problem, but this also maximizes the memory consumption, see [Figure 5-1](#). Conversely, the memory consumption will be minimized if the number of checkpoints is set to the square root of the number of time steps used for the forward problem.

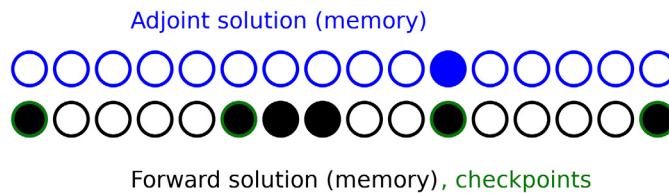


Figure 5-1: The memory consumption is illustrated during the checkpointing process of the adjoint solution with time going from left to right and each circle representing a time step.

Time continuous

When **Time continuous** is chosen, the backward time stepping can be chosen as

- **Automatic** (the default) try to determine the optimal time stepping automatically based on the tolerance.
- **From forward** uses the times steps from the forward problem, but can swap the order of steps by choosing **Use step-length from forward stepping**. You can also choose the **BDF order**.
- **Manual** allows complete control using the **BDF order**, **Initial step fraction**, **Initial step growth rate**, and the **Time step**. Only positive numbers are allowed, except for the **Time step**, which only accepts negative numbers.

The **Tolerances** of the backward time stepping is determined automatically by default, but it is also possible to manually specify the **Adjoint rtol factor** and **Adjoint scaled atol** factors, which control the accuracy of the adjoint solution, similarly to the corresponding Forward sensitivity factors. In addition an **Adjoint quadrature rtol factor** and an **Adjoint quadrature atol** can be given. These settings control the relative and absolute accuracy of time integrals (or quadratures) used to calculate objective function gradients. Note that the absolute tolerance is unscaled.

OUTPUT

The **Keep solutions** setting is synchronized from the optimization study step. It is described in [The General Optimization Study](#), see [Keep Solutions \(Gradient Based\)](#).

RESULTS WHILE SOLVING

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

ADVANCED

Use the **Compensate for nojac terms** list to specify whether to try to assemble the complete Jacobian if an incomplete Jacobian has been detected. Select:

- **Automatic** (the default) to try to assemble the complete Jacobian if an incomplete Jacobian has been detected. If the assembly of the complete Jacobian fails or in the case of nonconvergence, a warning is written and the incomplete Jacobian is used in the sensitivity analysis for stationary problems. For time-dependent problems, an error is returned.
- **On** to try to assemble the complete Jacobian if an incomplete Jacobian has been detected. If the assembly of the complete Jacobian fails or in the case of nonconvergence, an error is returned.
- **Off** to use the incomplete Jacobian in the sensitivity analysis.

From the **Adjoint solution** choose **Initialize with zero solution** (the default) or **Initialize with forward solution**. The latter option can reduce the computational time, if the problem is self-adjoint and an iterative solver is used. This typically happens in linear structural mechanics when the total elastic strain energy variable (WS_{tot}) is used as objective. The option only affects the adjoint problem of stationary solvers.

See [Theory for Stationary Sensitivity Analysis](#) for details about the algorithm.

CONSTANTS

In this section you can define constants that can be used as temporary constants in the solver. You can use the constants in the model or to define values for internal solver parameters. Click the **Add** () button to add a constant and then define its name in the **Constant name** column and its value (a numerical value or parameter expression) in the **Constant value** column. By default, any defined parameters are first added as the constant names, but you can change the names to define other constants. Click **Delete** () to remove the selected constant from the list.

LOG

The **Log** displays the information about the progress of the solver. See [The Optimization Solver Log](#).

Cluster Computing

Optimization always involves several iterations, and it is normal that the forward problem has to be solved 100–1000 times before a good solution is found. Therefore it is common that computational cost becomes important. To address this cost, you can use some of these methods:

- Reduce the mesh size (while still capturing the relevant physical effects)
- Exploit symmetry
- Eliminate couplings and physical effects that are of lesser importance
- Make sure that the solver setup is efficient

The computational cost might still be higher after following these best practices. Cluster computing can be used to reduce the cost further. There are three options (see below), and they cannot be combined. The first option is used by default. The settings for the other options are accessible via the **Batch and Cluster** checkbox in the **Show More Options** dialog.

Distributing the Forward Problem

The forward problem is always run in distributed mode on clusters, unless you check any of the options listed below. For a list of algorithms that support distribution, see the section [Benefits of Running COMSOL in a Distributed Mode](#) in the *COMSOL Multiphysics Reference Manual*.

Parametric Sweep and Gradient Free Optimization

Both the Nelder–Mead and EGO solvers perform various internal operations that involve evaluating the objective function for multiple sets of control variables. In the Nelder–Mead method, this occurs as part of the simplex adjustments, while in EGO, the objective function is evaluated for initial sets of sampled control variables and potentially for multiple new sets of control variables selected simultaneously from the acquisition function. In both methods, these evaluations are independent of each other, allowing them to be computed in parallel. You can enable this distributed approach in the optimization study step using the **Distribute parametric sweep** checkbox.

Similarly, you might be using a **Parametric Sweep** to solve several independent optimization problems in order to generate a Pareto optimal curve. You can use the same **Distribute parametric sweep** checkbox to ensure that the optimizations are carried out in parallel. Note that the optimizations have to be independent, and therefore you cannot combine this with the **Reuse solution from previous step** checkbox.

Load Cases, Multiple Frequencies, and Other Inner Sweeps

Some problems have several load cases, multiple frequencies, or other inner sweeps that can be computed in parallel. This requires that you select the **Distribute parametric solver** checkbox in the **Stationary**, **Frequency Domain**, or other study step used for the inner sweep. This approach can be used with gradient-based optimization, and it tends to be more efficient than distributing each individual forward problem to different cluster nodes.

Advanced Solver Properties

This section provides detailed explanations of some of the properties that control the behavior of the MMA optimization solver in the Optimization Module.

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 always take their default values. Modifying the value of those properties requires that the value is changed using LiveLink™ for MATLAB® or by running a compiled COMSOL API history file.

	In the following sections, ϵ represents the machine precision (available as <code>eps</code> in MATLAB) and is approximately equal to $2.2 \cdot 10^{-16}$.
	For a list of all available optimization solver properties see Optimization in the <i>COMSOL Multiphysics Programming Reference Manual</i> .

MMA Solver Properties

In addition to the settings available in the COMSOL Desktop, MMA when called through the API allows explicit tuning of the approximating subproblems solved in each inner iteration. In particular, update rules for the moving asymptotes can be modified. It is also possible to switch off the automatic transformation performed on least-squares, minimax and maximin problems, as well as disable the globally convergent extension of the MMA method.

For a list of all available options see [Optimization](#) in the *COMSOL Multiphysics Programming Reference Manual*.

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 Reference Manual*. For references to more information about a term, see the index.

Glossary of Terms

adjoint method The *adjoint method* for sensitivity analysis is based on exploiting the adjoint identity given an objective functional to derive and solve a set of adjoint equations. The adjoint solution is then used to compute the functional sensitivity with respect to sensitivity parameters.

BFGS (Broyden–Fletcher–Goldfarb–Shanno) A specific family of optimization algorithms where updates are made to approximate the inverse of the Hessian matrix.

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 constraint A constraint which can be evaluated before any multiphysics simulation has been performed. A design constrain can be expressed explicitly in the control variables, without involving the multiphysics problem solution.

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 can 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 can also be implicitly limited by the existence of a solution to a multiphysics problem.

forward method The *forward method* for sensitivity analysis is based on solving the equations obtained by applying the chain rule of differentiation, with respect to sensitivity parameters, to the original DAEs.

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 parametric 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 parametric model matches measured data. Replacing the parameters with control variables leads to an optimization problem, which can 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.

performance constraint A constraint involving the multiphysics simulation result. Performance constraints in general have the same structure as the objective function, and are as expensive to evaluate.

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.

superbasic variable A variable is *superbasic* if it is not currently at one of its bounds.

I n d e x

- A**
 - adjoint sensitivity 26, 31
 - Application Libraries window 13
 - application library examples
 - sensitivity 85
- B**
 - BOBYQA solver 132, 134
 - bounds 20
 - control variable 20
- C**
 - classical optimization 17
 - COBYLA solver 132, 134
 - confidence intervals 121
 - constraints 16
 - bounds 20
 - design 19
 - explicit 19
 - expression 51
 - implicit 19
 - integral inequality 20, 51
 - performance 19
 - point sum inequality 51
 - pointwise inequality 20, 51
 - contribution
 - global 18, 30, 160
 - integral 19, 30, 160
 - probe 18, 160
 - control variable 17, 45, 160
 - control variable bounds 20
 - control variable bounds (node) 53
 - control variable field (node) 52, 80, 86
 - control variables 16, 56
 - convergence 140
 - coordinate column (node) 49
 - cost function 161
 - curve fitting 10
- D**
 - decision variables 160
 - density model (node) 58
 - derivative-free optimization 117
 - design constraints 19
 - design constraints, enforcing 119
 - design problems 10
 - design variables 160
 - displaying
 - sensitivity analysis 32
 - documentation 12
- E**
 - emailing COMSOL 14
 - equality constraints 52
 - evaluations, model 148
 - explicit constraints 19
 - expression, objective 50, 54
- F**
 - feasible set 16–17
 - forward methods 28
 - forward sensitivity 25, 27, 30
 - free shape boundary (node) 69, 71
 - free shape domain (node) 67
 - function
 - cost 161
 - goal 161
 - functions
 - objective 11, 16–17
- G**
 - GCMMA solver 137
 - general constraints 143
 - geometrical sensitivity 33
 - global
 - contribution 18, 30, 160
 - control variable (node) 56
 - inequality constraint (node) 55
 - objective (node) 54
 - global control variables (node) 87
 - global inequality constraint 20
 - global least-squares objective (node) 54
 - global method 132

- global objective (node) 87
- global optimization 45
- globalleastquaresobjective (node) 77–79
- goal function 161
- gradient-based optimization 149
- gradient-free optimization 117
- I**
 - ignored column (node) 49
 - implicit constraints 19
 - in 144
 - incomplete Jacobian, in sensitivity 24
 - inequality constraints
 - global 55
 - integral 51
 - point sum 51
 - pointwise 51
 - integral
 - contribution 19, 30
 - inequality constraints 20
 - integral contribution 160
 - integral inequality constraints (node) 51
 - integral objective (node) 50, 85
 - internet resources 12
 - inverse problem 161
 - inverse problems 10
- K** knowledge base, COMSOL 14
- L**
 - least-squares objective (node) 46
 - Levenberg-Marquardt method 149
 - Levenberg-Marquardt solver 133, 138
 - linear shape domain (node) 68
 - LiveLink for MATLAB 158
 - load from excel file (button) 48, 56
 - local method 132
 - lower bound 51
- M**
 - mathematics branch 44
 - minimization or maximization 94, 105, 111, 123
 - mirror symmetry (node) 64
- MMA solver 133, 137
- model inputs
 - optimizing 10
- MPH files 13
- N** Nelder-Mead solver 132–133
- O**
 - objective
 - expression 50, 54
 - functions 11, 16–17
 - global 54
 - integral 50
 - point sum 50
 - objective function 45
 - complex-valued 33
 - definition 22
 - specification of 29
 - optimality tolerance 93, 101, 104, 110, 117, 140, 147
 - optimization
 - gradient-based 149
 - problem formulations 16
 - problems 10
 - optimization interface 45
 - theory 16
 - optimization problem 161
 - optimization solver 147
 - optimization study node 115
 - optimization variable 160
- P**
 - parameter column (node) 49
 - parameter estimation 10
 - parameter estimation (node) 98, 129
 - PDE-constrained optimization problem 17
 - performance constraints 19
 - point sum inequality constraints (node) 51
 - point sum objective (node) 50
 - Pointwise constraints 143

- pointwise inequality constraint (node) 51
- pointwise inequality constraints 20
- polynomial boundary (node) 72
- polynomial shell (node) 73
- prescribed density (node) 63
- prescribed density boundary (node) 64
- prescribed material (node) 63
- prescribed material boundary (node) 63
- prescribed void (node) 62–63
- principle of virtual work 33
- probe contribution 18, 30, 160
- probe objective 18, 30
- probe objective (node) 50, 86
- Q**
 - quadrature 50
 - quantity of interest 161
- S**
 - save to excel file (button) 48, 56
 - sector symmetry (node) 65
 - sensitivity analysis 22
 - sensitivity interface 59, 84
 - sequential quadratic programming 135
 - shape mirror (node) 75
 - shape optimization study node 91, 109
 - simple bounds 17, 143
 - SNOPT 135
 - solution variable 19, 45
 - study reference node 115
 - symmetry/roller (node) 74–75
- T**
 - technical support, COMSOL 14
 - theory
 - optimization 16
 - time column (node) 49
 - tolerance
 - optimality 93, 101, 104, 110, 117, 147
 - value 140
 - topology optimization study node 103
 - trust region 134
- U**
 - upper bound 51
- V**
 - value column (node) 48
 - variables
 - control 16–17
 - decision 160
 - global 56
 - sensitivity solvers 27
 - solution 19
- W**
 - websites, COMSOL 14

