



LiveLink™ *for* Excel®

User's Guide

LiveLink™ *for* Excel® User's Guide

© 2012–2023 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/comsol-license-agreement) 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. Microsoft, Excel, Visual Basic and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. 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 or the above non-COMSOL trademark owners. For a list of such trademark owners, see www.comsol.com/trademarks.

Version: COMSOL 6.2

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. Other useful links include:

- 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

Part number: CM023401

C o n t e n t s

Chapter 1: Introduction

About This Product	8
Interacting with a Model from a Worksheet	8
Exporting Data to a Material Library	10
Sharing a Model Between Excel® and the COMSOL Desktop®	11
Loading and Saving Workbook Files in the COMSOL Desktop®	11
Automating Using Visual Basic® for Application (VBA)	11
Help and Documentation	13

Chapter 2: Working With Models From Worksheets

Overview of the COMSOL Tab	18
Main	18
Graphics	19
Definitions	21
Study	23
Numerical Results	24
Report.	31
Material Export	33
Help	33
Working With Models From Worksheets	34
Accessing the Model Definitions	34
Computing the Solution	36
Running a Model in Sweep	36
Evaluating the Results	38
Displaying the Results	42
Updating Data in Cells Linked to the Model.	43

The COMSOL Backstage View	45
Opening a Model MPH-File	45
Saving a Model MPH-File	47
Connecting Excel [®] to a COMSOL Server	48
Disconnecting Excel [®] from a COMSOL Server	49
Managing Model Loaded on a COMSOL Server	49
Connecting the COMSOL Desktop [®] with the COMSOL Multiphysics Server	50
Run Application	51
Preference Settings	52
 Exporting Material Data	 54
The Material Export Settings Window	54
The Cell Comment	59
Saving the Material Library	59
Exporting the Material Library in Batch	60

Chapter 3: Loading and Saving Workbook Files

Importing and Exporting Model Definitions	64
Support for Excel [®] Files	64
Importing Data from a Workbook	64
Exporting Data to a Workbook	65
Supported Formats	67
Exchange Workbook Using the COMSOL [®] API	69
Exchange Workbook in the Application Builder	70

Chapter 4: The COMSOL API for VBA

Introduction	74
Support for VBA	74
Introductory Example	75
 Before You Start	 78
Enabling the Developer Tab in Excel [®]	78

Adding ComsolCom Reference to the VBA Project. 78

Registering the Component for the COMSOL API 79

Saving Excel Workbooks Containing Macros 79

Working with Models Using VBA 80

Saving the Model File for VBA 80

The Model Object and the ModelUtil Object 81

The ComsolUtil and RibbonUtil Objects 81

ComsolCom Version Control 82

Declaring Utility Objects with Dim 83

Error Handling. 83

Starting and Connecting to a COMSOL Multiphysics Server 85

Updating and Solving a Model Using Excel[®] Macros 87

Objects and Methods 97

The COMSOL API 97

Renamed Methods 97

Utility Methods 100

Commands Grouped by Function 101

Methods in the ComsolUtil Class. 104

Methods in the RibbonUtil Class 106

Introduction

This guide introduces you to LiveLink™ *for* Excel®, which extends your COMSOL modeling environment with an interface between COMSOL Multiphysics® and Excel®.

In this chapter:

- [About This Product](#)
- [Help and Documentation](#)

About This Product

LiveLink™ for Excel® extends your modeling capabilities by running COMSOL Multiphysics® simulations from an Excel workbook. Model definitions and results can easily be synchronized between your workbook and the simulation. In addition, LiveLink™ for Excel® adds the capability to create a COMSOL material library from data stored in a worksheet, and it enables support for saving and loading Excel® files for parameter and variable lists in the COMSOL Desktop®.

Read this section for an overview of:

- [Interacting with a Model from a Worksheet](#)
- [Exporting Data to a Material Library](#)
- [Sharing a Model Between Excel® and the COMSOL Desktop®](#)
- [Loading and Saving Workbook Files in the COMSOL Desktop®](#)
- [Automating Using Visual Basic® for Application \(VBA\)](#)



Connecting Excel to a COMSOL Multiphysics Server is only supported on Windows®.

Interacting with a Model from a Worksheet

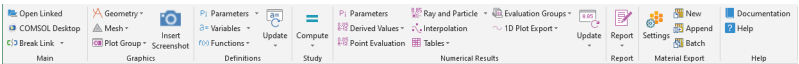


Figure 1-1: The COMSOL ribbon tab.

Using the tools on the COMSOL tab in the Excel® user interface (Figure 1-1) you can extract or modify model definitions, extract simulation results, and recompute a simulation. This allows you, for instance, to implement a simplified interface to a model (Figure 1-2) where you can access only the most important simulation parameters, and results from Excel.

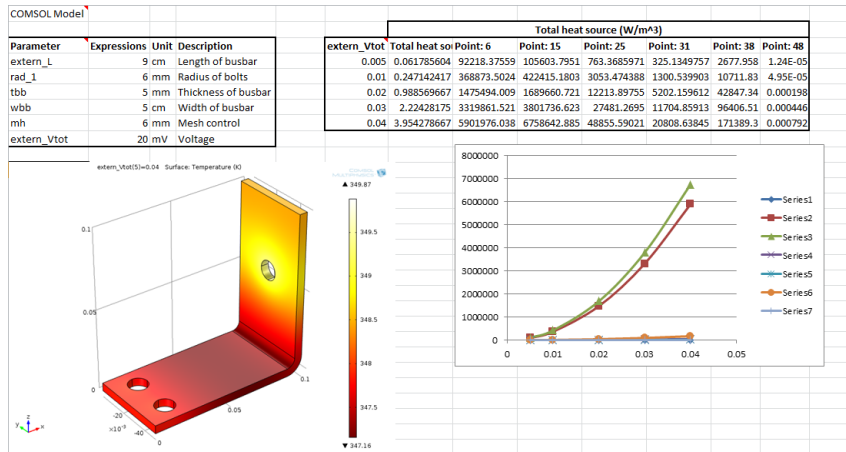


Figure 1-2: Overview of a worksheet linked to a COMSOL model.

While you work with COMSOL models you can still interact with 3D graphics in a separate dedicated window. You can insert COMSOL graphics into your worksheet as graphics images. This makes it possible to share your workbooks with people who do not have access to COMSOL Multiphysics.

As soon as you open a model in Excel, a COMSOL Multiphysics Server starts in graphics mode where the model is loaded. The data transfer between Excel and the COMSOL Multiphysics Server is performed using a TCP/IP communication protocol.

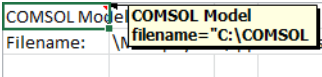


The first time you start a COMSOL Multiphysics Server you need to enter a user name and password. Once this information is entered, the client/server communication is established. The information is stored in the user preferences, so that subsequent starts do not require it to be entered again.

LINKING A WORKBOOK TO A MODEL

When importing information from a model to a workbook in Excel, there are links placed in certain cells to ensure that data can be kept associated to the appropriate nodes in the COMSOL model. These links are represented in the Excel worksheet by

a comment on the cell, visible as a red mark at the top right cell corner. Hold the cursor over the cell comment to get a short description of the link.



Removing an automatically generated comment breaks the link between that cell range and the associated model feature.

Exporting Data to a Material Library

With LiveLink™ you can create a material library accessible by COMSOL models from material data stored in an Excel file. Using an interface (see [Figure 1-3](#)) you can configure the export such that COMSOL recognizes the material properties. The Material Export supports Excel sheet containing constant or field dependent material property.

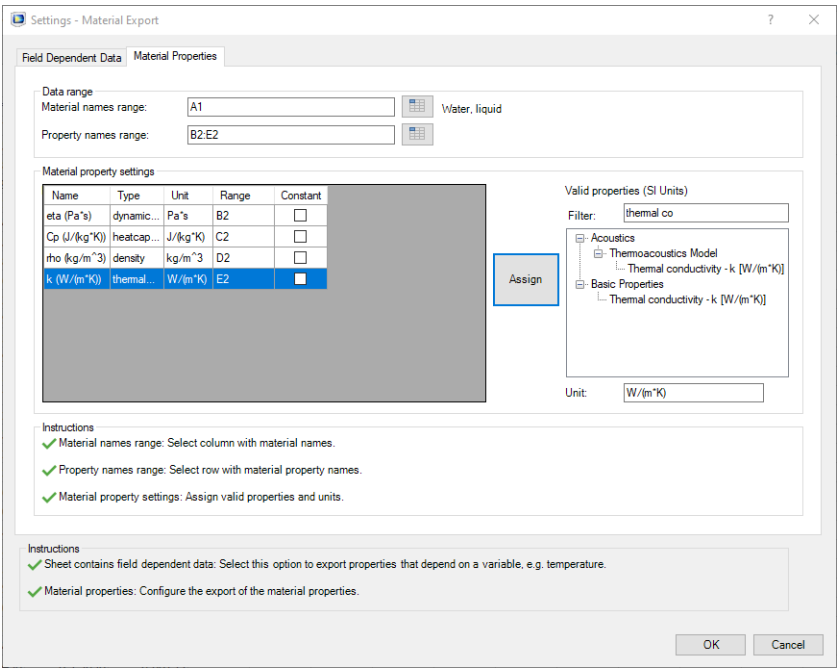


Figure 1-3: The Material export Settings dialog box.

Sharing a Model Between Excel® and the COMSOL Desktop®

You can share a model that is open in Excel with COMSOL Desktop by connecting COMSOL Desktop to the same COMSOL Multiphysics Server as Excel is connected to. This makes it possible to access the model settings simultaneously from Excel and from COMSOL desktop.

LiveLink™ for Excel® connects Excel with a COMSOL Multiphysics Server, started in graphics mode. This is called a graphics server. The graphics server does not support multiple clients connected at the same time. To share a model between the COMSOL Desktop and Excel you need to start a regular COMSOL Multiphysics Server when opening the model in Excel. See [Preference Settings](#) to set the connection between Excel and a COMSOL Multiphysics Server without graphics.

Loading and Saving Workbook Files in the COMSOL Desktop®

LiveLink™ for Excel® adds support to the COMSOL Desktop for the import and export files of the Excel workbook format (.xlsx). Use this functionality, for example, when saving data from a table to file. See [Figure 1-4](#).

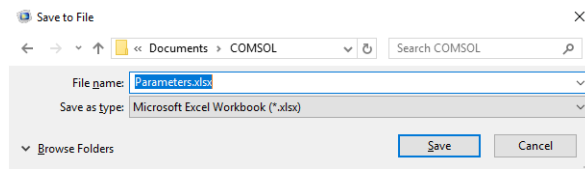


Figure 1-4: The Save to File dialog box.



Loading and saving workbook files is supported on all platforms even if Excel is not installed.

Automating Using Visual Basic® for Application (VBA)

Use VBA to control the content of Excel workbooks, create user interfaces, and perform advanced calculations on worksheet linked to a COMSOL model. Using

LiveLink™ *for* Excel® also provides the possibility to use the COMSOL Application Programming Interface (API) to implement a model.

COMSOL Model

Filename: \Multiphysics\applications\LiveLink_for_Excel\Tutor

Parameter	Expressions	Unit	Description	Value	Unit
extern_L	9	cm	Length of busbar	0.09	m
extern_Vtot	20	mV	Voltage	0.02	V

Name	Expressions
wbb	5.00E-02 0.1 0.15
extern_Vtot	5[mV] 10[mV] 20[mV] 30[mV] 40[mV]

Compute



You can save model files in the VBA format directly from the COMSOL Desktop.

Help and Documentation

A number of internet 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 other modules (as long as you have a license), application examples, and documentation sets.

THE DOCUMENTATION AND ONLINE HELP



The *COMSOL Multiphysics Reference Manual* describes all core physics interfaces and functionality included with the COMSOL Multiphysics license. This book also has instructions about 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 many of the features on the GUI. 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.

<div>Mac</div> <div>Linux</div>	<p>To open the Help window:</p> <ul style="list-style-type: none"> • 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.
---------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Opening the Documentation Window

<div>Win</div>	<p>To open the Documentation window:</p> <ul style="list-style-type: none"> • Press Ctrl+F1. • From the File menu select Help>Documentation ().
----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<div>Mac</div> <div>Linux</div>	<p>To open the Documentation window:</p> <ul style="list-style-type: none"> • Press Ctrl+F1. • In the main toolbar, click the Documentation () button. • From the main menu, select Help>Documentation.
---------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

THE APPLICATION LIBRARIES WINDOW






Each application includes documentation with the theoretical background and step-by-step instructions to create a model application. The applications are available in COMSOL as MPH-files that you can open for further investigation. You can use the step-by-step instructions and the actual applications as a template for your own modeling and applications. In most models, SI units are used to describe the relevant properties, parameters, and dimensions in most examples, 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 application and its properties, including options to open it or a PDF document.

	<p>The Application Libraries Window in the <i>COMSOL Multiphysics Reference Manual</i>.</p>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------

Opening the Application Libraries Window

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

	<ul style="list-style-type: none">• In the Home toolbar, click Windows and select Application Libraries. When the toolbar is compressed, you sometimes find it under Layout>Windows.• From the File menu select Application Libraries. <p>To include the latest versions of model examples, from the File>Help menu select () Update COMSOL Application Libraries.</p>
	<p>From the File or Windows menu select Application Libraries.</p> <p>To include the latest versions of model examples, from the Help menu select () Update COMSOL Application Libraries.</p>
	

CONTACTING COMSOL BY EMAIL

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

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

COMSOL WEBSITES

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 Video Gallery	www.comsol.com/videos
Support Knowledge Base	www.comsol.com/support/knowledgebase

Working With Models From Worksheets

The **COMSOL 6.2** ribbon tab inside the Excel[®] user interface is where you can access, modify, solve, and extract results from a simulation. With the provided functionality you can also easily create a COMSOL material library from data stored in Excel[®] worksheets.

In this chapter:

- [Overview of the COMSOL Tab](#)
- [Working With Models From Worksheets](#)
- [The COMSOL Backstage View](#)
- [Exporting Material Data](#)

Overview of the COMSOL Tab

Select the **COMSOL 6.2** tab in the Excel[®] user interface to access LiveLink[™] functionality for interacting with a COMSOL model from an Excel worksheet.



From here you can insert model definitions and results into a worksheet, you can also update model definitions with data from a worksheet, and compute the solution. A separate Graphics window provides a view of the geometry, the mesh, or the solution, and you can insert screenshots into the worksheet. From the COMSOL tab you can also easily convert material data stored in Excel worksheets to material data stored in a COMSOL material library for easy access from models. The functionality provided by the COMSOL tab is also available using Visual Basic[®] for Applications (VBA) as specific commands that can be run in a VBA script for automation.


The tools on the COMSOL tab are organized into the following groups:

- **Main**, where you can open models, and break the link between the worksheet and a model.
- **Graphics**, where you can view the geometry, mesh, and results plots, and insert screenshots into worksheets.
- **Definitions**, where you can retrieve and update model definitions.
- **Study**, where you can compute and update a solution, and access the settings for parametric sweeps.
- **Numerical Results**, where you can evaluate and extract data to the worksheet.
- **Report**, where you can generate a report for the model.
- **Material Export**, where you can create a material library.
- **Help**, where you can access the documentation.


Main

Use the buttons in the **Main** group to open the model linked to the Excel workbook, open the current model in the COMSOL Desktop, and to break the link between cells and the model.


OPEN LINKED

Click **Open Linked** () to open the model linked to the current Excel workbook. The path to the model is saved as a cell comment in one of the worksheets of the workbook. Note that a workbook can only be linked to one model at a time.

COMSOL DESKTOP

Click **COMSOL Desktop** () to open with the COMSOL Desktop the model that is currently active on the COMSOL Multiphysics Server. This enables working with a model both from the Excel workbook and the COMSOL Desktop at the same time, and it is especially useful during the initial setup of worksheets that interface a model.


BREAK LINK

Click **Break Link** () to remove the comment added by the LiveLink™ interface to link the selected cell range to a model. The data in the cells can no longer be updated after the link is removed.



If you remove the comment on a cell that contains a material name or property configured for material export, the material or property is not exported to the material library, and the settings are lost.

See [BreakLink](#) for information about the corresponding VBA command.


Click **Break Link > Break All Links** () to remove all comments that link cell ranges from the current worksheet to a model. Note that this action also removes the comment with the path to the linked model, if it is included on the worksheet.

See [BreakAllLinks](#) for information about the corresponding VBA command.

Graphics


Using the buttons in the **Graphics** group you can display the model geometry, mesh, and plot groups in the COMSOL Multiphysics Server graphics window.

GEOMETRY

Click **Geometry** () to display the model geometry in the COMSOL Multiphysics Server graphics window. In case the model contains multiple geometry nodes, select the geometry to display from the menu. To change the level of detail for the displayed geometry, see [Preference Settings](#).


See [InsertGeometryGraphics](#) for information about the corresponding VBA command.

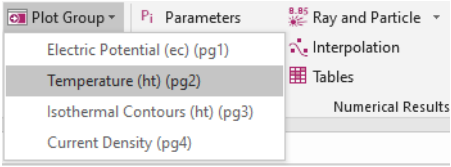
MESH

Click **Mesh** () to display the mesh in the COMSOL Multiphysics Server graphics window. In case the model contains multiple mesh nodes, select the mesh node to display from the menu.


See [InsertMeshGraphics](#) for information about the corresponding VBA command.

PLOT GROUP


Click **Plot Group** () and select a plot group to display it in the COMSOL Multiphysics Server graphics window. Only plot groups that are defined in the open model can be displayed.




PLOT DATA

To select the parameter value, time, or eigenfrequency for which to display results for in the currently displayed plot group click **Plot Data** ()

INSERT SCREENSHOT

Click **Insert Screenshot** () to generate and insert into the worksheet a screenshot image of the currently displayed contents of the COMSOL Multiphysics Server Graphics window. The image can be exported from the graphics or directly taken as a screenshot. You can specify how you want the image and its size from the **Preferences** window.




See [Preference Settings](#) to access the **Preferences** window.


See [InsertGraphics](#) for information about the corresponding VBA command.

Definitions

The **Definitions** group includes functionality for inserting model definitions, for example, parameters, variables, and functions, into the current worksheet. When inserting the data, a link, which keeps the data associated to the model, is also added in a form of a comment on a cell. You can edit the data in linked cells, and update the model definitions from data from the worksheet.

PARAMETERS


Click **Parameters** () to insert into the worksheet all global parameters from the open model. Parameters are added to the worksheet starting at the currently active cell, where a comment is also inserted linking the cell range to the model and the parameters nodes.


Click **Parameters>Parameters Nodes** () to only insert into the worksheet parameters listed in a selected parameters node. This way you can easily limit the parameters you would like to expose in the worksheet.

Once inserted from the model, you can edit the expression, unit, and description of a parameter in the worksheet. To add a new parameter, enter it on the row below the last parameter in the cell range. Any changes to the parameters in the worksheet are detected and transferred automatically to the model. To manually update the model with data in the worksheet, see [Update](#).

See [Parameters](#) for information about the corresponding VBA command.

VARIABLES


Click **Variables** () to insert into the worksheet all variables from the open model. Variables are added to the worksheet starting at the currently active cell, where a comment is also inserted linking the cell range to the model and the variables nodes.

Click **Variables>Variables Nodes** () to import only variables from a selected variables node in the model.

Once inserted from the model, you can edit the expression and description of variables in the worksheet. The model is automatically updated with changes to variables in the worksheet. To manually update the model with data in the worksheet, see [Update](#).

See [Variables](#) for information about the corresponding VBA command.

FUNCTIONS

Click **Functions** () to insert function definitions from the open model. Functions are added to the worksheet starting at the currently active cell, where a comment is also inserted linking the cell range to the model and the function nodes.



It is only possible to insert functions of the following type: analytic, interpolation, step, and random.


In the worksheet, you can edit the inserted function definitions. After you edit a cell in a linked cell range, by default an update of the model is performed automatically. To manually update the model with the modifications in the worksheet, see [Update](#).

See [Functions](#) for information about the corresponding VBA command.


UPDATE

By default, the update of parameters, variables, and functions in the model is performed automatically when a cell is edited in a linked cell range. In case you disable the automatic update, you can manually update model definitions as described in this section.


To disable the automatic update for the model definitions, see [Preference Settings](#).

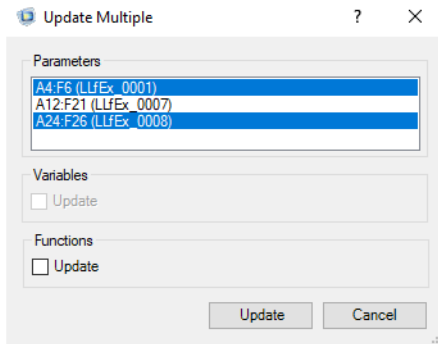
Click **Update** () to update the open model with definitions found in the range of cells including the selected cell. The update requires that there is a link to the model within the cell range. The model is not updated if the open model does not correspond to the model referenced in the link. Using this button you can update parameters, variables, and functions.

See [UpdateDefinitions](#) for information about the corresponding VBA command.

Click **Update>Update All** () to update the model with all model definitions found in the current worksheet. Only cell ranges containing a link to a model are used for the update.

See [UpdateAllDefinitions](#) for information about the corresponding VBA command.

Click **Update>Select** () to open the **Update Multiple** window where you can select the cell ranges (linked to parameters nodes) to update.




In the **Parameters** section, select the cell ranges with parameters to be updated in the model. If you have edited functions or variables inserted into the worksheet, you can also select the corresponding **Update** check boxes to update the model with the values from the worksheet. Click **Update** to apply the changes to the model definition in the open model, and to close the window. Click **Cancel** to close the window without updating the model.


Study

The **Study** group provides access to parametric sweep settings, and makes it possible to compute the solution for a selected study from the model.


COMPUTE

Click **Compute** () to solve the model linked to the current workbook. By default, this action computes the first study node in the model.


STUDY

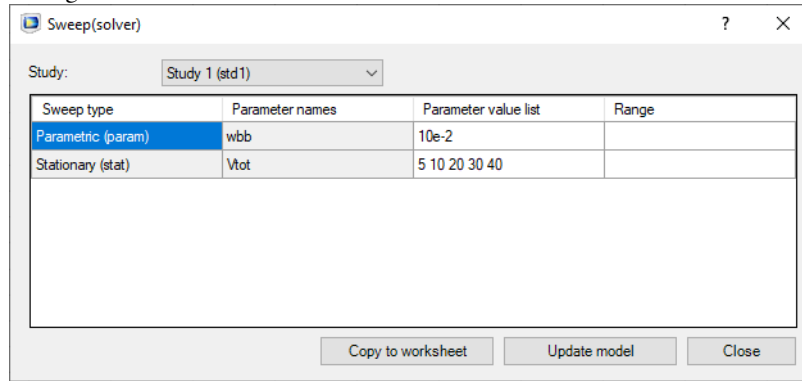
Click **Study** () then select a study node from the model for computation. Selecting the study computes the solution.

UPDATE SOLUTION

Click **Update Solution** () to update all solutions in the model linked to the current workbook.

SWEEP

Click **Sweep** () to open the **Sweep** window where you can view the parametric sweep settings in the model.



From the **Study** list select the Study node for which you want to see the parametric sweeps. The list contains all sweep parameters and their values as defined in the selected Study node, including Auxiliary sweeps.

Click **Copy to Worksheet** to insert the parameter names and values into the current worksheet. The data will be inserted at the selected cell, and a link to the model, in form of a comment, is automatically added to the cells.

In the worksheet select the cell range with sweep settings linked to a Study node in the model, then click **Update Model** to update the model with the parameters and values defined in the cell range.


Click **Close** to close the **Sweep** window.

See [Sweep](#) for information about the corresponding VBA command.

Numerical Results

With the tools in the **Numerical Results** group you can evaluate expressions and insert the evaluation results from a table in the model into the worksheet. In order to extract numerical results from a model, first select an empty cell in the worksheet where the data will be inserted.

PARAMETERS

Click **Parameters** () to insert the parameters found in the **Parameters** node in the **Results** branch of the model tree into the worksheet. In contrast to the parameters

available in the **Definitions** section, the results parameters can be used for calculating results without having to update or solve the model.

Parameters are added to the worksheet starting at the currently active cell, where a comment is also inserted linking the cell range to the model and the Parameters node.

See [ResultsParameters](#) for information about the corresponding VBA command.

DERIVED VALUES

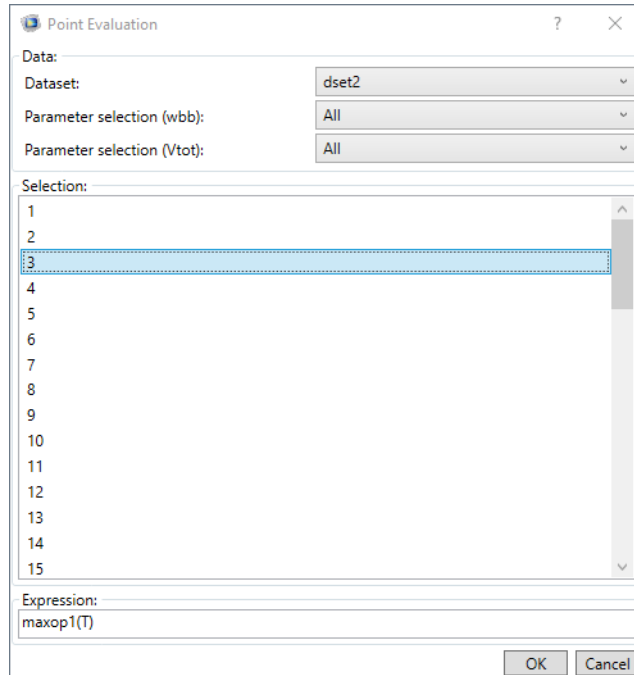
Click **Derived Values** ([8.85](#) [8-12](#)) and select a Derived Values node to evaluate and insert into the worksheet. Expressions need to be defined under the Derived Values node of the model tree to be displayed in the list.

The values are added to the worksheet starting at the currently active cell, where a comment is also inserted linking the cell range to the model and the evaluated Derived Values node.

See [ResultsDerivedValue](#) for information about the corresponding VBA command.

POINT EVALUATION

Click **Point Evaluation** ([8.85](#) [8-12](#)) to open the **Point Evaluation** window.



In the **Point Evaluation** window select the **Dataset** to evaluate on, and specify the points where the expression should be evaluated. Enter a valid expression into the **Expression** field; then click **OK** to perform the evaluation. The results are inserted into the worksheet at the selected cell, and linked to the model with a comment on the cell.

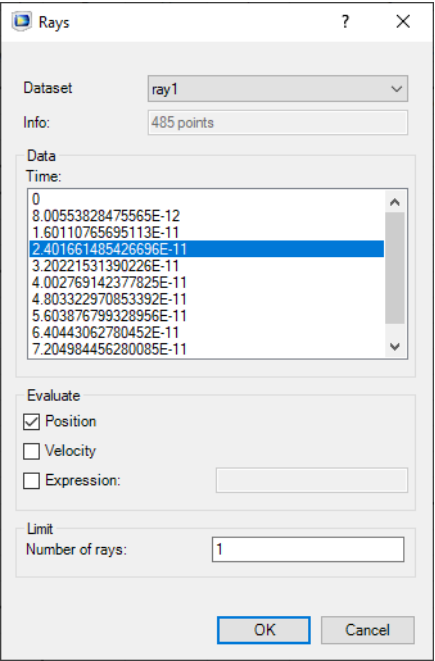
See [ResultsPointEvaluation](#) for information about the corresponding VBA command.

RAY EVALUATION

Under **Ray and Particle** click **Ray Evaluation**  to open the **Rays** window where you can evaluate expressions along ray trajectories.



The ray evaluation requires a license for the Ray Tracing Module.




Select a ray dataset from the **Dataset** list. From the **Evaluate** section select any of the following options:

- **Position**, to include in the evaluation the ray position along the trajectories.
- **Velocity**, to include in the evaluation the ray velocity along the trajectories.
- **Expression**, to include in the evaluation along ray trajectories the expression you specify.

Reduce the evaluation output by entering the number of rays for the evaluation in the **Number of rays** text field. Click **OK** to perform the evaluation and to insert the results into the worksheet. The results are inserted at the selected cell, and linked to the model with a comment on the cell.

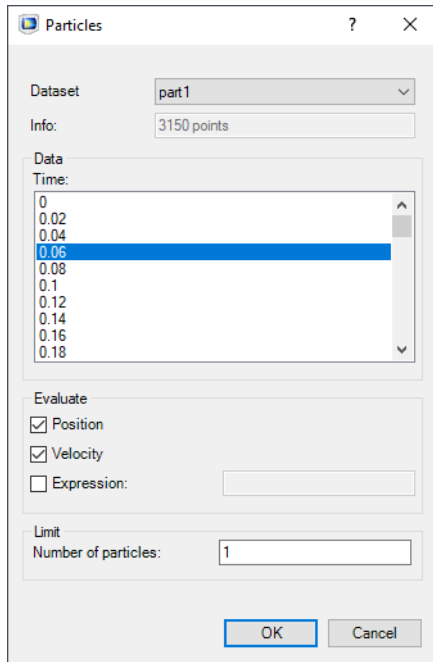
See [ResultsRayEvaluation](#) for information about the corresponding VBA command.

PARTICLE EVALUATION

Under **Ray and Particle** click **Particle Evaluation** () to open the **Particles** window where you can evaluate expressions along particle trajectories.



The particle evaluation requires a license for the Particle Tracing Module.



Select a particle dataset from the **Dataset** list. From the **Evaluate** section select any of the following options:

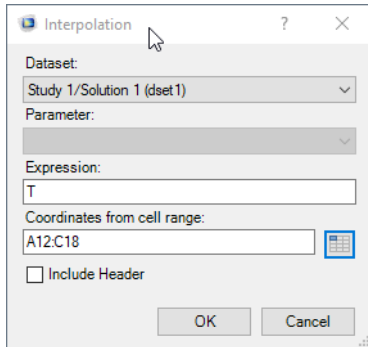
- **Position**, to include in the evaluation the particle position along the trajectories.
- **Velocity**, to include in the evaluation the particle velocity along the trajectories.
- **Expression**, to include in the evaluation along particle trajectories the expression you specify.

Reduce the evaluation output by entering the number of particles for the evaluation in the **Number of particle** text field. Click **OK** to perform the evaluation and to insert the results into the worksheet. The results are inserted at the selected cell, and linked to the model with a comment on the cell.

See [ResultsParticleEvaluation](#) for information about the corresponding VBA command.

INTERPOLATION

Click **Interpolation** (🔍) to open the window with the same name where you can set up the evaluation of expressions at specified coordinates.

The image shows the 'Interpolation' dialog box in COMSOL. It has a title bar with a question mark and a close button. The 'Dataset' dropdown is set to 'Study 1/Solution 1 (dset1)'. The 'Parameter' dropdown is empty. The 'Expression' text field contains 'T'. The 'Coordinates from cell range' text field contains 'A12:C18', with a small grid icon to its right. There is an unchecked checkbox labeled 'Include Header'. At the bottom are 'OK' and 'Cancel' buttons.

From the **Dataset** list, choose the solution dataset from the model. For solution datasets of type parametric, select the parameter value from the **Parameter** list. In the **Expression** text field enter the expression to evaluate. Enter the cell range that contains the coordinates for the interpolation points in the **Coordinates from cell range** text field, or click the **Select range** button (📊) to select the range from the worksheet. Make sure that you have formatted the point coordinate data such that the x-, y-, and z-coordinates are listed in separate columns, with each row defining an evaluation point, as illustrated in the figure below:


x	y	z
0	0	0
2.50E-02	0	0
5.00E-02	0	0
0	-1.25E-02	0
2.50E-02	-1.25E-02	0
5.00E-02	-1.25E-02	0

Select **Include Header** to add a header row to the evaluated data. In this case the data rows are shifted one row down, and a header row is included at the location of the cell comment.

Finally, click **OK** to evaluate and insert the results into the worksheet. The results are inserted at the selected cell, and linked to the model with a comment on the cell.


See [ResultsInterpolation](#) for information about the corresponding VBA command.

TABLES

Click **Tables** () and select a table from the model for inserting into the worksheet. The table data is inserted at the selected cell, and linked to the model with a comment on the cell.


See [ResultsTable](#) for information about the corresponding VBA command.

EVALUATION GROUPS

Click **Evaluation Groups** () and select an Evaluation Group node from the model for evaluating and inserting into the worksheet. The data is inserted at the selected cell, and linked to the model with a comment on the cell.


See [EvaluationGroup](#) for information about the corresponding VBA command.

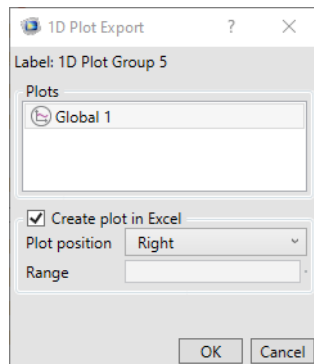
ID PLOT EXPORT

Use **ID Plot Export** () to extract data from 1D plot groups, insert the data into the worksheet, and create an Excel plot.



This functionality is only available for 1D plot groups, since the chart types available in Excel do not support the type of 2D and 3D plots that are commonly produced in models.

To extract plot data from the model, first select the cell in the worksheet where you want to insert the data, then click **ID Plot Export** () and select one of the 1D plot groups that are defined in the model. This opens the **ID Plot Export** window where you can change the settings.




From the **Plots** list select the plots to extract data from. By default all plots from the plot group are selected.

Select the **Create plot in Excel** check box (selected by default) to generate an Excel chart with the inserted data. To specify where the generated Excel chart is placed in relation to the inserted data select one of the options from the **Plot position** list: **Left**, **Right** (default), **Top**, **Bottom**, or **Custom** to enter the cell in the **Range** field for placing the top-left corner of the chart.


See [Export1DPlot](#) for information about the corresponding VBA command.

UPDATE

Click **Update** () to update all data evaluations in the current worksheet. Only cell ranges containing a valid link to the model are updated. Note that this will not update cells linked to results tables in the model, to do this see [Clear and Evaluate All](#).

See [UpdateAllResults](#) for information about the corresponding VBA command.


CLEAR AND EVALUATE ALL

Click **Clear and Evaluate All** () to clear, on the current worksheet, cells linked to results tables in the model, evaluate all derived values, and then update all data evaluations, and the cells linked to results tables.

See [ClearAndEvaluateAllResults](#) for information about the corresponding VBA command.


Report

Use the **Report** group to generate a report using the current model and write it in a given format.


Under **Report** (), select the type of report you want to generate. It can be a brief, intermediate, or complete report, depending the level of detail you want.



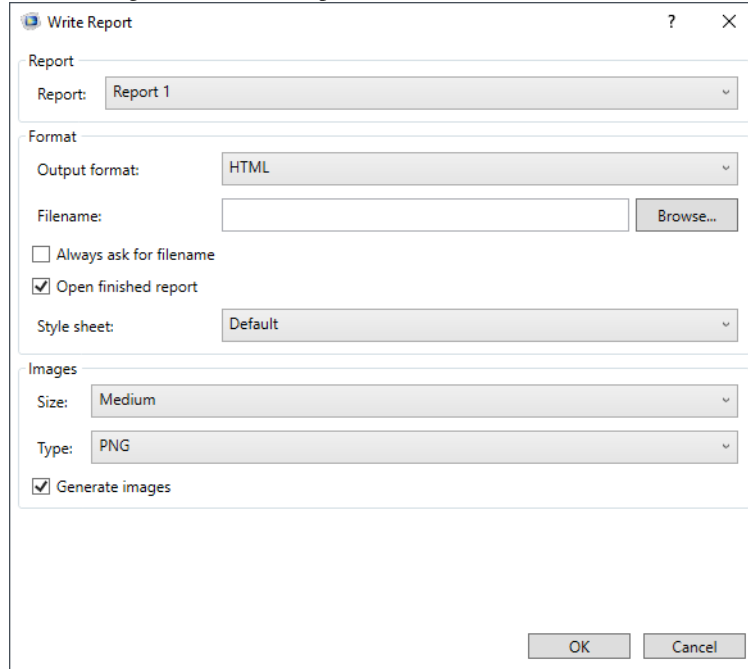
A description of the built-in report types is included in the *COMSOL Multiphysics Reference Manual*, under [Reports and Presentations](#).

To create a report based on user-defined templates available in the model, click **Report > Template** (), and select a template for the report.

When generating a new report in the model you are asked to enter the report node label in the **Generate Report** window.

If the model already contains reports, click **Report > Regenerate** () and select the report to update with the current model settings.

Click **Report > Write** () to open the **Write Report** windows where you can set the file and image format of the report.



The **Write Report** dialog box is divided into four sections: **Report**, **Format**, **Filename**, and **Images**. The **Report** section has a dropdown menu for 'Report' with 'Report 1' selected. The **Format** section includes a dropdown for 'Output format' set to 'HTML', a text field for 'Filename' with a 'Browse...' button, a checkbox for 'Always ask for filename' (unchecked), a checkbox for 'Open finished report' (checked), and a dropdown for 'Style sheet' set to 'Default'. The **Images** section has a dropdown for 'Size' set to 'Medium', a dropdown for 'Type' set to 'PNG', and a checkbox for 'Generate images' (checked). At the bottom right are 'OK' and 'Cancel' buttons.

In the **Report** section, select which report to write. The settings in the window are automatically updated from the selected report properties.

In the **Format** section, you can choose between HTML and Microsoft Word® (.docx) format as output format.

In the **Filename** field, enter the name of the report file. You can also browse to a folder to specify where to save the file. If no path is defined in the filename, the model path is used.

Select **Open finished report** if you want to visualize the report just after being generated.



You can choose between default and customized style sheets in the corresponding **Style sheet** list. When using a custom style sheet, browse to the style-sheet file, in the .css format, to use it.

In the **Images** section, you specify the images format to be included in the report. You can choose between five sizes, from extra small to extra large; the default image size is set to medium. In the **Type** list, you specify the image format to generate. You can choose between PNG, JPEG, or BMP.

Material Export


Use the **Material Export** group to export data from a worksheet to a material library.

Click **Settings** () to open the **Material Export** window where you can configure the export. For details, see [Exporting Material Data](#).


After you configure the data click **New** () to create a new material library or click **Append** () to add the material definitions to an existing library.

Click **Batch** () to open the **Material Export Batch** window where you can configure the batch export. For details, see [Exporting the Material Library in Batch](#).

Help

Click **Documentation** () to open a browser with the COMSOL Multiphysics documentation.

See [OpenDocumentation](#) to get more information about the corresponding command in VBA.

Click **Help** () to access the *LiveLink™ for Excel® User's Guide*.

See [OpenHelp](#) for information about the corresponding VBA command.

Working With Models From Worksheets

In this section:

- [Accessing the Model Definitions](#)
- [Computing the Solution](#)
- [Running a Model in Sweep](#)
- [Evaluating the Results](#)
- [Displaying the Results](#)
- [Updating Data in Cells Linked to the Model](#)

Accessing the Model Definitions

The LiveLink™ interface has the functionality to import model definitions to a worksheet — for example, parameters, variables, and functions. You find the related tools grouped under **Definitions** on the **COMSOL** tab in the Excel® user interface.

PARAMETERS

To import all global parameters, select a cell in the worksheet and click the **Parameters** button. The parameters are inserted to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, ensures that you can update the model with changes to the parameters in the worksheet. The imported data is formatted as in [Figure 2-1](#), with the parameter definitions organized into six columns.

Parameter	Expressions	Unit	Description	Value	Unit
param1		1 m	First parameter		1 m
param2		2 K	Second parameter		2 K
extern_param3		3 s	Third parameter		3 s

Figure 2-1: Model parameters after import to a worksheet.

The **Value** column is automatically updated with the expressions you enter. You can change this behavior in the Users Preferences window; see [Preference Settings](#).

The first **Unit** column is the unit to append to the expression, while the second Unit column is the unit for the parameter evaluation. You can change the formatting to

include the parameter unit together its expression in the users preferences; see [Preference Settings](#).

Parameter	Expressions	Description	Value
param1	1[m]	First parameter	1[m]
param2	2[K]	second parameter	2[K]
extern_param3	3[s]	Third parameter	3[s]

Figure 2-2: Model parameters after import to a worksheet without the unit column.

If you want to import and link only some of the parameters, you can create several **Parameters** nodes in the COMSOL model. Then click **Parameters>Parameters Nodes**, to select and import the parameters defined in a parameters node only.

Parameter	Expressions	Unit	Description	Value	Unit
Parameters 2					
param2		2 K	Second parameter	2	K

Figure 2-3: Model parameters nodes import to a worksheet.

Alternatively define parameters with the prefix **extern_** in their name, and click **Parameters>Filter**, to import only these parameters.

Parameter	Expressions	Description	Value
extern_param3	3[s]	Third parameter	3[s]

Figure 2-4: Model parameters after filtered import to a worksheet.

VARIABLES

To import all model variables, select a cell in the worksheet and click the **Variables** button. Variables are inserted to the right and down from the selected cell. A link to the COMSOL model, represented by a comment inserted in the selected cell, ensures that the model can be updated with changes to the variables in the worksheet. After the import, the variable definitions are formatted as in [Figure 2-5](#).

Name	Expressions	Description
var1		1 Global variable 1
var2		2 Global variable 2
select_var3		3 Global variable 3
bnd1	x	Boundary variable 1
bnd2	y	Boundary variable 2
extern_bnd3	x+y	Boundary variable 3

Figure 2-5: Model variables inserted into a worksheet.

Variables>Select, to import all the variables from the selected feature node only.

You can edit imported variables in your worksheet. To update the model with the modified definitions, see [Updating Data in Cells Linked to the Model](#).

FUNCTIONS

To import function definitions from the model, select a cell in the worksheet and click the **Functions** button. The **Functions** window opens, allowing you to select analytic or interpolation functions in the model. The imported data is inserted to the right and down from the selected cell. A link to the COMSOL model, represented by a comment inserted in the selected cell, ensures that the model can be updated with changes to the function in the worksheet.

For an analytic function the data is imported into two columns as in [Figure 2-6](#).

Expression	x^2
------------	-------

Figure 2-6: Analytic function table format after import.



For an interpolation function the data is formatted as in [Figure 2-7](#).

t	f(t)
1	1
2	4
3	9
4	16

Figure 2-7: Interpolation function table format after import.

You can edit imported model functions in your worksheet. To update the model with the modified definitions, see [Updating Data in Cells Linked to the Model](#).


Computing the Solution

To solve the model directly from the Excel user interface, click the **Compute** button () from the **Study** group. Note that if the model contains several studies only the first study is computed. In this case you can also click the **Study** button (), and from the menu select the study to compute.



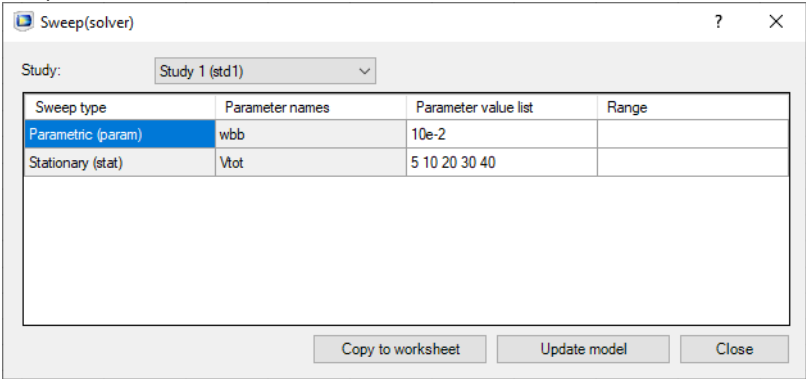
You have limited access to the solver settings from the **COMSOL** tab. It is recommended that you make sure that the model is suitably configured in the COMSOL Desktop before computing it from Excel.

Running a Model in Sweep

This section describes how to run a model in sweep using data value from the worksheet. To access the sweep parameter value list, select **Sweep** () to open the Sweep window.

If you plan on linking a cell range in the worksheet with the parameter list in the model, make sure to select the cell where the link is to be included before you open the Sweep window.

The Sweep window contains the list of the parameter name and value set in the parametric sweep node and in the auxiliary sweep settings available in the selected study.



In the **Study** list you can select the study where to define the parameter sweep value.

The **Parameter names** column lists the model parameters defined to run in sweep. The **Parameter value list** column contains the associated value. The **Range** column lists the cell range in the worksheet linked to the model parameters. You can directly edit the **Parameter value list** column. Click **Update Model** to update the model with the edited values.

LINKING PARAMETER SWEEP VALUES WITH THE WORKSHEET

In the Sweep window, click **Copy to Worksheet** to insert the parameter sweep value list into the current worksheet. A link, represented by a cell comment, is created between the worksheet and the model. In the worksheet, the parameter sweeps are stored in line from the cell containing the comment, the first column including the parameter names.


If the parameter value list only contains data separated by a space or a comma, the data are inserted in separate cells. If the parameter value list contains a string, such as a range function, or includes units, the data are exported to one unique cell.

Name	Expressions			
wbb	5.00E-02	1.00E-01		0.15
extern_Vtot	10[mV] 20[mV] 40[mV] 80[mV] 160[mV]			

After the data has been inserted into the worksheet, the **Range** column is updated with the cell range defining the sweep parameter values. Note that a cell comment is also created for each parameter name allowing specific update of a selected parameter if necessary.


You can now edit the parameter value linked with the selected cell comment in the worksheet. Then close the **Sweep** window. The parameter value has to be defined in the cells beside the one containing the parameter name.

UPDATING THE PARAMETER VALUE LIST TO THE MODEL

If the worksheet contains a link between a range and a parameter sweep list, select the cell with the comment and select **Sweep** (). The **Parameter value list** and the **Range** columns are automatically updated with the value from the worksheet. Click **Update Model** to update the model with the value defined in the worksheet.




It is not necessary to have links between the worksheet and the model to modify the sweep parameter value list, you can edit the value in the **Sweep** window and click **Update Model**.

To update the model with the modified parameter value list, you need to select the cell with the comment and select **Sweep** (). The parameter value list now contains the value from the worksheet. Click **Update Model** to update the model with the value defined in the worksheet.


Evaluating the Results

With LiveLink™ for Excel® you can evaluate and insert simulation results into a worksheet. The tools needed for this are grouped under **Numerical Results** on the **COMSOL** tab in the Excel user interface. To perform the evaluations at least one solution dataset needs to be present in the model.


DERIVED VALUES

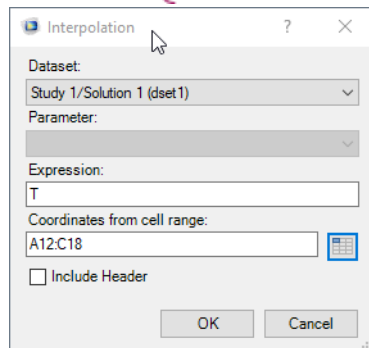
To evaluate data found in the Derived Value nodes defined in the model, select a cell in the worksheet, then click the **Derived Values** () button and select the Derived Values node to import the data from. The evaluated data is inserted to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, assures that the evaluation can be updated with changes to the model solution.

POINT EVALUATION

To evaluate expressions on specified geometry vertices select a cell in the worksheet and click the **Point Evaluation** () button. In the **Point Evaluation** window, select the solution dataset, one or more points, and specify the expression to evaluate. Click **OK** to insert the evaluated data to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, assures that the point evaluation can be updated with changes to the model solution.

INTERPOLATION


To evaluate expressions at arbitrary location select a cell in the worksheet, click **Interpolation** () to open the **Interpolation** window.

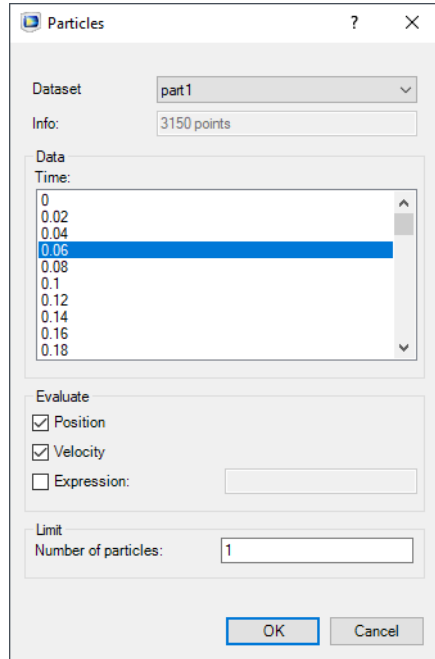


Select the solution dataset, if a parametric solution dataset specify the parameter value. Enter the expression to evaluate, and the cell range that contains the point coordinates for the evaluation. The coordinates need to be formatted so there is one point per row, with the coordinates in different columns. The number of evaluation points correspond to the number of rows.

The evaluated data is inserted to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, assures that the point evaluation can be updated with changes to the model solution.

PARTICLE EVALUATION

To evaluate expressions along particle trajectories, first select a cell in the worksheet then, under **Ray and Particle**, click the **Particles**  button.

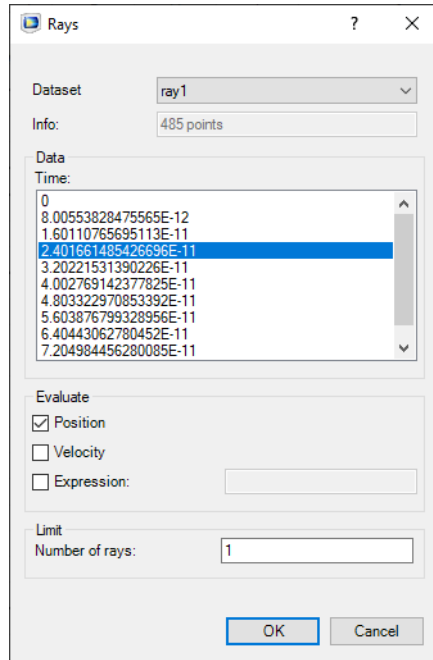


In the **Particles** window, select a valid particle tracing solution dataset. Then specify the time for data evaluation in the **Time** list. Under the **Evaluate** section select what to extract to the worksheet: the **Position**, the **Velocity**, and an **Expression**, which you can enter in the text field. Finally you can define the number of particles for which you want to perform the evaluation. The amount of particles in the particle dataset is available in the **Info** field, under the **Dataset** list.

The evaluated data is inserted to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, ensures that the point evaluation can be updated with changes to the model solution.

RAY EVALUATION

To evaluate expressions along ray trajectories, first select a cell in the worksheet then, under **Ray and Particle**, click the **Rays** (8.85) button.



Rays

Dataset: ray1

Info: 485 points

Data

Time:

- 0
- 8.00553828475565E-12
- 1.60110765695113E-11
- 2.401661485426696E-11**
- 3.20221531390226E-11
- 4.002769142377825E-11
- 4.803322970853392E-11
- 5.603876799328956E-11
- 6.40443062780452E-11
- 7.204984456280085E-11

Evaluate

☒ Position

☐ Velocity

☐ Expression:

Limit

Number of rays:

OK Cancel

In the **Rays** dialog box, select a valid ray tracing solution dataset. Then specify the time for data evaluation in the **Time** list. Under **Evaluate**, you select what to extract in the worksheet: the position, the velocity, and a specified expression. Finally, you can define the number of rays for which you want to perform the evaluation. The amount of rays in the Ray dataset is available in the **Info** field under the **Dataset** list.

The evaluated data is inserted to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, ensures that the point evaluation can be updated with changes to the model solution.

TABLES

To extract data from tables defined in the model, select a cell in the worksheet, click the **Tables** (8.86) button, and select the table to insert into the worksheet. The evaluated data is inserted to the right and down from the selected cell. A link to the model, represented by a comment inserted in the selected cell, ensures that the point evaluation can be updated with changes to the model solution.


MODIFY THE EVALUATION SETTINGS

You can modify the settings for an existing data evaluation by following these steps:

- From a cell range containing evaluation results, select the cell that contains the link to the model.
- Click the button corresponding to the evaluation operation.
- Edit the settings for the evaluation.
- Click **OK** to perform the evaluation and replace the data in the cell range.


To update the existing evaluations after recomputing a solution, see [Updating Data in Cells Linked to the Model](#).

Displaying the Results


To select a plot group to display in the COMSOL Multiphysics Server graphics window, click the **Plot Group** button () and from the list select an available plot group from the model.



Graphics display are enable with graphics server only, see [Preference Settings](#) to enable graphics server.

If the plot group uses a solution dataset containing multiple solution parameters, you can specify the parameter you would like to view in the **Plot Data** dialog box accessible by clicking the **Plot Data** button ()

IMPORTING GRAPHICS

To insert a screenshot image of the current view of the COMSOL Multiphysics Server graphics window, click the **Insert Screenshot** button () . The inserted image can either be a screenshot of the current graphics window or an export of the current plot. You can specify how to import the image and its size in the **Preferences** window available from the COMSOL backstage view.


Updating Data in Cells Linked to the Model


UPDATING MODELS DEFINITIONS


You can edit imported model definitions in the worksheet. The cell comment maintains the link between the data in the worksheet and the model, so that you can update the model with the new data.

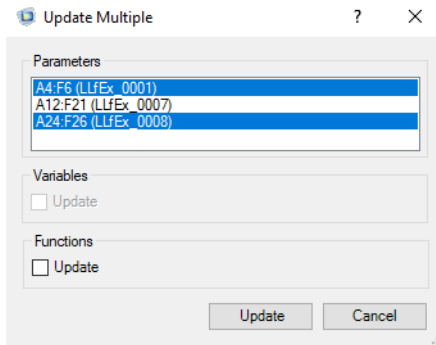


Imported parameters are updated automatically; see [Preference Settings](#) to disable automatic update.

To perform an update, select the cell with the comment and then click the **Update** button ()

To update all model definitions in a worksheet, click the **Update All** button () . The **Update All** button is accessible in the menu under the **Update** button.


If you have multiple cell ranges with model definitions, click the **Update > Select** button () , then select which cell ranges linked to parameters to use for the model update from the dialog box.



You can also update model definitions with imported the variables and functions tables by selecting the corresponding check boxes.


To solve the model after the update see [Computing the Solution](#).



UPDATING SWEEP PARAMETER VALUE LIST

First select the cell with the comment representing the link to the sweep node in the model. In the **Study** group, click the **Sweep** button () . In the dialog box that opens,


click **Update Model** to update the parametric sweep node with the one stored in the worksheet.

UPDATING NUMERICAL RESULTS

In the **Numerical Results** group, click the **Update** button () to update all results data found in a worksheet.

Click **Parameters** button () to import results parameters from the model. These parameters allow you to update simulation results without recomputing the solution. Once you have edited the parameters in the worksheet, click the **Update** button ().

THE CELL COMMENT

For model definitions and numerical results imported into the worksheet the LiveLink™ *for* Excel® interface adds a comment to the cell selected at the time of the import. This cell comment is a link to the model ensuring that the data can be updated. The comment can also contain the evaluation settings to allow subsequent editing of the evaluation operation. Click **Break Link** (), or **Break All Links**, to remove a comment from a cell, or all comments from the current worksheet respectively.



The data in the cell range cannot be updated if the comment is missing.

The COMSOL Backstage View

The COMSOL backstage view inside the Excel® user interface is where you can open and save Model MPH-files, connect to a COMSOL *server*, run COMSOL Apps, and change preference settings. To open the COMSOL backstage view, inside the Excel user interface, select **File** and click **COMSOL**.



The term *COMSOL server* designates either a COMSOL Multiphysics Server or COMSOL Server™ for running applications.

In this section:



- [Opening a Model MPH-File](#)
- [Saving a Model MPH-File](#)
- [Connecting Excel® to a COMSOL Server](#)
- [Disconnecting Excel® from a COMSOL Server](#)
- [Managing Model Loaded on a COMSOL Server](#)
- [Connecting the COMSOL Desktop® with the COMSOL Multiphysics Server](#)
- [Run Application](#)
- [Preference Settings](#)

Opening a Model MPH-File

To extract model definitions or results to an Excel® worksheet you need to link the workbook with a model loaded in a COMSOL *server*. You can directly open the model from Excel. If it is not already running, this starts a COMSOL Multiphysics Server in the background where the model is loaded. You can open any new Model MPH-file or open a model already linked in the workbook.

OPENING A NEW MODEL

Under the **File** Excel ribbon tab click **COMSOL** to access the COMSOL Backstage view.

Click **Open** (), and again **Open** (), then find the model MPH-file and click **OK**.

The model geometry, or a result plot when one is available in the model file, is displayed in an external graphics window. In Excel, the model name is inserted into the worksheet, see [Figure 2-8](#).

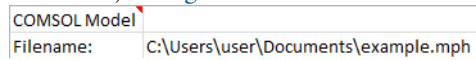


Figure 2-8: Spreadsheet after model import.


The model description and image, if contained in the model file, are also inserted. In addition, a comment is added to the cell selected before loading the model with the text **COMSOL Model**. This comment is used by LiveLink™ interface to maintain the connection between the worksheet and the model on the COMSOL Multiphysics Server.






Only one model can be opened per workbook. Opening another model updates the link to point to the last opened model.

The comment is automatically updated if you save the model under a different name. Removing the comment breaks the link between the workbook and the model.

OPENING A LINKED MODEL IN THE WORKSHEET

If the worksheet already contains a cell link with a model path you can open directly open the associated model. Under the **File** Excel ribbon tab click **COMSOL** to access the COMSOL Backstage view. Select **Open > Open linked** () , to open the file with the path defined in the cell comment link.

OPENING A MODEL FROM DATABASE

You can choose to open a model from a Model Manager database. Select **File** inside the Excel user interface, then click **COMSOL** to access the COMSOL Backstage view. Click **Open > Open from database** () , this opens a window where you can click any available Model Manager database () to open models from that database. For more details, see [Opening Models from Databases](#) in the *Model Manager Reference Manual*. Select a model and click the **Open** button () to load the model in the COMSOL Multiphysics Server and create a link to the model in the worksheet.

Unlike a regular model import, the file path is not displayed below the cell comment that link the model. Instead you will see the model title.

	A	B	C	D	E	F	G
1	COMSOL Model						
2	Title:	Electrical Heating in a Busbar Using LiveLink™ for Excel®					
3							



Saving file to a Model Manager database is not supported.

OPENING RECENT MODEL

Under the **File** Excel ribbon tab click **COMSOL** to access the COMSOL Backstage view. Click **Open** (📁), and then **Recent** (🕒), to get the list of the recent files open in a Multiphysics Server. Select the desired file and click **OK** to load the file in the server and establish the connection with the worksheet.

OPENING A MODEL CONTAINING USER DEFINED PHYSICS INTERFACE


If the model contains a user defined physics interface created using the COMSOL Physics Builder, you need to save the compiled archive (.jar) in your user home folder .comsol/v62/archives. Any compressed archive (with extension .jar) is loaded next time the COMSOL Multiphysics Server starts.

OPENING A MODEL WITH A CLASS KIT LICENSE

For Class Kit License (CKL) type, you need to start the COMSOL Multiphysics Server with the class kit license, see the [Preference Settings](#) to start the server with Class Kit License.

Saving a Model MPH-File

Click **Save** (💾) to save the currently open model on the COMSOL Multiphysics Server. The model is saved in the MPH-file format. Then the link between the spreadsheet and the model is updated automatically.

Click **Save As** () to save the currently open model file using a new name. The model is saved in the MPH-file format.hen the link between the spreadsheet and the model is updated automatically.




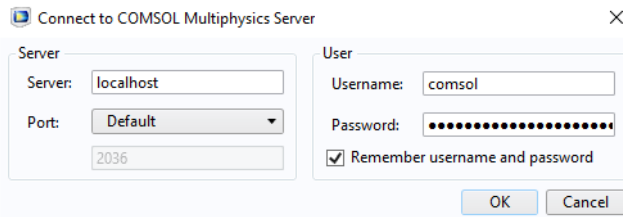
Saving the Excel workbook does not automatically save the linked model.

Connecting Excel[®] to a COMSOL Server

The default settings using LiveLink[™] for Excel[®] consist of starting a new COMSOL Multiphysics Server on the local machine the first time a model is loaded from the spreadsheet. You can specify a running COMSOL *server* to connect with Excel.

To specify the COMSOL *server* to be connected to Excel:

- 1 Start Excel and separately start a COMSOL *server*.
- 2 In Excel, go to **File** ribbon tab and select **COMSOL**.
- 3 In the COMSOL Backstage go to **Server** and click **Connect** (). The Connect to Server window appears.




- 4 In the **Server** text field enter the COMSOL *server* name (or server IP address). Enter localhost if the COMSOL *server* and Excel are running on the same machine.
- 5 In the **Port** menu list select **manual** if the COMSOL *server* is not listening to the default port number (2036). The port number is the one displayed in the COMSOL *server* window.
- 6 In the **User** and **Password** text field enter the login information requested by the COMSOL *server*.

7 Click **OK**.



When connecting Excel with a COMSOL Multiphysics Server not using graphics, you need to connect the server with a COMSOL Desktop to display graphics of the current model. See [Connecting the COMSOL Desktop® with the COMSOL Multiphysics Server](#).



Disconnecting Excel® from a COMSOL Server

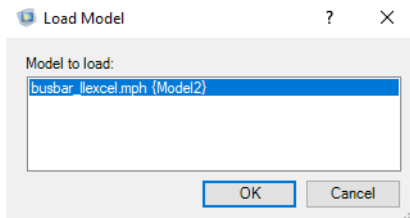
To disconnect Excel from a running COMSOL *server*, go to the **File** menu and select **COMSOL**. In the COMSOL backstage view, click **Disconnect** ()

Managing Model Loaded on a COMSOL Server

It is possible to import model that is loaded on a COMSOL *server* and to delete such models.



LOAD MODEL FROM SERVER

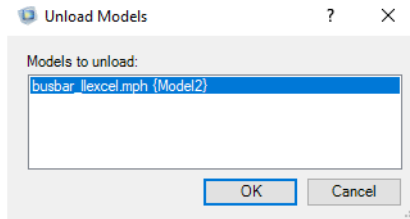
To load a model that is already loaded on a COMSOL *server* go to the Excel **File** menu and click **COMSOL**. In the COMSOL backstage view click **Server** () and then **Load Model** ()



In the Import Application from Server dialog box select in the list the application you want to open in the workbook and click **OK**.

UNLOAD MODEL FROM SERVER

To remove a model from a COMSOL *server* go to the Excel **File** menu and click **COMSOL**. In the COMSOL backstage view click **Server** () and then **Unload Models** ().




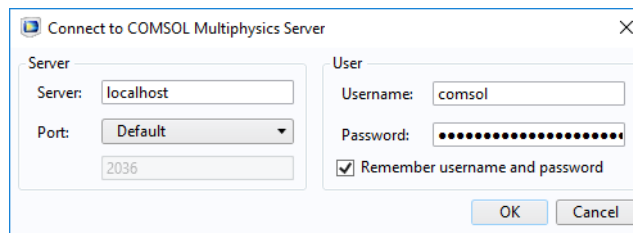
In the Remove Application from Server dialog box select in the list the application you want to remove from the COMSOL *server*.

Connecting the COMSOL Desktop[®] with the COMSOL Multiphysics Server


This section describes how to manually connect a COMSOL Desktop with a model open in Excel[®]. The COMSOL Multiphysics Server without graphics allows multiple clients connection, for example, Excel and a COMSOL Desktop. By default, Excel starts a COMSOL Multiphysics Server using graphics mode when opening a model. This server does not support multiple client connections. See [Preference Settings](#) to use a COMSOL Multiphysics Server without graphics.


To display the model open from Excel in a COMSOL Desktop:

- 1 Start Excel and load a model in Excel as indicated in [Opening a Model MPH-File](#).
- 2 In the COMSOL Desktop, from **File** menu (Windows users) or from the **Options** menu (Mac and Linux users), select **Client Server>Connect to Server** ().






You can now access the same application from both Excel and the COMSOL Desktop. Every change operated from Excel can be visualized from the COMSOL Desktop once the model is updated.

If you have already open an application in Excel and you want to open it also in the COMSOL Desktop, select **COMSOL Multiphysics Server>Import Application from Server** () from the **File** menu (Windows users) or from the **Options** menu (Mac and Linux users). In the **Import Application from Server** window select the model to import.


If you want to open an application loaded in the COMSOL *server* in Excel, click the arrow beside the **Connect to Server** button and select **Import Application from Server** ()

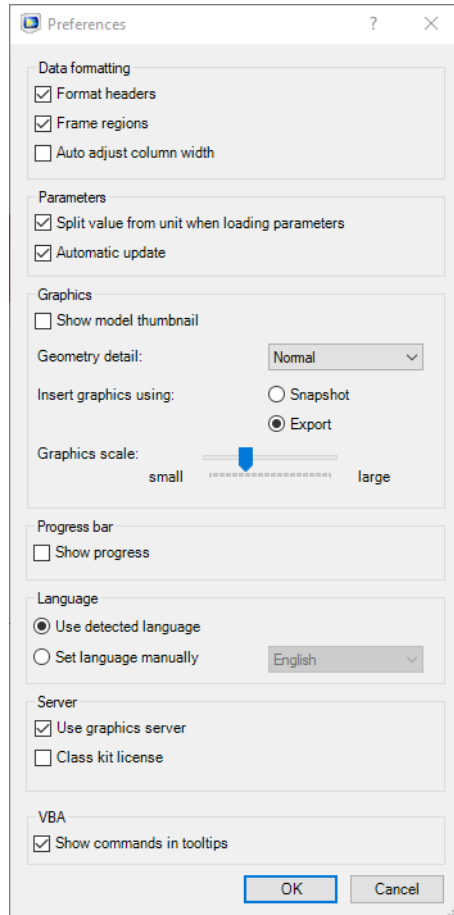
Run Application

From Excel® you can launch apps created using the COMSOL Applications Builder, go to the File menu and select COMSOL. In the COMSOL backstage view under **Applications** () , click **Run Application** () . Then browse to the MPH-file of the application to run and click Open.

You can also run apps that are loaded in the COMSOL Server. To open the COMSOL Server™ web interface from Excel, go to the File menu and select COMSOL. In the COMSOL backstage view under **Applications** () , click **COMSOL Server**() . Finally enter the credentials to connect to the server.

Preference Settings

Go to the **File** ribbon tab and click **COMSOL** to open the COMSOL Backstage view. Click **Preferences** () to open the **Preferences** window.



Under **Data formatting**, select **Format headers** to create column or row headers with bold fonts. **Frame regions** adds a frame around the inserted data. Select the **Auto adjust column width** check box to automatically adjust cell width to fit the imported data.

Under **Parameters**, clear the **Split value from unit when loading parameters** check box if you want the value and unit of parameters to be inserted together as a string into a single cell. By default, the check box is selected so that the parameter values are formatted as numbers, and the parameter units are imported into a separate cell. Clear

Automatic update if you do not want the parameters imported in Excel to have their values updated on the fly before computing the solution.

Under **Graphics**, select **Show model thumbnail** if you want the model image to be inserted into the worksheet when a model is loaded. In the **Geometry detail** list you can change the level of detail for geometry displayed in the COMSOL Multiphysics Server graphics window. Click **Snapshot** to insert the graphics as it looks like in the Graphics window (including background image). Click **Export** to insert the graphics without background image. Use the **Graphics scale** slider to change the size of the image to be inserted into the worksheet; see [Insert Screenshot](#).

Under **Progress bar**, select **Show progress** to display an external progress bar. The progress bar is shown when plotting the geometry or the mesh and when computing the solution.

Under **Language** select **Set language manually** radio button to manually define the Display and Screen Tip language, choose the language in the list box. **Select Use detected language** to use the language set in Excel.

Under **Server**, select the **Use a graphics server** check box (selected by default) to enable the graphics window for the COMSOL Multiphysics Server the next time it is started automatically. In this mode COMSOL Multiphysics Server has a graphics window to display the geometry, the mesh, and the result plots.

Select **Class kit license** to connect to a COMSOL Multiphysics Server that is running using a class kit license type.

In the **VBA** section, select the **Show commands in tooltips** check box (selected by default) to display information about the VBA commands corresponding to the buttons in the COMSOL tab in the buttons' tooltips.


Exporting Material Data

To export material properties from a workbook in the Excel user interface to a COMSOL material library, use the tools from the **Material Export** group of the **COMSOL** tab.

LiveLink™ for Excel® supports both material properties stored in the spreadsheet as constant or as field dependent variables (as such as temperature dependent properties, B-H curve, and so forth). Depending on the original format, you have to specify the export settings in different windows.

Automatically export the data to a new COMSOL material library or to an existing one. Perform the export for a single worksheet or in batch operations if the data are stored in different workbooks.

The Material Export Settings Window

Click **Settings** () in the **Material** group to open the **Material Export**. Specify the material and the properties to export to a COMSOL material library format. The material export procedure depends on the data format in the worksheet; the data can be defined as constant or as field variables.

EXPORT MATERIAL FROM DATA STORED AS CONSTANT

In the **Field dependent data** tab, specify the data format in the worksheet. If the worksheet contains only constant data formats, make sure that **Sheet contains field dependent data** check box is not selected.

The screenshot shows the 'Settings - Material Export' dialog box with the 'Field Dependent Data' tab selected. The 'Material Properties' sub-tab is also active. A checkbox labeled 'Sheet contains field dependent data' is unchecked. Below it, the 'Field variable settings' section contains a 'Field variable name range' input field, a unit dropdown menu showing '[Pa]', and an 'Assigned model input' dropdown menu showing 'Absolute Pressure, pA'. An 'Instructions' section below provides guidance on selecting the field variable name range and the assigned model input. At the bottom, another 'Instructions' section explains the 'Sheet contains field dependent data' checkbox and the 'Material properties' section. 'OK' and 'Cancel' buttons are at the bottom right.

Settings - Material Export

Field Dependent Data | Material Properties

☐ Sheet contains field dependent data

Field variable settings

Field variable name range: Assigned model input: Absolute Pressure, pA

Unit: [Pa]

Instructions

- Field variable name range: Select the cell with the name of the field variable.
- Assigned model input: Select a valid model input to assign to the field variable. Edit the unit to match the unit in the Excel sheet.

Instructions

- Sheet contains field dependent data: Select this option to export properties that depend on a variable, e.g. temperature.
- Material properties: Configure the export of the material properties.

OK Cancel

In the **Material properties** tab, specify the material and the properties to be included in the export.

Settings - Material Export

Field Dependent Data | **Material Properties**

Data range
 Material names range: A1 Water, liquid
 Property names range: B2:E2

Material property settings

Name	Type	Unit	Range
eta (Pa*s)			B2
Cp (J/(kg*K))			C2
rho (kg/m^3)			D2
k (W/(m*K))			E2

Valid properties (SI Units)

Filter:

- Compressibility of fluid - chif [1/Pa]
- Density - rho [kg/m^3]
- Diffusion coefficient - D [m^2/s]
- Dynamic viscosity - eta [Pa*s]**
- Electrical conductivity - sigma [S/m]
- Electron mobility - mue [m^2/(V*s)]
- Extinction coefficient - betaR [1/m]
- Frequency factor - alpha [1/s]

Unit:

Assign

Instructions

- ✓ Material names range: Select column with material names.
- ✓ Property names range: Select row with material property names.
- ✗ Material property settings: Assign valid properties and units.

Instructions

- Sheet contains field dependent data: Select this option to export properties that depend on a variable, e.g. temperature.
- ✗ Material properties: Configure the export of the material properties.

OK Cancel

In **Material names range** select the cell range that contains the material names. The material names have to be stored in a column in the worksheet.

In **Property names range** select the cell range that contains the property names. The property names have to be stored in a row in the worksheet.

In the **Material property settings** section you assign properties recognized by COMSOL to the properties in the worksheet. Select a property name from the **Name** column, then select a property from the **Valid properties (SI Units)** list.

Click **Assign** to assign the selected valid property to the select name from the current worksheet.

In the **Filter** field you can enter a text that is used to filter the valid properties list, for instance, ‘thermal conductivity’ or ‘sigma’ in order to access the thermal conductivity or the electrical conductivity, respectively.

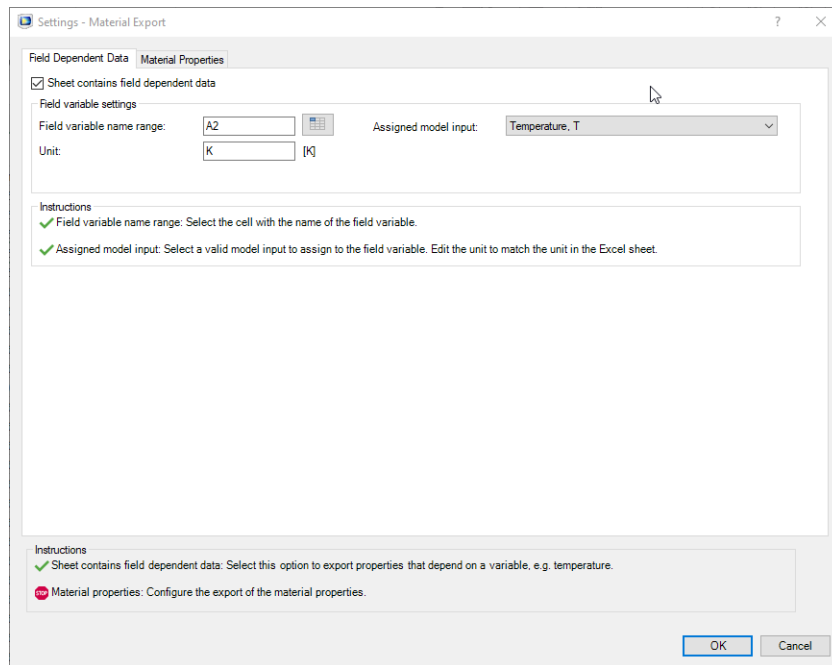
In the **Unit** column specify the unit in which the data is defined for the property. The default is SI units for all material properties.

Under the **Instructions** section you can follow the number of unassigned material properties for the selection. When you have completed the configuration all steps are marked with the icon ✓. If you have unassigned properties they will be indicated using the icon STOP.

Click **OK** to save the settings and to close the window.

EXPORT MATERIAL PROPERTIES FROM FIELD DEPENDENT DATA

In the **Field dependent data** tab select the **Sheet contains field dependent data** check box to specify the field variable settings.



In **Field variable name range** enter the cell number that contains the name of the field variable. The default data selection includes all data below the selected cell. You can manually specify the data cell range later in the **Material Properties** tab.

Once the field variable name is selected in the worksheet, in **Assigned model input** list select the model input in the COMSOL model that corresponds to the selected field variable name in the worksheet. The field dependent data is always stored in a single column. The field dependent data can correspond to the norm of the model input, for

instance with the following model input: the current density, the electric field, the magnetic field, the magnetic flux density, the stress tensor, and the velocity field.

In the **Unit** field enter the unit used in the worksheet if it differs from the default settings.

When you have completed the configuration all steps under the **Instructions** section are marked with the icon ✓.

In the **Material properties** tab specify the material and the properties to be included in the export.

Settings - Material Export

Field Dependent Data | **Material Properties**

Data range:

Material names range:

Property names range:

Material property settings

Name	Type	Unit	Range
eta (Pa*s)	dynamic...	Pa*s	B2
Cp (J/(kg*K))	heatcap...	J/(kg*K)	C2
rho (kg/m^3)	density	kg/m^3	D2
k (W/(m*K))	thermal...	W/(m*K)	E2

Assign

Valid properties (SI Units)

Filter:

- Seebeck coefficient - s [V/K]
- Speed of sound - cp [m/s]
- Storage - S [1/Pa]
- Surface emissivity - epsilon rad [1]
- Thermal conductivity - k [W/(m*K)]
- Young's modulus - E [Pa]
- Electrochemistry
- Electromagnetic Models

Unit:

Instructions

- ✓ Material names range: Select column with material names.
- ✓ Property names range: Select row with material property names.
- ✓ Material property settings: Assign valid properties and units.

Instructions

- ✓ Sheet contains field dependent data: Select this option to export properties that depend on a variable, e.g. temperature.
- ✓ Material properties: Configure the export of the material properties.

OK Cancel

In **Material names range** select the cell range that contains the material names. The material names have to be stored in a column in the worksheet.

In **Property names range** select the cell range that contains the property names. The property names have to be stored in a row in the worksheet.

In the **Material property settings** section assign properties recognized by COMSOL to the properties in the worksheet. Select a property name from the **Name** column, then select a property from the **Valid properties (SI Units)** list.


Click **Assign** to assign the selected valid property to the select name from the current worksheet.

In the **Filter** field you can enter a text that is used to filter the valid properties list, for instance ‘thermal conductivity’ or ‘sigma’ in order to access the thermal conductivity or the electrical conductivity, respectively.

In the **Unit** column specify the unit in which the data is defined for the property. The default is SI units for all material properties.

In the **Constant** column, select the radio button for the property you want to export as constant. If you select constant the value in the selected cell range in Property name value is used as property data to export in the material library.

In the **Range** column, enter the property data cell range to export. For constant data export select a unique cell. For field dependent data export specify the list of the data cell to export. The cell range has to be a unique column.


Under the **Instructions** section you can follow the number of unassigned material properties for the selection. When the configuration is completed, all steps are marked with the icon .

Click **OK** to save the settings and close the window.


The Cell Comment

After you have configured the material data export according to the previous section the settings are stored in comments added to the cells that contain the material names and material property names. These comments enable you to edit the settings later, and you can also easily copy the comments to another worksheet with similar data to skip the steps of configuring the material export again. Do not remove the comments from the cells unless you want to prevent a certain material or property from being exported to the material library.

Saving the Material Library

In the **Material Export** group select **Create** () to export the material properties to a new material library. In the **Save Material Library** window specify the filename and location. Click **OK** to export.

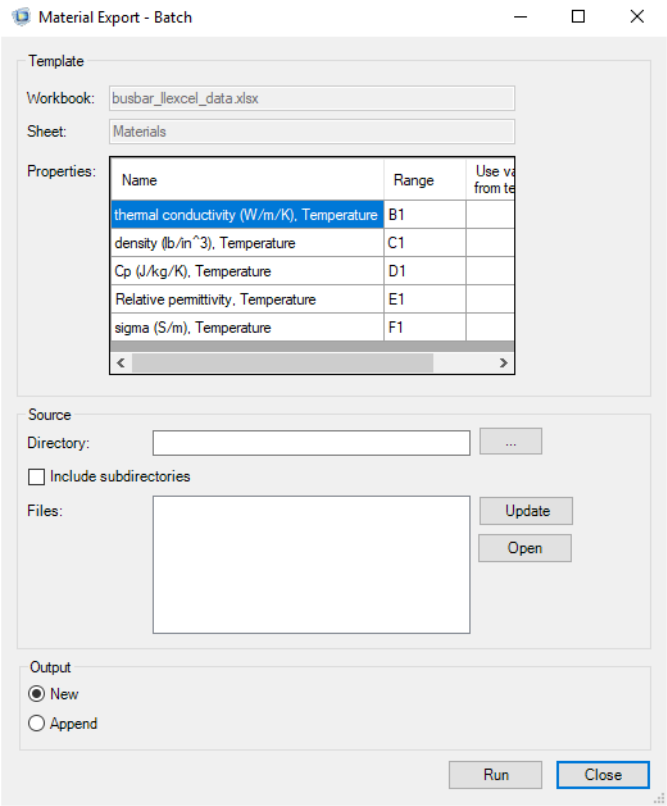
In the **Material Export** group select **Append** (📄) to export the material properties to an existing material library. In the **Append Material Library** window select an existing material library, and click **OK** to export.



COMSOL automatically finds and imports material libraries saved in the materials folder of the COMSOL preferences directory. This is set as the default location when you create new material libraries or when you append data to existing material libraries.

Exporting the Material Library in Batch

In the **Material Export** group select **Batch** (📁) to open the **Material Export - Batch**.



In the **Template** section you find the settings defined for the material export. The **Properties** table lists the material, the property name, and the cell range to export.

Select **Use value from template** if the property is only defined in the template worksheet but should be included in all material during the batch export.

In the **Source** section define the worksheet to use during the export. First, in the **Directory** text field enter the path of the directory that contains the source Excel files. Select **Include subdirectories** to include workbooks stored under the main directory. Click **Update** to list the files to be used during the batch export. If you need to open one of the listed files, select the file in the list and click **Open**.

In the **Output** section you define where to export the material data. Select **New** to export the material data in a new material library. Select **Append** to export the material data to an existing material library.

Loading and Saving Workbook Files

LiveLink™ adds the Excel® workbook (.xlsx) format to the list of formats for loading and saving model definitions from the COMSOL Desktop®. Note that it is not required to have Excel installed on the machine where the COMSOL is running. If you are using Windows and have a supported version of Excel installed you will have some extra features that are not provided otherwise. Read this chapter for a description of how to load and save tabular data to a workbook.

In this chapter:

- [Importing and Exporting Model Definitions](#)

Importing and Exporting Model Definitions


In this section:

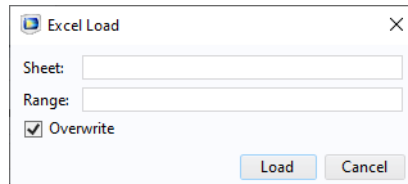
- [Support for Excel® Files](#)
- [Importing Data from a Workbook](#)
- [Exporting Data to a Workbook](#)
- [Supported Formats](#)

Support for Excel® Files

The following model features support the saving or loading of Excel® files: Parameters, Variables, Interpolation, Piecewise, and the study step including continuation parameters.

Importing Data from a Workbook

To load data from a worksheet to a table in the **Settings** window of a feature, click the **Load from File** button () below the table. In the file type list select **Microsoft Excel Workbook (*.xlsx)**. Select the desired file and click **Open**. Specify the sheet and cell range for the data.



Loading data from workbook files in a COMSOL Multiphysics model is also supported when Excel is not available.

In the **Sheet** text field enter the name of the worksheet containing the data. If no sheet name is defined, the first worksheet in the Excel file is selected by default.

In the **Range** field enter the cell range that holds the data. The range can be either defined using:

- the top-left cell number, to import all data up to the next empty cell.
- the cell range, to specify which cells are used for the import.
- empty cell range, to import the entire worksheet.

Also see [Supported Formats](#).

Clear the **Overwrite** check box to append the imported data to the end of the table. Note that you need to resolve conflicting data after the import.

Click **Load** to load the data to the table. Excel automatically starts in the background, if available. Excel will close after the import has been performed.



Excel usually locks the file when open it and hence the file cannot be loaded by other applications. Use Excel for reading the file to avoid this limitation.

IMPORTING MODEL PARAMETERS FROM A WORKBOOK

In case you want to import a list of parameters from a workbook, the Excel Load dialog box is slightly different as you have more import options.

Excel Load

Sheet:

Range:

☒ Overwrite


☐ Separate units column

☐ Calculated values

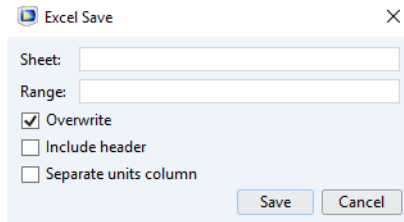
Load Cancel

For parameters import select **Separate units column** if the worksheet contains the units in separate but adjacent columns. Select **Calculated values** if the worksheet contains the parameters values in separate but adjacent columns.

Exporting Data to a Workbook

In the **Settings** window of the feature click the **Save to File** button (), usually located below the table. From the **Save as type** list select **Microsoft Excel Workbook (*.xlsx)**, then

specify the desired filename and click **Save**. Specify the sheet and cell range for the data export.



Saving data to workbook files from a COMSOL model is also supported when Excel is not available.

In the **Sheet** text field enter the worksheet name where to store the data. If no sheet name is defined, the first worksheet of the Excel workbook is selected by default.

In the **Range** field enter the cell range where data should be written. The range can be either defined using:

- a unique cell number defining the top cell in the worksheet.
- a cell range, to specify the cell where to save the data. Only the data within the specified cell range are exported.
- an empty cell range, all the data are saved starting from the cell A1.

Also see [Supported Formats](#).

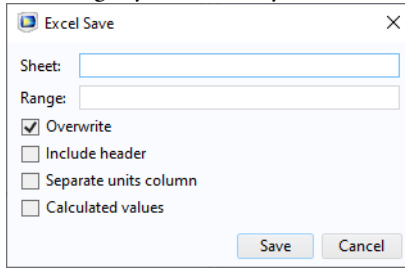
Clear the **Overwrite** check box if you want to append the data to the Excel file. Note that you can only append data to empty cells.

Select **Include headers** to export the table header row together with the data.

Click **Save** to save to export the data. Excel automatically starts in the background, if available. Excel will close after the export has been performed.

EXPORTING MODEL PARAMETERS TO A WORKBOOK

In case you want to export the model parameters to a workbook, the Excel Save dialog box is slightly different as you have more export options.



For parameters import select **Separate units column** to save the units in separate columns. Select **Calculated values** to save the parameter values in separate columns.

Supported Formats

The format of the data to be imported or exported depends on the model definition that you are working with. It is recommended that you export a table to a file to check how the data is formatted, you can then format data similarly for the import.

The following is an overview of the data format for each feature that supports reading or writing of Excel files:

- **Parameters:** the first column in the cell range defines the parameter name, the second its expression, and the third optional column the parameter description.

extern_L	9[cm]	Length of busbar
rad_1	6[mm]	Radius of bolts
tbb	5[mm]	Thickness of busbar

The parameter unit can also be defined in the file with an extra column included between the expression and the parameter description. For these data formats you need to select **Separate units column** in the Excel load dialog box.

extern_L	9 [cm]	Length of busbar
rad_1	6 [mm]	Radius of bolts
tbb	5 [mm]	Thickness of busbar

Same as for the units, the parameters value can be included in an extra column. For these data formats you need to select **Calculated values** in the Excel load dialog box.

extern_L	9[cm]	0.09 m	Length of busbar
rad_1	6[mm]	0.006 m	Radius of bolts
tbb	5[mm]	0.005 m	Thickness of busbar

If you want to include both units and parameters values, the units column must be placed between the expression and the parameters value columns. For these data formats you need to select both **Separate units column** and **Calculated values** in the Excel load dialog box.

extern_L	9 cm	0.09 m	Length of busbar
rad_1	6 mm	0.006 m	Radius of bolts
tbb	5 mm	0.005 m	Thickness of busbar

- **Variables:** the first column in the cell range defines the parameter name, the second its expression, and the third optional column the parameter description.

xi	abs(dest(x)-x)/D_i	
k	$1-(2*xi^3+3*xi)/(2*(xi^2+1)^{1.5})$	Integral kernel
Q_source	$4/(D_o^2-D_i^2)*epsilon*sigma_{const}*intop1(k*T^4)$	Heat source

- **Interpolation functions:** the first cell column in the cell range defines the value of the input argument parameter t , and the second column defines the value of the function $f(t)$.

293	5.0136
303	8.5
313	9.3272
323	9.8918
333	10.3303
343	10.6921
353	11.0012
363	11.2717
373	11.5123

- **Piecewise** functions: the first column in the cell range defines the start of the interval, the second column defines the end of the interval, and the third column defines the expression for the function.

0	1	x
1	2	x^3

- **Parametric Sweep** and **Auxiliary Sweep**: the first column in the cell range defines the parameter name, and the second column contains the list of the parameters separated by space.

L	5e-2 10e-2 15e-2
Vtot	5e-2 10e-2 20e-2 30e-2 40e-2

Exchange Workbook Using the COMSOL® API

The COMSOL API provides commands to exchange file with the model, set the filename including the Excel extension (.XLSX) to specify workbook.



See the *COMSOL Multiphysics Programming Reference Manual* for the following feature:

- `model.param()` and `model.result().param()`
- `model.variable()`
- `model.func()`

In the case the client/server connection such as using COMSOL with MATLAB, the Excel file is available on the client machine.

IMPORTING WORKBOOK USING THE COMSOL API

To load an Excel file type:

```
model.featur.loadFile(<filename>, "sheet", <range>);
```

where `<filename>` is the Excel filename including the .XLSX extension. `<range>` is the top-left cell range from which to import the data.

EXPORTING WORKBOOK USING THE COMSOL API

To save an Excel file type:

```
model.featur.saveFile(<filename>, "sheet", <range>, header, overwrite);
```

where `<filename>` is the Excel filename including the .XLSX extension. `<range>` is the top-left cell range to which saving the data. `header` is a Boolean you set to `true`

to include the table header and *false* to not include the table header. *overwrite* is a Boolean you set to *true* to overwrite nonempty cells without warning and *false* to not overwrite nonempty cells without warning.

Exchange Workbook in the Application Builder

LiveLink™ also supports workbook exchange with the model in the Application Builder using either a Table form object or dedicated Language Elements.

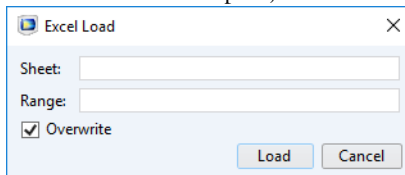
EXCHANGE WORKBOOK USING A TABLE FORM OBJECT

In the Application Builder you can add a Table form object with the Load from file and/or Save to file toolbars.



See [Table](#) section in the *COMSOL Multiphysics Application Builder Reference Manual*.

When running the application to load data from an Excel workbook click **Load from file** button and select Microsoft Excel Workbook (*.xlsx) as File format. Once you have selected the file to import, the Excel Load window pops-up.

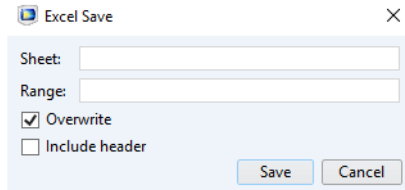


This is where you specify the worksheet and range where to import the data. In the **Range** field enter the top-left cell number or the cell range to load only the specified data within the cell range.

Clear **Overwrite** if you want to keep the data in the current table.

Click **Load** to import the table.

To save the data available in the table, click **Save to file** button and select Microsoft Excel Workbook (*.xlsx) as File format. Once you have selected the file where to save, the Excel Save window pops-up.



The image shows a standard Windows-style dialog box titled "Excel Save". It has a close button (X) in the top right corner. Inside the dialog, there are two text input fields: "Sheet:" and "Range:". Below these fields are two checkboxes: "Overwrite" (which is checked) and "Include header" (which is unchecked). At the bottom right of the dialog are two buttons: "Save" and "Cancel".

This is where you specify the worksheet and range where to export the data. In the **Range** field enter the top-left cell number or the cell range to save only the data that fit the specified cell range.

Clear **Overwrite** if you want to keep the nonempty cell in the file.

Select **Include header** if you want also to include the table header in the workbook.

Click **Save** to export the table.

EXCHANGE WORKBOOK USING THE METHOD EDITOR

LiveLink™ offers specific language elements in the Application Builder to exchange data between the application and an Excel workbook. These are `writeExcelFile` and `readExcelFile` to write to an Excel file and read the data from an Excel file, respectively.

Using the method editor, the file you are working can also be available on the server.



See [File Schemes and File Handling](#) section in the *COMSOL Multiphysics Application Builder Reference Manual*.

You will find below the commands to exchange data with an Excel workbook using the method editor available in the Application Builder.

```
writeExcelFile(String filename, String[][] data)
```

Writes the given string data to the first cell onward in the first sheet of an Excel-file.

```
writeExcelFile(String filename, String sheet, String cell,
String[][] data)
```

Writes the given string data to the specified cell onward on the specified sheet of an Excel-file.

The strings set in `data` can defines number, text, or Excel formulas as in the example below:

```
String [][] data = new String[][] {{ "Mydata", "=SUM(4,7)", "5",  
                                     "=SUM(A1:C1)" }}
```



Formulas have to be entered following English number format.

```
String[][] readExcelFile(String filename)
```

Reads from the first cell onward on the first sheet of an Excel-file into a `String[][]`.

```
String[][] readExcelFile(String filename, String sheet,  
                          String cell)
```

Reads from the specified cell onward on the specified sheet of an Excel-file into a `String[][]`.



See the application example Beam Section Calculator (using LiveLink™ for Excel®) in the LiveLink for Excel Application library.

The COMSOL API for VBA

LiveLink™ *for* Excel® supports Microsoft® Excel® built-in language, Visual Basic® for Applications (VBA), to access and modify the COMSOL model directly from a worksheet. Using Visual Basic for Applications makes it possible to control the content of Excel workbooks, create user interfaces, and perform advanced calculations.

In this chapter:

- [Introduction](#)
- [Before You Start](#)
- [Working with Models Using VBA](#)
- [Objects and Methods](#)
- [Utility Methods](#)

Introduction

In this section:

- [Support for VBA](#)
- [Introductory Example](#)

Support for VBA

Visual Basic for Applications (VBA) is a macro language that is built into Excel. This language makes it possible to control the content of workbooks, create user interfaces and perform advanced calculations. Using LiveLink™ for Excel® you can connect to a COMSOL server from Excel and access all the features of the COMSOL Application Programming Interface (API).

The interface based on VBA is using the model structure and the methods (commands) for accessing the model settings and data provided by the COMSOL API, which programming syntax is similar. You can refer to the *COMSOL Programming Reference Manual* for more information.

The following parts of the documentation show the steps necessary to work with COMSOL models starting with a very simple example based on an existing model.



The *LiveLink™ for Excel® User's Guide* assumes that you have some knowledge of VBA.



The link information, which is added as comments to cells in a worksheet when using the buttons from the COMSOL tab and the methods from the RibbonUtil class, is not supported by methods in the COMSOL API for VBA.



See [Updating and Solving a Model Using Excel® Macros](#) for short examples to get started with using the COMSOL API for VBA.

Introductory Example

Follow this short example that illustrates some of the possibilities for using VBA when working with a COMSOL model. Running the code in the example requires that LiveLink™ for Excel® is properly installed and configured, for information see [Before You Start](#).

To load or create a COMSOL model it is necessary to use the class ModelUtil, which is part of the COMSOL API. In addition to ModelUtil, the class ComsolUtil provides additional utility functions, for example for starting a COMSOL Multiphysics Server, and access the class RibbonUtil that contains the wrapper functions corresponding to the buttons of the COMSOL tab.

The example loads the busbar model from the Application Libraries, creates a plot, and extracts numerical data into a worksheet. Follow the below steps with detailed explanations, and type in the commands, or jump ahead and copy the entire subroutine from [Code for use with VBA](#), then paste it into the VBA editor.

- 1 In the **Developer** tab in Excel, click the **Visual Basic** button to start the Microsoft Visual Basic for Applications interface.
- 2 From the **Insert** menu, select **Module**.
- 3 In the Module 1 editor start editing the subroutine:

```
Sub Example()
```

- 4 Create the utility objects dedicated for LiveLink™ for Excel®:

```
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil
```

- 5 Open the model file busbar.mph from the COMSOL Multiphysics Application Libraries

```
Range("A1").Select  
Set model = RibbonUtil.OpenModel("busbar.mph")
```

You do not need to include the file path as long as you are opening a model from the COMSOL Application Libraries, or the model is saved at the same location as the current workbook. Once the model is loaded, a message is displayed in the COMSOL Multiphysics server window.

- 6 Write in the worksheet the coordinates to use for the data interpolation as in the figure below:

x	y	z
0	0	0
2.50E-02	0	0
5.00E-02	0	0
0	-1.25E-02	0
2.50E-02	-1.25E-02	0
5.00E-02	-1.25E-02	0
0	-2.50E-02	0
2.50E-02	-2.50E-02	0
5.00E-02	-2.50E-02	0

```

Range("A3").Value = "X"
Range("B3").Value = "Y"
Range("C3").Value = "Z"
Range("A4:C12").Value = 0
Range("A5").Value = 2.5e-2
Range("A6").Value = 5e-2
Range("B7").Value = -1.25e-2
Range("B10").Value = -2.5e-2
Range("A7:A9").Value = Range("A4:A6").Value
Range("A10:A12").Value = Range("A4:A6").Value
Range("B8:B9").Value = Range("B7").Value
Range("B11:B12").Value = Range("B10").Value

```

- 7 Extract the temperature from the model using interpolation coordinates

```

Range("D3").Select
RibbonUtil.ResultsInterpolation "dset1", "T", "A4:C12", True

```

The above code creates a link between Excel and the COMSOL model, which defines an interpolation evaluation at the coordinates provided in the cell range A4:C12. You can reuse later this link to update the evaluation after you have computed a new solution.

- 8 End the subroutine

```
End Sub
```

- 9 To run the subroutine click the **Run Sub/User Form** button or press F5

Once you have run the subroutine the ribbon is connected to the COMSOL Multiphysics Server. You do not need to open the model again.

Code for use with VBA

Below you find the full script of the example. You can copy it and paste it into the VBA editor. In the code below replace `<COMSOLPATH>` with your local COMSOL Multiphysics installation directory.

```

Sub Example()

Set ComsolUtil = CreateObject("comsolcom.comsolutil")
Set ModelUtil = CreateObject("comsolcom.modelutil")
Set RibbonUtil = ComsolUtil.GetRibbonUtil

Range("A1").Select
Set Model = RibbonUtil.OpenModel("busbar.mph")

Range("A3").Value = "X"
Range("B3").Value = "Y"
Range("C3").Value = "Z"
Range("A4:C12").Value = 0
Range("A5").Value = 2.5e-2
Range("A6").Value = 5e-2
Range("B7").Value = -1.25e-2
Range("B10").Value = -2.5e-2
Range("A7:A9").Value = Range("A4:A6").Value
Range("A10:A12").Value = Range("A4:A6").Value
Range("B8:B9").Value = Range("B7").Value
Range("B11:B12").Value = Range("B10").Value

Range("D3").Select
RibbonUtil.ResultsInterpolation "dset1", "T", "A4:C12", True

End Sub

```

Before You Start

If you have installed LiveLink™ for Excel® you are ready to use the COMSOL API commands in your VBA scripts. This chapter contains information on how to configure Excel to make this easier, and also how to manually register the COMSOL API component with Excel.

In this section:

- [Enabling the Developer Tab in Excel®](#)
- [Adding ComsolCom Reference to the VBA Project](#)
- [Registering the Component for the COMSOL API](#)
- [Saving Excel Workbooks Containing Macros](#)

Enabling the Developer Tab in Excel®

VBA is installed with Excel, and the VBA editor is accessible just by pressing Alt+F11 keys within Excel. To run the subroutine directly from the worksheet you need to enable the Developer toolbar in the Excel ribbon. In the Developer toolbar you will find the different control tools that can be inserted in the worksheet.

The steps below describe how to activate the Developer toolbar in the Excel ribbon.

- 1 In Excel right-click on any tab of the ribbon and select **Customize the Ribbon...**
- 2 In the **Excel Options** window, locate the **Main Tabs** list and select **Developer**.
- 3 Click **OK**.

The Developer tab is now available in the ribbon.

Adding ComsolCom Reference to the VBA Project

To be able to run the code created in the Visual Basic for Applications interface, you need to activate the ComsolCom references as static reference. To proceed, in the Microsoft Visual Basic for Applications window, go to **Tools** menu and click **References**. In the **Available References** list select **ComsolCom 6.2** and click **OK**.

Registering the Component for the COMSOL API

When installing LiveLink™ for Excel® the ComsolCom.dll component necessary for accessing the COMSOL API for VBA is installed and registered with Excel. You can also register or unregister this component manually by following the instructions below.

REGISTERING THE API COMPONENT

In order to register manually the ComsolCom.dll component, open a Command Prompt as an Administrator, then run the following command:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe  
"<COMSOLPATH>\ext\LiveLink\Excel\ComsolCom.dll" /codebase /tlb
```

where <COMSOLPATH> is the path to your COMSOL Multiphysics installation directory.

UNREGISTER THE API COMPONENT

In order to unregister manually the ComsolCom.dll component open a Command Prompt as an Administrator, then run the following command:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe  
"<COMSOLPATH>\ext\LiveLink\Excel\ComsolCom.dll" /u /tlb
```

where <COMSOLPATH> is the path to your COMSOL Multiphysics installation directory.

Saving Excel Workbooks Containing Macros

If you have an Excel workbook that contains VBA code (or macros) you need to save it as an Excel Macro-Enabled Workbook, which has the .xlsm extension.

Depending on your security settings for your computer as well as the security settings for Microsoft Office you may receive a warning when opening an Excel XLSM-file.

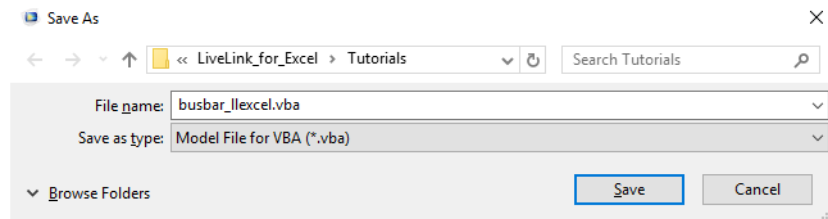
Working with Models Using VBA

In this section:

- [Saving the Model File for VBA](#)
- [The Model Object and the ModelUtil Object](#)
- [The ComsolUtil and RibbonUtil Objects](#)
- [ComsolCom Version Control](#)
- [Declaring Utility Objects with Dim](#)
- [Error Handling](#)
- [Starting and Connecting to a COMSOL Multiphysics Server](#)
- [Updating and Solving a Model Using Excel® Macros](#)

Saving the Model File for VBA

You can save the COMSOL model as a file for VBA ready to use in an Excel® macro. In the COMSOL Desktop go to **File** menu, choose **Save As**, and, in the **Save As** window, from the **Save as type** list select **Model File for VBA (*.vba)**.



The commands for each operation on the model are listed in the file in the order that they were performed, this way it is easy to get the corresponding command for a specific operation as it is the last one in the file.

If you do not want to save the entire model history and save only the commands that correspond to the current model status, in the File menu click **Compact History** prior to saving the model.

The Model Object and the ModelUtil Object

The model object is used by the COMSOL Desktop to represent the model. This means that the model object and the COMSOL Desktop behavior are virtually identical. The following applies:

- All algorithms and data structures for the model are integrated in the model object.
- The model object includes methods to set up and run *sequences of operations* to create geometry, meshes, and to solve your model.



The [Model Object](#) in the *COMSOL Multiphysics Programming Reference Manual*.

The model object has a large number of methods. The methods are structured in a tree-like way, similar to the nodes in the model tree in the *Model Builder* window on the COMSOL Desktop. The top-level methods just return references that support further methods. At a certain level the methods perform actions, such as adding data to the model object, performing computations, or returning data.



Detailed documentation about model object methods is in [About General Commands](#) in the *COMSOL Multiphysics Programming Reference Manual*.



The links to features described outside of this user guide do not work in the PDF, only from the online help.

In VBA you can access the model object methods using a ModelUtil object. To create a ModelUtil object enter:

```
Set modelutil = CreateObject("comsolcom.modelutil")
```

See [The COMSOL API](#) section to get more information about the COMSOL API methods accessible from the ModelUtil object.

The ComsolUtil and RibbonUtil Objects

LiveLink™ offers dedicated methods to interact with the COMSOL *server* from Excel. These methods are accessible using a ComsolUtil object.

To create a ComsolUtil object enter:

```
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
```

See [Methods in the ComsolUtil Class](#) section to get more information about the methods available using the ModelUtil object.

In addition to the COMSOL Utility object you can create the COMSOL RibbonUtil object, which allows to access each operation available from the COMSOL ribbon using specific methods.

To create a RibbonUtil object enter:

```
Set RibbonUtil = ComsolUtil.GetRibbonUtil
```

See [Methods in the RibbonUtil Class](#) section to get more information about the methods available using the ModelUtil object.

ComsolCom Version Control

It is possible to manually specify which ComsolCom version to run. To create a ComsolUtil and a ModelUtil object from a specific COMSOL version use the commands below:

```
Set ComsolUtil = CreateObject("comsolcom.comsolutil.<version>")  
Set ModelUtil = CreateObject("comsolcom.modelutil.<version>")
```

where *<version>* is the COMSOL version number, for instance 6.2 for the current version.

Once you have set a COMSOL version to use, make sure LiveLink™ for Excel® is installed with this version number.

If you do not want to specify the version number and use the highest COMSOL version available, use the commands below:

```
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set ModelUtil = CreateObject("comsolcom.modelutil")
```

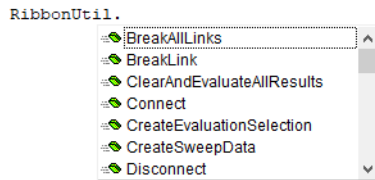
Note: Make sure the corresponding ComsolCom references is set as static reference to be able to run the code, see [Adding ComsolCom Reference to the VBA Project](#).

Declaring Utility Objects with Dim

It is possible to declare variables of the ModelUtil, ComsolUtil, and RibbonUtil types using Dim, to proceed enter the command below:

```
Dim ModelUtil As ComsolCom.ModelUtil
Dim ComsolUtil As ComsolCom.ComsolUtil
Dim RibbonUtil As IRibbonUtil
```

Once these are declared automatic completion is accessible as shown below:



You also get indication how to use the method:

```
RibbonUtil.ResultsPointEvaluation |
ResultsPointEvaluation([dataset As String], [expression As String], [pointIndices], [evaluationSelection])
```

Error Handling

When working with scripts it can be useful to catch any errors that either originate from COMSOL or from your own code. The VBA editor has settings that make it possible to catch errors in various ways. These are available in the **Options** window, which is accessible from the **Tools** menu of the VBA editor.

In the **Options** menu you can define the error handling settings on the **General** page, in the **Error Trapping** section:

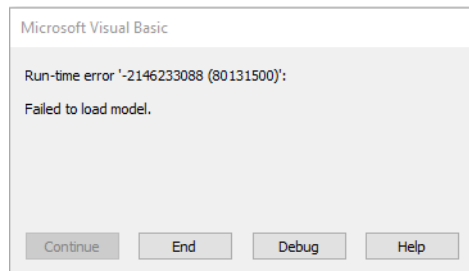
- Break on All Errors: breaks code execution whenever there is an error. Shows a VBA dialog box with the error message.
- Break on Class Module: stops at the actual error (line of code) when a class module has been written in VBA.
- Break on Unhandled Errors: breaks code execution whenever there is an error but also allows you to trap the error (using the On Error statement). If you fail to trap an error you will get a dialog box from VBA.

EXAMPLE

In the VBA editor enter the code below. The code starts a COMSOL Multiphysics Server and connects it with Excel. Finally, it tries to load a model MPH-file that does not exist, as it is meant to fail.

```
Sub loadfile_error()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
Set model = RibbonUtil.OpenModel("c:\\foo.mph")  
  
End Sub
```

Assuming that the file `foo.mph` does not exist and if the error trapping is set to Break on All Error, the following error message is returned:



The numbers do not have any useful meaning, but sometimes you are able to decipher the message in the window. In this case the message says something about The system cannot find the file specified, which is expected. Click the End button to stop the VBA code from running.

When writing your own code, you may want to be able to handle errors differently so that you can show a nice dialog box about the problem or perhaps fix the problem on the fly. This is especially useful if you write code that other people will use. The On Error statement allows you to handle any error that may occur.

Update the code above such as:

```
Sub loadfile_error()  
  
On Error GoTo errorHandler  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil
```

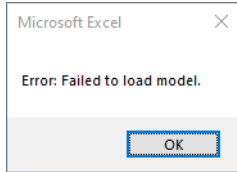
```

Set model = RibbonUtil.OpenModel("c:\\foo.mph")
Exit Sub

errorhandler:
Call MsgBox("Error: " + Err.Description)
End Sub

```

Make sure the Error trapping option is set to Break on Unhandled Errors and run the script. You will now get the following dialog box:



In this example we are just showing an error message and an OK button. For more advanced error handling you may want to customize the error message further and provide different options to the user.



Running a script without the On Error statement but with the Error trapping option set to Break on Unhandled Errors may return message without sufficient information.

Starting and Connecting to a COMSOL Multiphysics Server

The most common approach to start and connect Excel to a COMSOL Multiphysics Server is by opening a model using `OpenModel()` from the `RibbonUtil` utilities. However, `LiveLink™ for Excel®` also provides methods for starting and connecting to a server manually.

Ordinarily you would run `StartComsolServer()` to start the COMSOL Multiphysics Server:

```
Boolean ComsolUtil.StartComsolServer(useGraphics As Boolean)
```

This command will start the COMSOL Multiphysics Server in graphics mode when `useGraphics` is set to `True` on the local computer. A port number, which is used for the communication between the COMSOL server and Excel, is automatically assigned by the COMSOL server. The default number is 2036, but if other COMSOL servers are running then a higher number has to be chosen. It also returns a Boolean to indicate if the COMSOL server has started (`True`) or failed to start (`False`).

To connect Excel to the COMSOL *server* using the default port number, simply use the command:

```
Void RibbonUtil.Connect([hostname As String], [portnumber As Int],  
[login As String],[pwd As String])
```

where *hostname*, *portnumber*, *login*, and *pwd* are optional argument that set the connection credential: computer name, the port number the server is listening to, the user name and the password respectively. If you are already connected to a server and try to connect it again, nothing happens and you will remain connected. If however you try connect to a different server, it will disconnect automatically from the first one and connect to the specified one.

You can check if the RibbonUtil service is already connected to a server with the `IsConnected()` method:

```
Boolean RibbonUtil.IsConnected()
```

This returns `True` if already connected, `False` else.

To disconnect from the server use `Disconnect()`:

```
Void RibbonUtil.Disconnect()
```

Code for use with VBA

Below you find an example showing how to connect manually to a COMSOL Multiphysics server; feel free to copy and paste it to the VBA editor.

```
Sub ConnectServerExample()  
  
Dim ComsolUtil As ComsolUtil  
Dim RibbonUtil As IRibbonUtil  
  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
If Not RibbonUtil.IsConnected Then  
    ' Start a Multiphysics server  
    ok = ComsolUtil.StartComsolServer(True)  
    ' Return an error if not started properly  
    If ok = False Then  
        MsgBox "no server " & ComsolUtil.get_errormessage  
        Exit Sub  
    End If  
    ' Get the port number used  
    port = ComsolUtil.get_port  
    ' Connect to the server using the specified port number  
    (RibbonUtil.Disconnect  
    RibbonUtil.Connect "localhost", port
```

```
End If  
End Sub
```

Updating and Solving a Model Using Excel® Macros

In this section you will find several examples that illustrate some of the possible work flows when using the COMSOL API for VBA.



Some knowledge in VBA are required to run the examples. For more information please refer to the section [Before You Start](#).

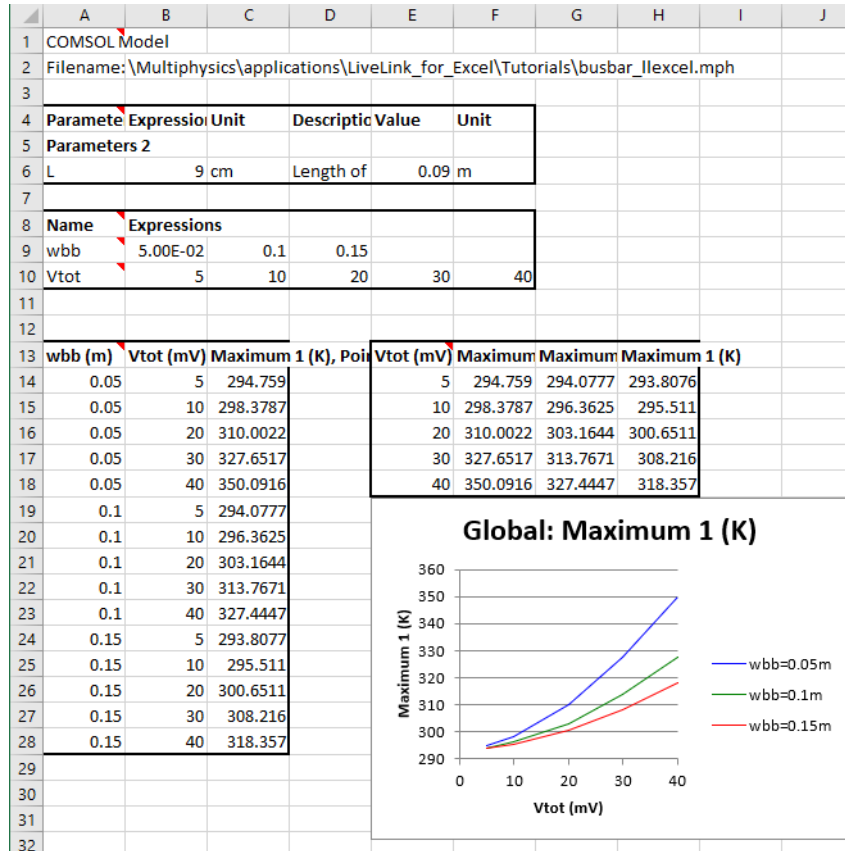
This example reuses the worksheet described in the section [Working with COMSOL® Models in Excel®](#) in the *Introduction to LiveLink™ for Excel® manual*. The steps below show how to write commands that automatically update the model with the settings in the worksheet, compute the solution and update the results data extracted in the worksheet.

When writing a subroutine in VBA the cell comment generated using the COMSOL toolbar are not usable. Enter the full command using the COMSOL API to reproduce the same behavior.

To get ready with the examples follow the steps below.

- 1 Start Excel, in the **File** tab, click **Open**.
- 2 In your COMSOL Multiphysics installation directory, find the folder `models/LiveLink_for_Excel/Tutorials`.

3 Select the file busbar_1l.xlsx and click the **Open** button.



You can see in cell range B2, the full name of the model MPH-file that is linked to the workbook. In the first worksheet you can assign value to model parameters and parameters sweep. It also contains the value of the maximum temperature in the busbar for the different parameter sweep values. The second worksheet contains the joule heating data interpolated at point coordinates defined in cell range A3 to C20.

4 Go to the Excel **File** menu and select **COMSOL**. In the COMSOL backstage view click **Open** button  and select **Open linked** . You will be asked if you want to replace the contents of the destination cell, click OK.

Note: When opening the linked model for this file, COMSOL will automatically search for the model busbar_llexcel.mph in the Application Library folder. If you do not have this model you need to specify the location. Find the folder applications/LiveLink_for_Excel/Tutorials. Select the file busbar_llexcel.mph and click the Open button.

The COMSOL Multiphysics Server starts and loads the model file busbar_llexcel.mph that is linked to the worksheet.

5 In Excel press Alt+F11 keys to open the VBA editor.

6 In the VBA editor, go to **Insert** menu and select **Module**.

7 In the editor type:

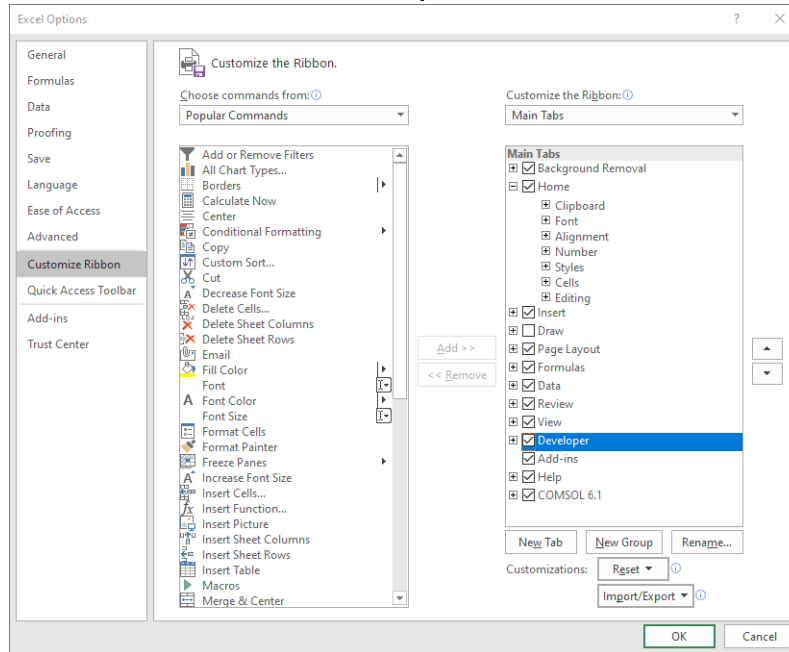
```
Sub busbarUpdate()
```

and press Enter key to get the end subroutine line automatically added to the script.

To run the macro a convenient way is to create a button. To proceed you need first to enable the Developer tab in the Excel ribbon.

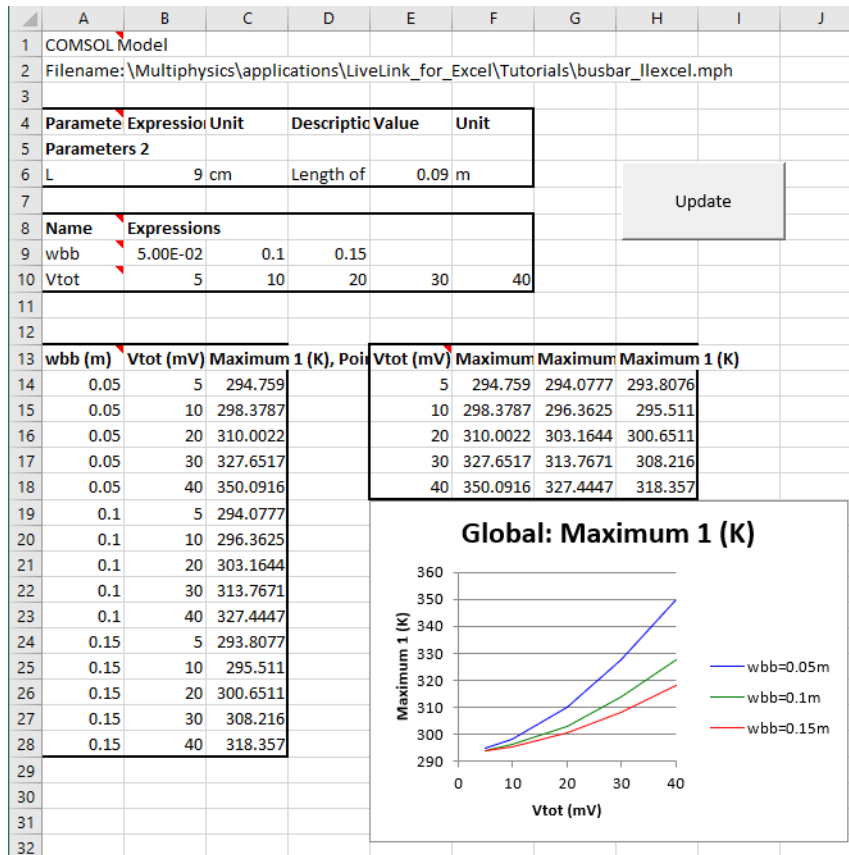
8 In Excel, go to the **File** tab and select **Options**.

- 9 In the **Excel Options** window, select **Customize Ribbon** and on the right side of the window under **Main Tabs** select **Developer**.



- 10 Click **OK**.
- 11 Go to the **Developer** tab and click **Insert**.
- 12 Under **Form Controls** select **Button (Form Control)** and draw the button in the worksheet.
- 13 In the **Assign Macro** window that just pops-up, under the **Macro name** list select **busbarUpdate**. Click **OK**.

- 14 You now have a button in the worksheet that runs the macro you will write in the example below. You can remove the inserted image and change the text in the button so that the worksheet looks like in the image below.



The worksheet including the full example code is available at
 XLSM-format in the LiveLink™ for Excel® Application Library
 LiveLink_for_Excel/Tutorials/busbar_llexcel.xlsm

In order to work with a COMSOL model it is necessary to use the class ModelUtil, which is part of the COMSOL API. In addition to ModelUtil there is a new object that is part of LiveLink™ for Excel®, which is called CmsolUtil. This object provides

extra utilities - such as the ability to start a COMSOL Multiphysics Server and create the RibbonUtil object where you can access the ribbon functionalities.

- 15** Make sure you have set the COMSOL references in the VBA project, in the Visual Basic for Applications window, under Tools select References. In the Available References list, select ComsolCom 6.2 and click OK.

Type the command at the line after the subroutine declaration line `Sub busbarUpdate()`.

- 16** Load the ComsolUtil, the ModelUtil, and the RibbonUtil objects.

```
Dim ModelUtil As ModelUtil
Dim ComsolUtil As ComsolUtil
Dim RibbonUtil As IRibbonUtil

Set ModelUtil = CreateObject("comsolcom.modelutil")
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
Set RibbonUtil = ComsolUtil.GetRibbonUtil
```

- 17** To allow long running jobs enter the command:

```
ComsolUtil.TimeOutHandler True
```

- 18** Open the linked model only if the ribbon is not connected to the COMSOL *server*.

```
If Not RibbonUtil.IsConnected Then
    RibbonUtil.OpenLinkedModel
End If
```

- 19** Set a link with the model, with the tag `Model`, available on the COMSOL *server*.

```
Set Model = ModelUtil.model("Model")
```

The cell range A6 to D6 define both the value and the unit for the model parameter L.

- 20** To update the model parameter defined in the cell range A4 enter:

```
Sheets("Sheet1").Activate
Range("A4").Select
RibbonUtil.UpdateDefinitions
```

The worksheet contains values for the busbar width (*wbb*) and the applied voltage (*Vtot*) as sweep parameter. In this example, the sweep parameters cannot have more than 5 value each.

- 21** Keep only the fifth first column for both sweep parameters:

```
Range("G9:J10").Clear
```

- 22** To update the parametric sweep parameters defined in the cell range A8 enter:

```
Range("A8").Select
RibbonUtil.Sweep "std1", , True
```

- 23** To enable the progress bar and compute enter:

```
ModelUtil.ShowProgress True  
Model.get_study("std1").Run
```

- 24** Display the third plot group and insert the corresponding image in L4.

```
Model.get_result("pg3").Run  
Range("L4").Select  
RibbonUtil.InsertGraphics "pg3"
```

- 25** Update all numerical results in sheet 1:

```
RibbonUtil.UpdateAllResults
```

In the step below, you will implement the code to perform the interpolation operation as it is currently done in Sheet2.

- 26** You need first to retrieve the value for each parametric sweep, that will be used to loop over when performing the interpolation evaluation.

```
Vtot = Range("B10:F10").Value  
For I = 0 To 4  
    If Not IsEmpty(Range("B9").Offset(, I).Value) Then  
        wbbLength = I + 1  
    End If  
    If Not IsEmpty(Range("B10").Offset(, I).Value) Then  
        VtotLength = I + 1  
    End If  
Next  
wbb = Range(Cells(9, 2), Cells(9, 2 + wbbLength)).Value  
Vtot = Range(Cells(10, 2), Cells(10, 2 + VtotLength)).Value
```

- 27** Clear Sheet 2, except the cell that contain the coordinates for the interpolation.

```
Sheets("Sheet2").Activate  
Range("D1:AB21").Delete
```

- 28** Loop over the parameter wbb and interpolate the total heat at the specified coordinates:

```
For I = 0 To wbbLength - 1  
    Range("D4").Offset(, I * VtotLength).Select  
    RibbonUtil.ResultsInterpolation "dset2", "ht.Qtot", "A4:C21",  
    , "wbb", wbb(1, I + 1)  
Next
```

- 29** Enter the commands below to improve the cell format:

```
Cells(1, 4) = "Qtot [W]"  
Cells(1, 4).Font.Bold = True  
Cells(1, 4).HorizontalAlignment = xlCenter  
Cell12 = 4 + VtotLength * wbbLength - 1  
Range(Cells(1, 4), Cells(1, Cell12)).Merge
```

```

For I = 0 To wbbLength - 1
    Idx = I * wbbLength
    Title = "wbb = " & wbb(1, I + 1) & "[m]"
    Cell1 = 4 + VtotLength * I
    Cell2 = 4 + (I + 1) * VtotLength - 1
    Cells(2, Cell1) = Title
    Cells(2, Cell1).Font.Bold = True
    Cells(2, Cell1).HorizontalAlignment = xlCenter
    Range(Cells(2, Cell1), Cells(2, Cell2)).Merge
    Range(Cells(2, Cell1), Cells(2, Cell2)).Borders.Weight =
xlThick
    For j = 1 To VtotLength
        Idx2 = I * VtotLength + j - 1
        Title = "Vtot = " & Vtot(1, j) & "[mV]"
        Range("D3").Offset(, Idx2).Value = Title
        Range("D3").Offset(, Idx2).Font.Bold = True
        Range("D3").Offset(, Idx2).Borders.Weight = xlThick
    Next
Next

```

- 30** You can now close the Microsoft Visual Basic for Applications window and test to run the subroutine by editing the model parameter L, cell B6, or the sweep parameters wbb and Vtot, cell B9 to F9 and B10 to F10, respectively. Click the Update button to update the workbook with the current solution.

Code for use with VBA

Below you find the complete VBA code; feel free to copy and paste it to the VBA editor.

```

Sub busbarUpdate()

    Dim ModelUtil As ModelUtil
    Dim ComsolUtil As ComsolUtil
    Dim RibbonUtil As IRibbonUtil

    Set ModelUtil = CreateObject("comsolcom.modelutil")
    Set ComsolUtil = CreateObject("comsolcom.comsolutil")
    Set RibbonUtil = ComsolUtil.GetRibbonUtil

    ComsolUtil.TimeOuthandler True

    If Not RibbonUtil.IsConnected Then
        RibbonUtil.OpenLinkedModel
    End If

    Set Model = ModelUtil.Model("Model")

    Sheets("Sheet1").Activate
    Range("A4").Select

```

```

RibbonUtil.UpdateDefinitions

Range("G9:J10").Clear

Range("A8").Select
RibbonUtil.Sweep "std1", , True

ModelUtil.ShowProgress True
Model.get_study("std1").Run

Model.get_result("pg3").Run
Range("L4").Select
RibbonUtil.InsertGraphics "pg3"

RibbonUtil.UpdateAllResults

Vtot = Sheets("Sheet1").Range("B10:F10").Value
For I = 0 To 4
    If Not IsEmpty(Sheets("Sheet1").Range("B9").Offset(, I).Value)
    Then
        wbbLength = I + 1
    End If
    If Not IsEmpty(Sheets("Sheet1").Range("B10").Offset(, I).Value)
    Then
        VtotLength = I + 1
    End If
Next
wbb = Sheets("Sheet1").Range(Cells(9, 2), Cells(9, 2 +
wbbLength)).Value
Vtot = Sheets("Sheet1").Range(Cells(10, 2), Cells(10, 2 +
VtotLength)).Value

Sheets("Sheet2").Activate
Range("D1:AB21").Delete

For I = 0 To wbbLength - 1
    Range("D4").Offset(, I * VtotLength).Select
    RibbonUtil.ResultsInterpolation "dset2", "ht.Qtot", "A4:C21",
, "wbb", wbb(1, I + 1)
Next

Cells(1, 4) = "Qtot [W]"
Cells(1, 4).Font.Bold = True
Cells(1, 4).HorizontalAlignment = xlCenter
Cell12 = 4 + VtotLength * wbbLength - 1
Range(Cells(1, 4), Cells(1, Cell12)).Merge
For I = 0 To wbbLength - 1
    Idx = I * wbbLength

```

```

    Title = "wbb = " & wbb(1, I + 1) & "[m]"
    Cell1 = 4 + VtotLength * I
    Cell2 = 4 + (I + 1) * VtotLength - 1
    Cells(2, Cell1) = Title
    Cells(2, Cell1).Font.Bold = True
    Cells(2, Cell1).HorizontalAlignment = xlCenter
    Range(Cells(2, Cell1), Cells(2, Cell2)).Merge
    Range(Cells(2, Cell1), Cells(2, Cell2)).Borders.Weight =
xlThick
    For j = 1 To VtotLength
        Idx2 = I * VtotLength + j - 1
        Title = "Vtot = " & Vtot(1, j) & "[mV]"
        Range("D3").Offset(, Idx2).Value = Title
        Range("D3").Offset(, Idx2).Font.Bold = True
        Range("D3").Offset(, Idx2).Borders.Weight = xlThick
    Next
Next

End Sub

```


Objects and Methods

The COMSOL API for VBA is using the same model structure and the same methods for accessing the model settings and data as the COMSOL API for Java[®]. Most of the methods of the COMSOL API are available in VBA, although some methods are renamed to comply with the Component Object Model (COM) requirements.

In this section:

- [The COMSOL API](#)
- [Renamed Methods](#)

The COMSOL API

COMSOL provides an API based on Java[®] that makes it possible to access COMSOL models settings and results. The COMSOL API consists on approximately 200 classes. These classes contain many methods each. All these methods are covered in the *COMSOL Programming Reference Manual*.

ADAPTING COMSOL API CODE IN VBA

The COMSOL Desktop supports saving model as JAVA-file, which is very convenient to learn the COMSOL API syntax. However, the code generated in the JAVA-file requires some adaption to be run in VBA. The difference between the code to be used with JAVA and with VBA consist in:

- Methods with the same name resulting in different operation have to be renamed, see the section [Renamed Methods](#)
- The keyword `Call` has to be used on every line calling a method in the COMSOL API when the method does not return anything.
- The `set` method in the COMSOL API does not return anything in VBA, while it returns a reference to the object being changed in Java.
- Methods that involve three- or four-dimensional arrays are not supported in VBA. These methods are `faceDDX`, `faceDX`, `faceFF1`, and `faceFF2`.

Renamed Methods

The COMSOL API can have methods with the same name resulting in different operations, for example the methods that can get and set properties. This is not

allowed by the COM requirements. To avoid conflicts, methods are renamed as described in this section.

METHODS THAT GET AND SET PROPERTIES

In the COMSOL API you can get and set properties with the same method name. In the COMSOL API for VBA the methods to get properties are appended the prefix `get_` to their name, while the name of the methods that set properties are using the same name as in the COMSOL API for Java.

For example, the method `author()` can return the author name of a node in the model using the syntax:

```
String author()
```

But it can also set the author's name of the node in the model using the syntax:

```
ModelEntity author(String name)
```

As a standard, the methods used for getting the value of a property is appended with the prefix `get_`. The methods used for setting the value of a property keep their name in the COMSOL API.

Thus to run the first method above in VBA you can type:

```
string get_author()
```

While to run the second method in VBA, use the same syntax:

```
ModelEntity author(string name)
```

Some of the methods that have been renamed in this way are:

```
string tag();  
string label();  
string comments();  
String author();  
string version();  
string model();
```

METHODS THAT RETRIEVE NODES

Methods that are used to retrieve nodes from the model tree can return a list of all available nodes, or can return a requested node. To avoid conflict in VBA, the method to retrieve a specific node is appended with the prefix `get_`, while the method that returns all available nodes is the same as in the COMSOL API for Java.

For example, the COMSOL API syntax to return a list that contains all the studies in the model is the following:

```
StudyList study()
```

You can type the same command in VBA to retrieve the list of available study nodes:

```
StudyList study()
```

The COMSOL API command to return the study specified by tag is:

```
Study study(String tag)
```

To do the same in the COMSOL API in VBA type instead:

```
Study get_study(String tag)
```

The following command in the COMSOL API sets the input selection of the mesh size node to a point:

```
model.mesh("mesh").feature("size").selection().geom("geom",0)
```

To do the same in VBA the feature method is appended with the get_ prefix:

```
Call model.get_mesh("mesh").get_feature("size").Selection.geom("geom",0)
```

METHODS THAT RETURN ARRAYS OR VECTORS

The method `getAdj()` returns an array or a vector depending on the number of arguments. The method that returns an array is renamed to `getAdj1` and the method that returns a vector is renamed into `getAdj2`.

A similar naming convention is applied to the method `getAdjOrient`. The method that returns an array is renamed to `getAdjOrient1`, and the method that returns a vector is renamed into `getAdjOrient2`.

Utility Methods

In addition to the COMSOL API for VBA, LiveLink™ *for* Excel® also provides two utility classes, ComsolUtil and RibbonUtil. In ComsolUtil you find some useful general purpose methods. The RibbonUtil class contains wrapper functions for the COMSOL API that correspond to the functionality of the buttons found on the **COMSOL** tab in Excel.

In this section:

- [Commands Grouped by Function](#)
- [Methods in the ComsolUtil Class](#)
- [Methods in the RibbonUtil Class](#)

Commands Grouped by Function

METHODS IN THE COMSOLUTIL CLASS

METHOD	PURPOSE
ConvertToDoubleMatrix	Convert the incoming object to an array of doubles in COMSOL API format.
ConvertToDoubleMatrixDecimal	Convert the incoming object to an array of doubles in COMSOL API format.
ConvertToDoubleVector	Convert the incoming object to a vector of doubles in COMSOL API format.
get_errormessage	Return error messages in model.
get_port	Return the server port number.
get_Version	Return the ComsolCom interface version.
GetRibbonUtil	Return IRibbonUtil object.
isGraphicsServer	Return graphics server status.
StartComsolServer	Start a COMSOL Multiphysics Server.
TimeOutHandler	Set timeout handler to Excel.

EDIT METHODS IN THE RIBBONUTIL CLASS

METHOD	PURPOSE
BreakAllLinks	Break all links to the model.
BreakLink	Break link between a cell and the model.
Connect	Connect the COMSOL ribbon to the server.
Disconnect	Disconnect the COMSOL ribbon from the server.
IsConnected	Connection status between the COMSOL ribbon and the server.
OpenLinkedModel	Start a COMSOL Multiphysics Server.
OpenModel	Open a model object from a MPH-file.
ReadModelLink	Read model link.

DEFINITIONS METHODS IN THE RIBBONUTIL CLASS

METHOD	PURPOSE
Parameters	Import model parameters.
Functions	Import selected model functions.

METHOD	PURPOSE
UpdateAllDefinitions	Update all model definitions links.
UpdateDefinitions	Update selected model definitions link.
Variables	Import model variables.

VIEW METHODS IN THE RIBBONUTIL CLASS

METHOD	PURPOSE
InsertGraphics	Insert graphics in the worksheet.
InsertGeometryGraphics	Insert geometry graphics in the worksheet.
InsertMeshGraphics	Insert mesh graphics in the worksheet.

STUDY METHODS IN THE RIBBONUTIL CLASS

METHOD	PURPOSE
CreateSweepData	Return ISweepData object.
Sweep	Export or update parametric sweeps.

RESULTS METHODS IN THE RIBBONUTIL CLASS

METHOD	PURPOSE
Removes COMSOL related comments from the selected cells.	Update all evaluated data and results tables.
CreateEvaluationSelection	Return IEvaluationSelection object.
EvaluationGroup	Evaluate and insert data from evaluation groups into a worksheet.
Export1DPlot	Export 1D plot data.
ResultsDerivedValue	Export derived values data.
ResultsParameters	Export results parameters.
ResultsParticleEvaluation	Export particle evaluation data.
ResultsPointEvaluation	Export point evaluation data.
ResultsTable	Export results table data.
ResultsRayEvaluation	Export ray evaluation data.
ResultsInterpolation	Export interpolated data.
UpdateAllResults	Update all evaluated data.

HELP METHODS IN THE RIBBONUTIL CLASS

METHOD	PURPOSE
OpenDocumentation	Open COMSOL Multiphysics documentation.
OpenHelp	Open LiveLink™ for Excel® help.

PREFERENCES METHODS IN THE RIBBONUTIL CLASS

FUNCTION	PURPOSE
GetRibbonPreferences	Return IRibbonPreferences object.
AutomaticParameterUpdate	Set automatic parameters update.

Methods in the ComsolUtil Class

Additionally to the COMSOL API, LiveLink™ for Excel® provides its own object ComsolUtil, which contains the methods to handle operation between the COMSOL server and Excel.

To make all COMSOL utility methods available using VBA, enter:

```
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
```

CONVERTTODOUBLEMATRIX

ConvertToDoubleMatrix(obj, transpose As Boolean) As Double() converts the incoming object to an array of doubles in COMSOL API format.

The input object obj is either a scalar or an array, with the type string, integer (long or short) or double. The returned array will have the start index as zero, as it is requested in the COMSOL API.

If the boolean transpose is true then the input array is transposed before the output is created.

Code for use in VBA

```
Sub Convert2DoubleMatrixExample()  
  
    ' Fill in data  
    Range("A1").Value = x  
    Range("B1").Value = y  
    Range("C1").Value = Z  
    Range("A2:C10").Value = 0  
    Range("A3").Value = 0.025  
    Range("A4").Value = 0.05  
    Range("B5").Value = -0.0125
```



```

Range("B8").Value = -0.025
Range("A5:A7").Value = Range("A2:A4").Value
Range("A8:A10").Value = Range("A2:A4").Value
Range("B6:B7").Value = Range("B5").Value
Range("B9:B10").Value = Range("B8").Value
' Reformat the data collected in the worksheet in a usable way for
' the COMSOL model
Dim coords() As Double
ReDim coords(0 To 2, 0 To 8)
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
coords = ComsolUtil.ConvertToDoubleMatrix(Range("A2:C10").Value,
True)
Range("E2:M4").Value = coords

End Sub

```

CONVERTTODOUBLEMATRIXDECIMAL

`ConvertToDoubleMatrixDecimal(obj, transpose As Boolean, dec As String) As Double()` converts the incoming object to an array of doubles in COMSOL API format.

The input object `obj` is either a scalar or an array, with the type string, integer (long or short) or double. The returned array will have the start index as zero, as it is requested in the COMSOL API.

If the boolean `transpose` is set to true then the input array is transposed before the output is created.

`dec` is a string that defines the decimal sign definition (usually ",", or "."). Decimal sign definition is important to specify when converting strings to values.

CONVERTTODOUBLEVECTOR

`ConvertToDoubleVector(obj) As Double()` converts the incoming object to a vector of doubles.

The input object `obj` may be a scalar or a vector. The input may be of the type string, integer (long or short) or double. The returned vector will have the start index as zero, which makes it possible to send such vectors to COMSOL when vectors have to be supplied as an argument to a method in the COMSOL API.

GET_ERRORMESSAGE

`get_errormessage() As String` returns error message.

This returns any error message that was created by `StartComsolServer()` or other methods in `ComsolUtil`.

GET_PORT

`get_port()` As Integer returns the port number of the COMSOL Multiphysics server.

To return the port number the Multiphysics server has to be started `StartComsolServer()` in the same module.

GET_VERSION

`get_Version()` As String returns the version of the ComsolCom interface.

The version of the ComsolCom interface does not include the build number as it is for the COMSOL version. The current ComsolCom version number is "6.2.0.0".

GETRIBBONUTIL

`GetRibbonUtil()` As IRibbonUtil returns the IRibbonUtil object to enable COMSOL ribbon utilities.

ISGRAPHICSSERVER

`isGraphicsServer()` As Boolean returns if the COMSOL Multiphysics server is started as a graphics server.

The method `isGraphicsServer` requires an established connection to a COMSOL server.

STARTCOMSOLSERVER

`StartComsolServer(usegraphics As Boolean)` As Boolean starts a COMSOL Multiphysics Server. The boolean `usegraphics` is true to start the graphics server.

The method `StartComsolServer` returns a boolean that indicates if the COMSOL server has started successfully or not, the value of the boolean being true or false respectively.

TIMEOUTHANDLER

`Timeouthandler(on As Boolean)` applies a timeout handler to Excel.

Set `on` to True to apply a timeout handler to Excel in order to prevent any timeout when performing long running tasks (such as starting a server or solving large models).

Methods in the RibbonUtil Class

COMSOL ribbon functionalities are also accessible using VBA commands. These commands help to manipulate the model, compute the solution and perform

postprocessing operations like showing results or extracting data to the worksheet. The VBA ribbon commands are accessible from the RibbonUtil object.

To make all COMSOL ribbon utility methods available using VBA, enter:

```
Set RibbonUtil = ComsolUtil.GetRibbonUtil()
```

This returns the RibbonUtil object that implements the VBA toolbar functionality methods.

GETRIBBONPREFERENCES

GetRibbonPreferences() As IRibbonPreferences returns the IRibbonPreferences object for handling ribbon preference operations.

AUTOMATICPARAMETERUPDATE

AutomaticParameterUpdate(on As Boolean) sets the automatic parameter update functionality; on is a boolean that specifies if the automatic update is turned on (true) or not (false).

AutomaticParameterUpdate is accessible from the RibbonPreferences object.

BREAKALLLINKS

BreakAllLinks() breaks all links to the model.

Removes all COMSOL related comments from the currently active worksheet.

BREAKLINK

BreakLink() breaks link to the model.

Removes COMSOL related comments from the selected cells.

CLEARANDEVALUATEALLRESULTS

ClearAndEvaluateAllResults() clears all currently linked results table entries and then evaluate all linked derived values.

CONNECT

Connect([hostname As String], [port As Integer], [username As String], [password As String]) connects the ribbon and the client for the API to a server. All arguments are optional where:

- hostname is the host name where the COMSOL *server* is running

- port is the port number the COMSOL *server* is listening to.
- username and password are the credentials requires when connected to a COMSOL *server* running on a different machine.

The Connect method also connect the ModelUtil object to access the API clients from a server.

CREATEEVALUATIONSELECTION

CreateEvaluationSelection() As IEvaluationSelection returns an IEvaluationSelection object to use in a point evaluation.

To set the parameter selection you can use one of the following approaches:

- evalSelection.SetLoopLevelIndices(level As long, int[] levels) sets the parameter selection using the loop level one based index level and the current level one based index levels.
- evalSelection.SetLoopLevelEvaluation(int[N-1][M-1] levelArray) sets the parameter selection using the integer array levelArray, where N is the number of loop levels in the solution and M the maximum number of current loop level indices to set.
- evalSelection.SetLoopLevelEvaluation(Array(int[] level1, int[] level2, ...)) sets the parameter selection using a dynamic array, where int1, int2,... are integer array that set the one based index for parameter 1, 2,..., respectively.

For transient studies you may want to evaluate point expression at interpolated time, use SetInterpolation to set the evalSelection object as in:

evalSelection.SetInterpolation(t As Double) where t is the time for evaluation.

See the section [ResultsPointEvaluation](#) from more information about point evaluation.

CREATESWEEPDATA

CreateSweepData() As ISweepData returns an ISweepData object that can be used in parametric sweep settings.

void ISweepData.AddParameterSweep(sweepType As String, pName As String, pValues) adds a parameter sweep. Provide an array of double values for the parameterValues argument.

DISCONNECT

`Disconnect()` disconnects the ribbon and API clients from a COMSOL *server*.

The COMSOL Multiphysics server also shuts down once the connection is broken.

EVALUATIONGROUP

`EvaluationGroup(tag As String)` evaluates and inserts the results from the evaluation group `tag` at the currently active cell.

EXPORT1D PLOT

`Export1DPlot(pgTag As String)` exports the 1D plot group `pgTag` data, and creates a line chart. The data are exported at the currently selected cell and the line chart is placed on the next cell available on the right side.

`void Export1DPlot(pgTag As String, featTag)` exports the data of the feature `featTag` of the 1D plot group `pgTag`, and create a line chart. `featTag` is either a string array or an object array.

`void Export1DPlot(pgTag As String, featTag, False)` exports the 1D plot data without creating the line chart.

`void Export1DPlot(pgTag As String, featTag,, plotPosition As String)` exports the 1D plot data and create a line chart at the position specified by `plotPosition`, which can either be "right"(default), "left", "bottom", or "top".

`void Export1DPlot(pgTag As String, featTag,, "custom", cellRange As String)` exports the 1D plot data and create a line chart at the cell `cellRange`.

Code for use in VBA

```
Sub Export1DPlotExample()  
  
    Set ModelUtil = CreateObject("comsolcom.modelutil")  
    Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
    Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
    Range("A1").Select  
    Set Model = RibbonUtil.OpenModel("busbar_1lexcel.mph")  
  
    ' Example 1: Export the 1D Plot data pg5 and create the chart  
    Range("A4").Value = "Export the 1D Plot data pg5 and create the  
    chart"  
    Range("A5").Select  
    RibbonUtil.Export1DPlot "pg6"  
  
    ' Example 2: Do not include the chart
```

```

Range("A20").Value = "Do not include the chart"
Range("A21").Select
RibbonUtil.Export1DPlot "pg6", , False

' Example 3: Include the chart below the data
Range("A28").Value = "Include the chart below the data"
Range("E29").Select
RibbonUtil.Export1DPlot "pg6", , , "bottom"

' Example 4: Set manually the location of the chart
Range("A50").Value = "Set manually the location of the chart"
Range("A51").Select
RibbonUtil.Export1DPlot "pg6", , , "custom", "A57"

End Sub

```

FUNCTIONS

Functions(*fTag* As String) imports the model function defined with the tag *fTag* at the currently active cell.

INSERTGRAPHICS

InsertGraphics(*pgTag* As String) inserts the graphics from the plot group *pgTag* at the currently active cell.

INSERTGEOMETRYGRAPHICS

InsertGeometryGraphics(*geomTag* As String) inserts the graphics of the geometry defined with the tag *geomTag* at the currently active cell.

INSERTMESHGRAPHICS

InsertMeshGraphics(*meshTag* As String) inserts the graphics of the mesh defined with the tag *meshTag* at the currently active cell.

ISCONNECTED

IsConnected() As Boolean returns the connection status between the COMSOL ribbon and the server: True when connected, else False.

OPENDOCUMENTATION

OpenDocumentation() opens COMSOL Multiphysics documentation.

OPENHELP

OpenHelp() opens LiveLink™ for Excel® help.

OPENLINKEDMODEL

`OpenLinkedModel()` As `ModelImpl` opens a mph model file from the model link in the active workbook and returns the Model object. The model link is updated when the model is opened.

`OpenLinkedModel(pwd As String)` As `ModelImpl` opens the password protected mph model file from the model link in the active workbook and returns the Model object. The model link is updated when the model is opened.

A model link is created and inserted in the active cell when the model is opened. If no connection is established for the ribbon a local COMSOL Multiphysics server is started and connected automatically.

If the model MPH-file is not found at the location specified in the cell comment, `OpenLinkedModel` searches in the same folder where the workbook is saved.

Example

```
Sub OpenLinkedModelExample()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
Set Model = RibbonUtil.OpenLinkedModel()  
  
End Sub
```

OPENMODEL

`OpenModel(filename As String)` As `ModelImpl` opens the model MPH-file `filename` and returns the Model object.

`OpenModel(filename As String, pwd As String)` As `ModelImpl` opens the password protected model MPH-file `filename` and returns the Model object.

A model link is created and inserted in the active cell when the model is opened. If no connection is established for the ribbon a local COMSOL Multiphysics server is started and connected automatically.

In case of the filename does not include the full path, the method `OpenModel` search for the model MPH-file in the following directories:

- the same folder as the workbook.
- the COMSOL application libraries root directory and its subfolders.

Code for use in VBA

```
Sub OpenModelExample()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
Range("A1").Select  
Set Model = RibbonUtil.OpenModel("busbar_llexcel")  
  
End Sub
```

PARAMETERS

`Parameters()` imports all parameters from the model to a linked field with upper-left corner at the currently active cell.

`Parameters(pGroupTag As String)` imports parameters in the model in the parameter node group to linked field with upper-left corner at the currently active cell.

`Parameters(pGroupTag As String, paramCaseTag As String)` imports parameters in the model in the parameter node group and case to linked field with upper-left corner at the currently active cell.

Code for use in VBA

The following example opens the model `busbar_llexcel` from the Application library. Then, if the current cell is empty, returns all model parameters. If not, returns the first group parameters. You can copy the code below and paste it into the VBA editor.

```
Sub ParametersExample()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
Set Model = RibbonUtil.OpenModel("busbar_llexcel")  
  
' Example 1: Import all parameters  
Range("A4").Value = "Import all parameters"  
Range("A5").Select  
RibbonUtil.Parameters  
  
' Example 2: Import the group parameters par2 only  
Range("A13").Value = "Import all parameters"  
Range("A14").Select  
RibbonUtil.Parameters "par2"  
  
End Sub
```


READMODEL LINK

`ReadModelLink()` As String returns the model file path found in the model link in the currently active worksheet.

If model link does not exist an empty string is returned.

Code for use in VBA

```
Sub ReadModelLinkExample()  
  
    Set ModelUtil = CreateObject("comsolcom.modelutil")  
    Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
    Set RibbonUtil = ComsolUtil.GetRibbonUtil  
    Range("A1").Select  
    Set Model = RibbonUtil.OpenModel("busbar_1lexcel")  
    Range("A5").Value = RibbonUtil.ReadModelLink  
  
End Sub
```

RESULTS DERIVED VALUE

`ResultsDerivedValue(dtag As String)` evaluates the derived values defined with the tag `dtag` and returns the data at the currently selected cell.

RESULTS INTERPOLATION

`ResultsInterpolation(dset As String, expr As String, coord As String)` evaluates an expression at an arbitrary location. Insert the linked field at the current cell.

The arguments consist in:

- `dset`, the solution dataset tag.
- `expr`, the expression to evaluate.
- `coord`, the cell range where the interpolation coordinates are defined.

`ResultsInterpolation(dset As String, expr As String, coord As String, header As Boolean)` evaluates an expression at an arbitrary location and specify whether to include the header or not. When the boolean `header` is set to `False` the header is not included. Default value for `header` is `True`.

`void ResultsInterpolation(dset As String, expr As String, coord As String, header As Boolean, coord As String, pName As String, pValue As String)` evaluates expression at arbitrary location. `pName`, a string to define the solution parameters name and `pValue`, its values to use for the evaluation.

Code for use in VBA

```
Sub ExampleInterpolation()

Set ModelUtil = CreateObject("comsolcom.modelutil")
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
Set RibbonUtil = ComsolUtil.GetRibbonUtil

Range("A1").Select
Set Model = RibbonUtil.OpenModel("busbar_llexcel")

Range("A4:C12").Value = 0
Range("A5").Value = 2.5e-2
Range("A6").Value = 5e-2
Range("A7:A9").Value = Range("A4:A6").Value
Range("A10:A12").Value = Range("A4:A6").Value
Range("B7:B9").Value = -1.25e-2
Range("B10:B12").Value = -1.25e-2

' Example 1: Data interpolation (T) at coordinates given in A4:C12
Range("A14").Value = "Data interpolation (T) at coordinates given
in A4:C12"
Range("A15").Select
RibbonUtil.ResultsInterpolation "dset1", "T", "A4:C12"

' Example 2: Data interpolation (T) at coordinates given in A4:C12,
' including header
Range("A25").Value = "Data interpolation (T) at coordinates given
in A4:C12, including header"
Range("A26").Select
RibbonUtil.ResultsInterpolation "dset1", "T", "A4:C12", True

' Example 3: Data interpolation (T) at coordinates given in A4:C12
' using the solution dataset dset2 and parameter wbb = 0.1
Range("A37").Value = "Data interpolation (T) at coordinates given
in A4:C12 using the solution dataset dset2 and parameter wbb = 0.1"
Range("A38").Select
RibbonUtil.ResultsInterpolation "dset2", "T", "A4:C12", True,
"wbb", "0.1"

End Sub
```

RESULTS PARAMETERS

`ResultsParameters()` import all results parameters from the model to a linked field with the upper-left corner at the currently active cell.

RESULTS PARTICLE EVALUATION

`ResultsParticleEvaluation(dset As String, pos As Boolean, vel As Boolean, expr As String, num As Long, [t As String])` evaluates an expression along particle trajectories. Insert the linked field at the current cell.

The following arguments are supported:

- `dset`, use to define the particle dataset tag.
- `pos`, use to specify whether to include the particle position coordinates (`true`) or not (`false`).
- `vel`, use to specify whether to include the particle velocity (`true`) or not (`false`).
- `expr`, use to define the expression to evaluate.
- `num`, use to define the number of particles to use for the evaluation.
- `t`, use to set the time step for evaluation.



The particle evaluation requires a license for the Particle Tracing Module.

Code for use in VBA

```
Sub ParticleEvalExample()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
Range("A1").Select  
Set Model = RibbonUtil.OpenModel("trapped_protons")  
  
' Example 1: Particle position at t = 3s  
Range("A3").Value = "Particle position at t = 3s"  
Range("A4").Select  
RibbonUtil.ResultsParticleEvaluation "part1", True, False, "", 1,  
Array(3)  
  
' Example 2: Particle position and velocity t = 3s  
Range("A6").Value = "Particle position and velocity t = 3s"  
Range("A7").Select  
RibbonUtil.ResultsParticleEvaluation "part1", True, True, "", 1,  
Array(3)  
  
' Example 3: Particle position, velocity, and the expression  
' cpt.mf1.normB at t = 3s
```

```

Range("A9").Value = "Particle position, velocity, and the
expression cpt.mf1.normB at t = 3s"
Range("A10").Select
RibbonUtil.ResultsParticleEvaluation "part1", True, True,
"cpt.mf1.normB", 1, Array(3)

' Example 4: Particle position at t = 0s and t = 3s
Range("A12").Value = "Particle position at t = 0s and t = 3s"
Range("A13").Select
RibbonUtil.ResultsParticleEvaluation "part1", True, False, "", 1,
Array(0, 3)

' Example 5
Range("A16").Value = "Particles position (two particles) at t = 3s"
Range("A17").Select
RibbonUtil.ResultsParticleEvaluation "part1", True, False, "", 2,
Array(3)

End Sub

```

RESULTSPOINTEVALUATION

`void ResultsPointEvaluation(dset As String, expr As String, ptInd, [evalSel As IEvaluationSelection])` evaluates an expression at a geometry point. Insert the linked field at the current cell.

The following arguments are available:

- `dset`, use to define the solution dataset tag.
- `expr`, use to define the expression to evaluate.
- `ptInd`, use to define the list of the point selection for the evaluation.
- `evalSel`, use to define the parameter selection.

The argument `evalSel` is optional, when it is not set the evaluation is performed for all parameters.



See the section [CreateEvaluationSelection](#) for more information about how to set the parameter/time for evaluation.

Code for use in VBA

The first example shows how to perform point evaluation. Copy the code and paste it in the VBA editor.

```
Sub PointEvalExample()
```

```

Set ModelUtil = CreateObject("comsolcom.modelutil")
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
Set RibbonUtil = ComsolUtil.GetRibbonUtil

Range("A1").Select
Set Model = RibbonUtil.OpenModel("busbar_llexcel")

' Example 1: Evaluate the temperature T at point 1 for parameter
' values in the dataset dset1
Range("A4").Value = "Evaluate the temperature T at point 1 for
parameter values in the dataset dset1"
Range("A5").Select
RibbonUtil.ResultsPointEvaluation "dset1", "T", Array(1)

' Example 2: evaluate the temperature T at points 1 and 5 for all
' solution steps in the dataset dset1
Range("A12").Value = "Evaluate the temperature T at points 1 and 5
for all solution steps in the dataset dset1"
Range("A13").Select
RibbonUtil.ResultsPointEvaluation "dset1", "T", Array(1, 5)

' Example 3: Evaluate the temperature T at point 1 for only the
' first parameter value in dataset dset1
Range("A20").Value = "Evaluate the temperature T at point 1 for only
the first parameter value in dataset dset1"
Range("A21").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
evaluationSelection.SetLoopLevelIndices 1, Array(1)
RibbonUtil.ResultsPointEvaluation "dset1", "T", Array(1),
evaluationSelection

' Example 4: Evaluate the temperature T at point 1 for the second
' and third values of parameter 1 (Vtot) and the first one of
' parameter 2 (wbb) in the dataset dset2
Range("A24").Value = "Evaluate the temperature T at point 1 for the
second and third values of parameter 1 (Vtot) and the first one of
parameter 2 (wbb) in the dataset dset2"
Range("A25").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
evaluationSelection.SetLoopLevelIndices 1, Array(2, 3)
evaluationSelection.SetLoopLevelIndices 2, Array(1)
RibbonUtil.ResultsPointEvaluation "dset2", "T", Array(1),
evaluationSelection

' Example 5: Evaluate the temperature T at point 1 for the second,
' third and fourth values of parameter 1 (Vtot) and the first one
' of parameter 2 (wbb) in the dataset dset2
Range("A29").Value = "Evaluate the temperature T at point 1 for the
second, third, and fourth values of parameter 1 (Vtot) and the first
one of parameter 2 (wbb) in the dataset dset2"

```

```

Range("A30").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
Dim intA(1, 2) As Long
intA(0, 0) = 2
intA(0, 1) = 3
intA(0, 2) = 4
intA(1, 0) = 1
evaluationSelection.SetLoopLevelEvaluation intA
RibbonUtil.ResultsPointEvaluation "dset2", "T", Array(1),
evaluationSelection

' Example 6: Evaluate the temperature T at point 1 for the second
' and third values of parameter 1 (Vtot) and the first one of
' parameter 2 (wbb) in the dataset dset2
Range("A35").Value = "Evaluate the temperature T at point 1 for the
second and third values of parameter 1 (Vtot) and the first one of
parameter 2 (wbb) in the dataset dset2"
Range("A36").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
evaluationSelection.SetLoopLevelEvaluation Array(Array(2, 3),
Array(1))
RibbonUtil.ResultsPointEvaluation "dset2", "T", Array(1),
evaluationSelection

End Sub

```

The second example illustrates how to perform a point evaluation at interpolated time. Copy the code and paste it in the VBA editor.

```

Sub PointEvalInterpTimeExample()

Set ModelUtil = CreateObject("comsolcom.modelutil")
Set ComsolUtil = CreateObject("comsolcom.comsolutil")
Set RibbonUtil = ComsolUtil.GetRibbonUtil

Range("A1").Select
Set Model = RibbonUtil.OpenModel("pid_control")
Set std1 = Model.get_study("std1")
std1.Run

' Example 1: Evaluate the concentration c at point 17 and at
' 0.125s using the solution dataset dset2
Range("A4").Value = "Evaluate the concentration c at point 17 and
at 0.125s using the solution dataset dset2"
Range("A5").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
evaluationSelection.SetInterpolation Array(0.125)
RibbonUtil.ResultsPointEvaluation "dset2", "c", Array(17),
evaluationSelection

```

```

' Example 2: Evaluate the concentration c at point 17 and at
' 0.125s and 2.37s using the solution dataset dset2
Range("A9").Value = "Evaluate the concentration c at point 17 and
at 0.125s and 2.37s using the solution dataset dset2"
Range("A10").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
evaluationSelection.SetInterpolation Array("0.125 2.37")
RibbonUtil.ResultsPointEvaluation "dset2", "c", Array(17),
evaluationSelection

' Example 3: Evaluate the concentration c at point 17 and at
' 0.125s and the second value of parameter 2 (k_P_ctrl) using the
' solution dataset dset2
Range("A16").Value = "Evaluate the concentration c at point 17 and
at 0.125s and the second value of parameter 2 (k_P_ctrl) using the
solution dataset dset2"
Range("A17").Select
Set evaluationSelection = RibbonUtil.CreateEvaluationSelection()
evaluationSelection.SetInterpolation Array(0.125)
evaluationSelection.SetLoopLevelIndices 2, Array(2)
RibbonUtil.ResultsPointEvaluation "dset2", "c", Array(17),
evaluationSelection

End Sub

```

RESULTS RAY EVALUATION

`void ResultsRayEvaluation(dset As String, pos As Boolean, vel As Boolean, expr As String, num As Long, [t As String])` evaluates an expression along ray trajectories. Insert the linked field at the current cell.

The following arguments are available:

- `dset`, the ray dataset tag.
- `pos`, use to specify whether to include the ray position coordinates (`true`) or not (`false`).
- `vel`, use to specify whether to include the velocity (`true`) or not (`false`).
- `expr`, the list of expression to evaluate.
- `num`, the number of rays to use for the evaluation.
- `t`, the time step for evaluation if available.



The ray evaluation requires a license for the Ray Optics Module.

Code for use in VBA

```
Sub RayEvalExample()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
Range("A1").Select  
Set Model = RibbonUtil.OpenModel("luneburg_lens")  
  
' Example 1: Ray position at t = 5.8374e-9s  
Range("A3").Value = "Ray position at t = 5.8374e-9s"  
Range("A4").Select  
RibbonUtil.ResultsRayEvaluation "ray1", True, False, "", 1,  
Array(5.8374e-9)  
  
' Example 2: Ray position and velocity at t = 5.8374e-9s  
Range("A6").Value = "Ray position and velocity at t = 5.8374e-9 s"  
Range("A7").Select  
RibbonUtil.ResultsRayEvaluation "ray1", True, True, "", 1,  
Array(5.8374e-9)  
  
' Example 3: Ray position, velocity, and evaluation of gop.rrel at t  
= 5.8374e-9s  
Range("A9").Value = "Ray position, velocity, and evaluation of  
gop.rrel at t = 5.8374e-9s"  
Range("A10").Select  
RibbonUtil.ResultsRayEvaluation "ray1", True, True, "gop.rrel", 1,  
Array(5.8374e-9)  
  
' Example 4: Ray position at t = 3.836e-9s and t = 5.8374e-9s  
Range("A12").Value = "Ray position at t = 3.836e-9s and  
t = 5.8374e-9s"  
Range("A13").Select  
RibbonUtil.ResultsRayEvaluation "ray1", True, False, "", 1,  
Array(3.836e-9, 5.8374e-9)  
  
' Example 5: Rays position (two rays) at t = 5.8374e-9s  
Range("A16").Value = "Rays position (two rays) at t = 5.8374e-9s"  
Range("A17").Select  
RibbonUtil.ResultsRayEvaluation "ray1", True, False, "", 2,  
Array(5.8374e-9)  
  
End Sub
```

RESULTSTABLE

ResultsTable(tblTag As String) exports the results table with the tag tblTag at the currently active cell.

SWEEP

`Sweep(stdTag As String, True)` exports the parametric sweeps from the study `stdTag` at the currently active cell. `study` is the study node tag from where to extract the parametric sweeps.

`Sweep(stdTag As String, True, False, sweepData As ISweepData)` exports parametric sweeps from the study `stdTag` using the sweep data information set in the object `ISweepData`.

`Sweep(stdTag As String, False, True)` updates parametric sweeps from the study `stdTag` using the sweep data information set in the currently active cell.

`Sweep(stdTag As String, False, True, sweepData As ISweepData)` updates parametric sweeps from the study `stdTag` using the sweep data information set in the object `ISweepData`.



See the section [CreateSweepData](#) for more information about how to set sweep data information.

Example

```
Sub SweepExample()  
  
Set ModelUtil = CreateObject("comsolcom.modelutil")  
Set ComsolUtil = CreateObject("comsolcom.comsolutil")  
Set RibbonUtil = ComsolUtil.GetRibbonUtil  
  
Range("A1").Select  
Set Model = RibbonUtil.OpenModel("busbar_llexcel")  
  
' Example 1: Export parametric sweep set in the model  
Range("A4").Value = "Export parametric sweep set in the model"  
Range("A5").Select  
RibbonUtil.Sweep "std1", True  
  
' Example 2: Export parametric sweeps using given parameter values  
Range("A9").Value = "Export parametric sweep using given parameter values"  
Set sweepData = RibbonUtil.CreateSweepData  
Dim param(0, 2) As Double  
param(0, 0) = 0.1  
param(0, 1) = 0.15  
param(0, 2) = 0.2  
sweepData.AddParameterSweep "param", "wbb", param  
Range("A10").Select  
RibbonUtil.Sweep "std1", True, , sweepData
```

```

' Example 3 :Update parametric sweeps using the parameter value
' linked in the current cell
Range("A14").Value = "Update parametric sweeps using the parameter
value linked in the current cell"
Range("E11").Value = 0.3
Range("A15").Select
RibbonUtil.Sweep "std1", , True

' Example 4: Update parametric sweeps using the parameter value
' linked in the cell A10 and modify parameter values for Vtot
Range("A19").Value = "Update parametric sweeps using the parameter
value linked in the current cell and given parameter values"
Range("E11").Value = 0.3
Set sweepData2 = RibbonUtil.CreateSweepData
Dim param2(0, 1) As Double
param2(0, 0) = 5
param2(0, 1) = 10
sweepData2.AddParameterSweep "stat", "Vtot", param2
Range("A10").Select
RibbonUtil.Sweep "std1", , True, sweepData2
Range("A20").Select
RibbonUtil.Sweep "std1", True

End Sub

```

UPDATEALLRESULTS

UpdateAllResults() updates all evaluated data in the current worksheet.

UPDATEDEFINITIONS

UpdateDefinitions() updates the model with linked definitions in the currently active cell.

UPDATEALLDEFINITIONS

UpdateAllDefinitions() updates the model with all linked definitions in the active worksheet.

VARIABLES

Variables() imports all variables from the model to linked field with upper-left corner at the currently active cell.

I n d e x

- A** Application Libraries window 14
- C** COMSOL tab 18
- D** Definitions group 21
 - documentation 13
- E** emailing COMSOL 15
 - exporting material data 54
- I** internet resources 13
- K** knowledge base, COMSOL 16
- M** material data
 - exporting 54
 - Material Export group 33
 - model object
 - methods 81
 - MPH-files 14
- N** Numerical Results group 24
- S** sequences of operations 81
 - Study group 23
- T** technical support, COMSOL 15
- V** View group 19
- W** websites, COMSOL 16
 - worksheets, working with 34

