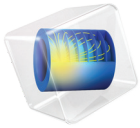


LiveLink™ *for* Simulink®

User's Guide

6.0

COMSOL

LiveLink™ *for* Simulink® User's Guide

Version: COMSOL 6.0

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 at 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/support/updates
- 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: CM024901

Contents

Chapter 1: Introduction

Chapter 2: Getting Started

Chapter 3: Cosimulation

Chapter 4: Reduced-Order Models

Chapter 5: Programming and Command Reference

1

Introduction

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

In this chapter:

About This Product

LiveLink™ *for* Simulink® connects COMSOL Multiphysics® to the Simulink® simulation environment that is an add-on to the MATLAB® technical computing software. Using this functionality you can do the following:

- Perform cosimulation of COMSOL models together with Simulink diagrams. Any time-dependent or static COMSOL model can be used for cosimulation.
- Export state-space models based on COMSOL models. These state-space models are linearized models based on the total number of degrees of freedom (DOFs) for the COMSOL model.
- Export reduced-order models based on COMSOL models. These state-space models are linearized models based on a modal order reduction of the COMSOL model. For some physics it is required to have additional modules in order to be able to set up the modal study that is required to perform the modal analysis.
- In addition, it is possible to use regular COMSOL and Simulink features to set up additional types of models, for example, models based on lookup tables in Simulink.

Additional Functionality with LiveLink™ for MATLAB®

LiveLink™ *for* MATLAB® provides an interface between MATLAB® and a COMSOL Multiphysics server to enable creating, editing, and running models in the MATLAB scripting environment. It also includes functions to extract data from a model to make it available in the MATLAB workspace, and to create MATLAB figures based on plot groups from a model.

LiveLink™ *for* Simulink® and LiveLink™ *for* MATLAB® do not depend on each other, but you can easily use them together to enhance your modeling experience.

Help and Documentation

In this section:

Getting Help

COMSOL Multiphysics® and LiveLink™ *for* Simulink® have several sources of help and information.

THE INTRODUCTION TO LIVELINK™ FOR SIMULINK®

To get started with LiveLink™, it is recommended that you read the *Introduction to LiveLink™ for Simulink®*. It contains detailed examples about how to get you started with the product.

THE APPLICATION LIBRARIES WINDOW

Study the LiveLink™ for Simulink® *Application Library*

LiveLink™ *for* Simulink® includes an Application Library with detailed tutorials.

If you have installed the COMSOL apps in the MATLAB Apps ribbon, click the COMSOL Application Libraries icon ().

The following are some models that can help you get started.

Tutorial Models

- *Battery Pack Discharge Control with Thermal Analysis* — This model computes the temperature distribution in a battery pack that is used at a given power. The current is controlled in Simulink to ensure constant power during usage.
- *Control of an Inverted Pendulum* — In this model, you will see how to control the base position of an inverted pendulum to keep its vertical position stable. The control is performed using a PID controller. An external force is applied at a base position depending on the rotation angle of the pendulum to prevent its fall. Moreover, the position of the pendulum is constrained within a range.
- *1D Lithium-Ion Battery Model Charge Control* — This model illustrates the charge/discharge control of a lithium-ion battery in a Simulink simulation.

- *Magnetic Brake — LiveLink™ for Simulink® Simulation* — This tutorial demonstrates how to use a stationary problem solved within a simulation diagram in Simulink.
- *On/Off Control of a Thermal Actuator* — This tutorial illustrates how to implement an on/off control system combined with a multiphysics analysis in COMSOL.

Terminology

This documentation uses the following terminology:

- *communication step*: the interval from t to $t+dt$ that is simulated in COMSOL at the communication time t.
- *communication time*: the time at which the programs exchange input and output data.
- *diagram*: a Simulink diagram.
- *model*: a COMSOL Multiphysics model.

Where Do I Access the Documentation and the Application Libraries?

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.

Mac

Linux

To open the **Help** window:

- In the **Model Builder** or **Physics Builder** click a node or window and then press F1.
- On the main toolbar, click the **Help** (?) button.
- From the main menu, select **Help>Help**.

Opening the Documentation Window

Win

To open the **Documentation** window:

- Press Ctrl+F1.
- From the **File** menu select **Help>Documentation** ( ).

Mac

Linux

To open the **Documentation** window:

- Press Ctrl+F1.
- On 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.

The Application Libraries Window in the *COMSOL Multiphysics Reference Manual.*

Opening the Application Libraries Window

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

Win

- From the **Home** toolbar, **Windows** menu, click ( ) **Applications Libraries**.
- From the **File** menu select **Application Libraries**.

To include the latest versions of model examples, from the **File>Help** menu, select ( ) **Update COMSOL Application Library**.

Mac

Linux

Select **Application Libraries** from the main **File>** or **Windows>** menus.

To include the latest versions of model examples, from the **Help** menu select ( ) **Update COMSOL Application Library**.

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/support/updates
COMSOL Blog	www.comsol.com/blogs
Discussion Forum	www.comsol.com/community
Events	www.comsol.com/events
COMSOL Video Gallery	www.comsol.com/video
Support Knowledge Base	www.comsol.com/support/knowledgebase

2

Getting Started

This chapter provides an introduction to run cosimulation using LiveLink™ *for* Simulink®.

In this chapter:

Starting COMSOL with Simulink

In this section:

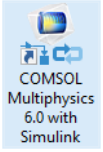
Starting COMSOL® with Simulink®

In order to use LiveLink™ *for* Simulink® the MATLAB® environment has to be set up in a way that LiveLink for Simulink can find its required files when running. This requirement means that MATLAB has to be started using one of the methods mentioned in the following subsections.

	Manual set up of the connection between Simulink and COMSOL is not supported.

STARTING ON WINDOWS®

On Windows® use the **COMSOL Multiphysics 6.0 with Simulink** shortcut icon that is created on the desktop during the product installation. A link is also available in the Windows start menu under **COMSOL Multiphysics 6.0>COMSOL Multiphysics 6.0 with Simulink**.



This opens the MATLAB desktop together with the COMSOL Multiphysics Server, which is represented by the command window appearing in the background. Note that the Simulink window does not open automatically. You have to click on the Simulink button on the MATLAB toolbar to open the Simulink window.

STARTING ON MACOS

On macOS, use the **COMSOL Multiphysics 6.0 with Simulink** application available in the **Application** folder: **Applications>COMSOL 6.0>Multiphysics**.

Alternatively, use the command mentioned in the section Starting on Linux®.

STARTING ON LINUX®

On Linux®, enter the command `comsol mphserver simulink` at a terminal.

Changing the Connected MATLAB® Installation

The path of the MATLAB® installation connected to COMSOL Multiphysics is defined during the initial COMSOL installation. To change the MATLAB root path you can either use the COMSOL 6.0 installer or set the MATLAB path in the startup command.

CHANGING THE MATLAB VERSION USING THE INSTALLER

1. Run the COMSOL Installer set up.
2. In the **COMSOL Multiphysics 6.0 Installer** select **Add/Remove Products and Reinstall**.
3. Select the appropriate existing COMSOL 6.0 installation folder.
4. Click **Next** until you reach the **LiveLink** section.
5. Set the MATLAB root directory path in the **MATLAB® installation folder** field, then click **Next**.
6. Finally click **Install**.

CHANGING THE MATLAB VERSION IN THE LAUNCHER

- On Windows®, right-click the **COMSOL Multiphysics 6.0 with Simulink** shortcut icon, and select Properties. In the **COMSOL Multiphysics 6.0 with Simulink Properties** window edit the **Target** field with the following:

  `<COMSOL_ROOT>\bin\win64\comsolmphserver.exe simulink -mlroot <MATLAB_ROOT>`

  where `<COMSOL_ROOT>` is your COMSOL installation root directory and `<MATLAB_ROOT>` is the MATLAB installation root.

- On macOS and Linux®, at a terminal enter the command

  `comsol mphserver simulink -mlroot <MATLAB_ROOT>`

  where `<MATLAB_ROOT>` is the MATLAB installation root.

Updating the Simulink System

Simulink systems that contain COMSOL Cosimulation blocks generated in a previous version needs to be updated. At the MATLAB prompt use the command `mphupdatesystem` as in the command below:

```
mphupdatesystem(<system>)
```

where *<system>* is the name of the Simulink system to be updated. This will update all the COMSOL Cosimulation FMU-file used in the system so that they can run with COMSOL Multiphysics 6.0, as well as all the COMSOL Cosimulation blocks. Moreover backup files for both Cosimulation file and Simulink system are saved with the suffix "versionUpdateBackup".

In case the Simulink system contains subsystems with COMSOL Cosimulation blocks, these subsystems have to be updated separately.

If a previous backup already exists you can set the property `overwritebackup` to `on` as shown below:

```
mphupdatesystem(<system>, 'overwritebackup', 'on')
```

this will overwrite the current backup to the newly updated system.

To not save a backup set the property `createbackup` to `off`:

```
mphupdatesystem(<system>, 'createbackup', 'off')
```

If you want to run the updated system, you can open it Simulink directly after the update, to proceed set the property `postupdateaction` to `open`:

```
mphupdatesystem(<system>, 'postupdateaction', 'open')
```

Note that the system SLX-file is not saved.

`mphupdatesystem` automatically search through the COMSOL installation directory for the COMSOL Cosimulation block library path. If needed you can specify it manually:

```
mphupdatesystem(<system>, 'librarypath', <path>)
```

A correct library path should point to the file `comsol`*<ver>*`lib.slx`, with *<ver>* the COMSOL version number.

To revert the Simulink system using a older COMSOL version, set the property `version` with desired version number:

```
mphupdatesystem(<system>, 'updateversion', <ver>)
```

where `<ver>` is the version number. Currently the only previous version supported is `'5.6'`.

You can also specify to update the COMSOL Cosimulation block and/or the COMSOL Cosimulation FMU-file using the properties `updateblock` and/or `updatefmu`, respectively. Both are updated by default.

Note for Floating Network License Users

LiveLink™ *for* Simulink® does not depend on LiveLink™ *for* MATLAB®, but the two products work together. If both products are installed, you can choose to start either product and work seamlessly with the LiveLink for MATLAB wrapper functions and the Simulink cosimulation environment. If a floating network license (FNL) is used, it may be important which product is started first, because a license for the first started product is checked out from the license server. For example, if you first start LiveLink for Simulink as described in Starting COMSOL® with Simulink® you will check out a license for LiveLink for Simulink. If you then later in that session use functionality that requires a license for LiveLink for MATLAB, a license for LiveLink for MATLAB will also be checked out at that time.

About the Model Object

The model object contains all the information about a model, from the geometry to the results. The model object is defined on the COMSOL Multiphysics Server and can be accessed using the COMSOL Cosimulation block in Simulink or at the MATLAB command line using a link in MATLAB.

Cosimulation

Cosimulation is a method of integrating two or more independent simulation software programs so that they can perform a joint simulation. Data is then transfered between the two programs as needed. Usually one of the programs is in charge of the overall simulation and decides when each program should communicate with the other programs. The points in time when data is exchanged is referred to as communication points or times, and the interval between these times is the communication step. The other programs must then be able to perform the simulation of a (usually small) communication step. The programs performing the simulation may use their own time-stepping algorithms and internal time stepping.

When performing cosimulation of a COMSOL model with Simulink in the described way, the two simulation environments perform the time stepping one after the other. Simulink is in charge of the overall time stepping and controls how and when the simulations of all the blocks in the Simulink diagram are performed.

When Simulink determines that the COMSOL cosimulation block should be simulated during a communication step t to $t + dt$, the following steps take place:

1. Simulink sends new input values to the COMSOL model.
2. Simulink notifies COMSOL that a simulation from t to $t + dt$ has to be performed.
3. COMSOL finds consistent initial conditions based on the changed input parameter values and performs the simulation.
4. COMSOL sends output values from the model when the simulation of the communication step has been completed.

It is important to note that the simulation of the COMSOL model is performed by the COMSOL solvers defined in the model. Hence the COMSOL solvers determine what type of time stepping is performed internally in COMSOL during the communication step t to $t + dt$. Often COMSOL will choose to take small time steps within the communication step in order to achieve the desired accuracy.

State-Space Systems

State-space system are linear dynamic system with input u and output y defined as in the equation below:

$$\begin{cases} \frac{dx}{dt} = Ax + Bu \\ y = Cx + Du \end{cases}$$

where x is the state variable vector.

An alternative representation of the above dynamic system is:

$$M_C\dot{x} = M_CAx + M_CBu$$
$$y = Cx + Du$$

The later form is more suitable for large systems because the matrices M_C and M_CA usually become much more sparse than A.

In COMSOL you can choose two approaches to assemble the state-space matrices Mc, MA, MB, C, and D. You can either choose to linearize your system using a reduced-order model, or assemble the linearized matrices from the full model.

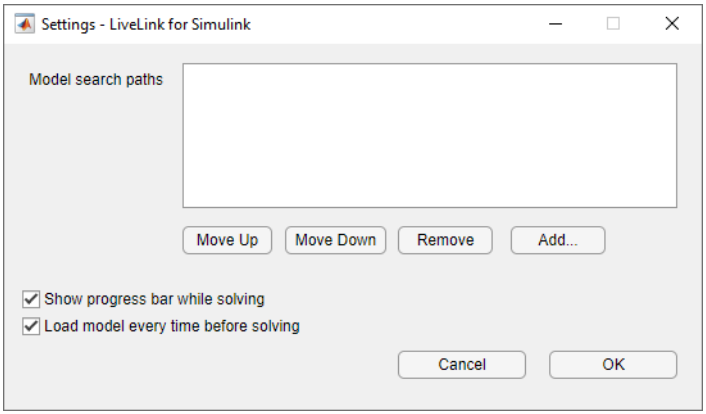
Model reduction has the advantage of returning reduced system matrices, which can be useful when working with a model that has many degrees of freedom because the model reduction reduces the number of states that are necessary to capture the behavior of the dynamic input/output relationship. However, as it is based on a modal solution, model reduction is restricted to physics interfaces that support eigenvalue analysis. Multiphysics problems may not be suited for reduced-order models. The assembly operation of the state-space matrices has to be done within the COMSOL model. LiveLink™ *for* Simulink® comes with the function `mphreduction` to extract the desired matrices in the MATLAB environment.

Directly extracting state-space matrices for the full-order system is an alternative to reduced-order modeling because the state-space system is not restricted to a specific physics interface and can be used with multiphysics problem as long as you use a suitable linearization point. The resulting matrices have the size of the number of degrees of freedom of the model, which may lead to very large matrices. The assembly operation of the state-space matrices has to be done within the COMSOL model. To

extract the matrices in MATLAB, you can use the function `mphstate`, which is included with LiveLink™ *for* MATLAB®.

Preferences

To change the settings related to the cosimulation open the **Settings — LiveLink for Simulink** window by running the command `mphsimsettings` in the MATLAB command window.



The sections below describe the available settings. To accept any changes to the settings and close the window, click the **OK** button. Click **Cancel** to close the window and discard any changes that you have made since opening it.

MODEL SEARCH PATHS

A number of path definitions can be added to the list of search paths such that COMSOL can find the COMSOL model that is referenced in the cosimulation file.

When the simulation is performed, Simulink does not notify COMSOL where the cosimulation file is placed. Hence some bookkeeping has to be performed in order to keep track of locations.

If the COMSOL model file (MPH) is placed in the same directory as the cosimulation file, COMSOL will be able to find the COMSOL model file automatically. This is the recommended procedure. Every time a simulation is set up, the path of the model file is added to the **Model search path** list such that COMSOL can find the file.

This list can become very long and it may be important to remove the paths that are not required anymore. Use the **Remove** button to remove any unwanted paths from the list.

If it is not possible to place the model file in the same location as the cosimulation file, the path of the model file must be added to the **Model search path** list manually. You can do this by clicking the **Add** button and selecting the desired folder.

The search order is important in case there are models with the same filename. The path that is higher up in the list will be used for the cosimulation. Use the buttons **Move Up** and **Move Down** to reorder the paths in the list. It is recommended to avoid saving models with identical filenames to different locations. If this is not possible, then, in the **Model search path** list, keep only the path of the model to run in the cosimulation.

SHOW THE PROGRESS BAR WHILE SOLVING

In order to verify that the simulation is progressing as planned, you can display the COMSOL progress bar when performing the cosimulation. To do this, select the **Show progress bar while solving** check box. The COMSOL progress bar will display the progress of the various steps of the solver, and you can see in case there are any problems during the solution process.

In the event that COMSOL gets stuck during a time step, you can also cancel the COMSOL simulation from the progress bar.

LOAD MODEL EVERY TIME BEFORE SOLVING

After a simulation is run the model is reset to its original state that corresponds to the state when it was first loaded. In the event that a model is using features that cannot reliably be set back to their original state, select the **Load model every time before solving** check box. With this setting the model is loaded every time the simulation starts. Note that loading the model can be time consuming for large models.

3

Cosimulation

To prepare a cosimulation between COMSOL Multiphysics and Simulink, you need to export the cosimulation file for the COMSOL Cosimulation block used in Simulink. The export is performed from the COMSOL model either using the COMSOL Desktop or the COMSOL API.

In this chapter:

The Cosimulation Workflow

In this section:

Before You Start

When setting up a COMSOL model for cosimulation observe the following requirements and recommendations:

- Studies to be used with cosimulation are only allowed to contain a single study step, which may be either a time-dependent or a stationary study step. It is most common to have a time-dependent study when using a model for cosimulation.
- Auxiliary sweep in time-dependent and stationary study step, as well as load cases for a stationary study step, are not supported in cosimulation.
- Adaptation, like adaptive mesh refinement or adaptation and error estimates, are not supported in cosimulation.
- In order to use the model for cosimulation the model must be saved with a filename, before exporting the cosimulation file. This is important since the name of the model is needed to load the cosimulation file in Simulink.
- Cosimulation becomes easier to use if you save the model in the same directory as the exported cosimulation file.
- Make sure that the model solves with the time interval and time steps that will be used when performing the cosimulation.
- Verify that the model solves with the intended input parameter range that will be used for the cosimulation.
- The time unit for Simulink is assumed to be seconds. There is no check that the time units in COMSOL and Simulink are the same.
- Unit conversion for inputs and outputs is not performed. Hence, all values that are exchanged are in the unit of the COMSOL model. When using probes to define output values the probe contains a setting for the unit that is used for the transfer.

- When performing cosimulation with Simulink, the parameters are changed abruptly at discrete communications points. In some cases, such abrupt changes make time stepping more challenging for the solvers compared to just performing the simulation within COMSOL.
- Solver settings may have to be updated to take into account the change of input values. Manual scaling is often advised if a parameter directly controls a field variable.
- The physics definitions in the model may have to be updated to describe the dynamics of the system accurately. This is especially important when converting a model from solving using a static solver to a time-dependent solver. For example, you may have to introduce damping or other material settings that affect the dynamics of the system.

The Cosimulation Workflow

Follow the workflow below to set up the cosimulation with COMSOL Multiphysics and Simulink:

1. Set up and save the COMSOL model and make sure that the study intended for the cosimulation runs. For requirements and recommendations on the COMSOL model see Before You Start.
2. Add the Cosimulation for Simulink feature node to the COMSOL model. Use this to define the inputs, outputs, and study for the cosimulation, see Configuring and Exporting the Cosimulation File.
3. From the Cosimulation for Simulink feature node export the *cosimulation file*. Any location will work, but it is good practice to export this file to the location where the MPH-file has been saved.
4. Create or load the simulation diagram in Simulink, and add the COMSOL Cosimulation block, see Adding a Cosimulation Block in Simulink.
5. Double-click the COMSOL Cosimulation block, and enter the name of the cosimulation file exported from COMSOL Multiphysics.
6. Run the cosimulation in Simulink.

Optionally save the model and postprocess it in the COMSOL Desktop, see After the Cosimulation.

Handling Study with Several Study Steps

Some studies requires that you have to run several study steps in sequence. As cosimulation only support one study step to export your model you will need to split the study in order to isolate the supported study step for cosimulation. Note that this approach only works when each study steps are run the one after the other and when the study step to run in cosimulation is the last study step in the sequence. In the case the study contains a sweep study step, this has to be disable from the study.

The steps below describes the procedure to follow:

1. In the Study you want to run in cosimulation, right-click the Time Dependent or Stationary study step and select Copy. This way you will be able to paste the study step in a separate study using the exact same settings. This Time Dependent or Stationary study step will run alone and so available for cosimulation.
2. Now right-click again the same study step and select Disable.
3. Run the study.
4. Add an empty study to the model.
5. Right-click the empty study node and select paste. You now have only the study step to run in cosimulation in the new study.
6. In the study step, expand the section **Values of Dependent Variables**.
7. Under **Initial values of variables solved for** in the **Settings** list select **User controlled**.
8. In the **Method** list select **Solution**.
9. In the **Study** list select the study from which you have copied the study step.
10. **Under Values of variables not solved for**, in the **Settings** list select **User controlled**.
11. In the **Method** list select **Solution**.
12. In the **Study** list select the study from which you have copied the study step.
13. Run the study.
14. You can now select the newly generated study in the Cosimulation for Simulink node.

Configuring and Exporting the Cosimulation File

This chapter describes the Cosimulation for Simulink node from where you export the cosimulation file.

In this section:

The Cosimulation for Simulink Node

To add a **Cosimulation for Simulink** node, go to the **Study** tab, or right-click **Global Definitions**, and select **Cosimulation for Simulink**.

You can add more than one export node if you have different scenarios that you want to simulate. The exported cosimulation file only contains information about how Simulink should interface to the COMSOL model. The cosimulation file itself does not contain the model, which remains solely in the COMSOL MPH-file.

In the **Filename** section, enter the name of the cosimulation file to export to. Enter the full name, including the path, if you want to save the file in a specific location; otherwise, the file is saved at the same location as the model MPH-file.

	The cosimulation file and the application MPH-file do not need to share the same location, even if it is recommended.

A proper cosimulation file is set with at least one output expression and a supported study step.

Click **Export** to generate the cosimulation file in FMU-format.

	Save the application MPH-file first before exporting the cosimulation file to set the model name.

	Simulink does not have to be installed on the computer that performs the export of the cosimulation file. Both Simulink and COMSOL Multiphysics are required for running the cosimulation.

THE INPUTS SECTION

In the **Inputs** section you specify the input variables and their order in the COMSOL Cosimulation block in Simulink. The input variables can be modified and updated in the model at every communication time. In the **Parameter name** column, the list contains all global parameters defined in the model. This list is automatically updated with respect to the block parameters settings, as a global parameter can either be set as input or block parameter.

In the **Initial value** cell enter the value to be set at initial time step.

To add a new input, click **Add**. To remove an input, select a row and click **Delete**.

In case you have several inputs, their order in the input list correspond to the order for the COMSOL Cosimulation block. To improve the visibility of the simulation diagram you may want to change the order of input. To proceed, select the row to move and click **Move Up** or **Move Down**.

Click **Clear Table** to clear all parameter from the input list.

If you want to save or import a list from an external file, click **Save to File** or **Load from File**, respectively.

THE BLOCK PARAMETERS SECTION

In the **Block Parameters** section you specify the parameters you want to change in Simulink but which are only evaluated during the initialization of the cosimulation. In the **Parameter name** column choose one of the parameters from the list. This list is automatically updated with respect to the inputs settings, as a global parameter can either be set as input or block parameter. There is no restriction on which settings are defined by Block Parameters.

	A global parameter can either be set as Inputs or Block Parameters.

In the **Initial value** cell enter the default value in the COMSOL Cosimulation block.

To add a new block parameter, click **Add**. To remove a block parameter, select a row and click **Delete**.

Click **Clear Table** to clear all parameter from the input list.

If you want to save or import a list from an external file, click **Save to File** or **Load from File**, respectively.

THE OUTPUTS SECTION

In the **Outputs** section you specify the output expression and their order in the COMSOL Cosimulation block in Simulink.

	At least one output is required to export the cosimulation file.

In case you have several output, their order in the output list correspond to the order for the COMSOL Cosimulation block. To improve the visibility of the simulation diagram you may want to change the order of output. To proceed, select the row to move and click **Move Up** or **Move Down**.

Click **Clear Table** to clear all parameter from the input list.

If you want to save or import a list from an external file, click **Save to File** or **Load from File**, respectively.

THE STUDY SECTION

In the study section you specify the study to run in the cosimulation and how the solution should be stored in the COMSOL model.

From the **Study** list, select the study to run in the cosimulation. This list is populated with studies that only contain a single study step, either a Stationary or a Time Dependent study step.

	A study step is does not necessary correspond to a solver type, for instance Parametric Sweep node, Optimization node, etc... See the section Handling Study with Several Study Steps to learn how to split a study in order to enable cosimulation.November 2021

In addition, the list does not contain studies that include auxiliary sweep, time-dependent study steps with adaptive mesh refinement, nor stationary study steps including load cases or adaptation and error estimates.

If you select a time-dependent study, you can then specify how to store the solution in the COMSOL model. The solution at the communication points is always stored. To store only the solution once the cosimulation step is computed, locate the list **At communication points** and select **End of communication step**. If you want the solution at the beginning and the end of the communication step, select **Start and end of communication step**. This last option can be useful if you want to evaluate the difference in the computed solution when the input changes at the communication point.

In case the communication steps are large compared to the variation of the solution in COMSOL, you may want to store extra steps in the COMSOL model. Select **According to study step settings** to store the solution at the time step defined in the study step, for instance output time step or steps taken by solver.

THE DEPENDENCIES SECTION

The dependencies section is only visible for Time Dependent study.

Specify any direct dependencies between input parameters and output expressions using the radio buttons. The information is included in the exported cosimulation file in order to notify Simulink about the direct feedthrough in the model.

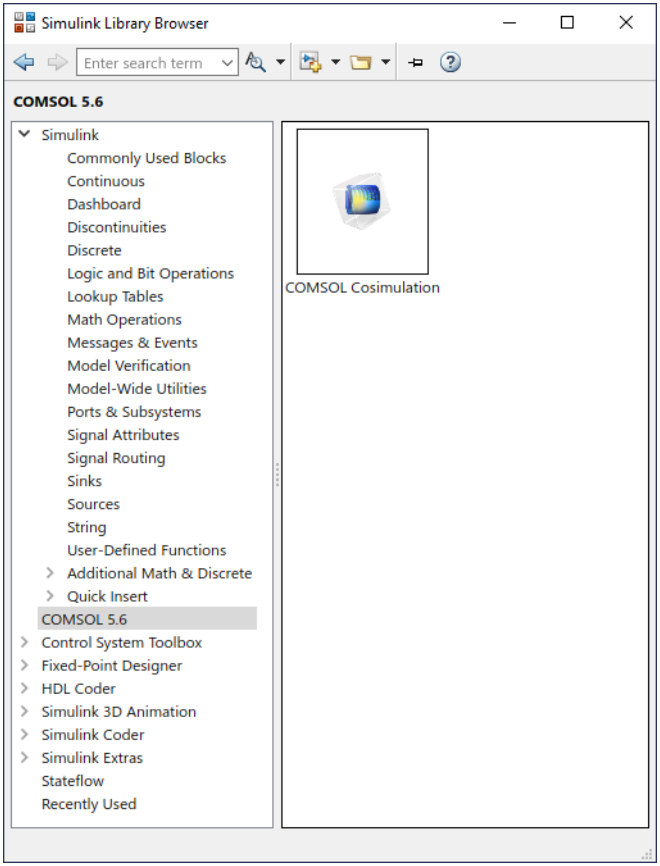
THE IMAGE SECTION

To easily recognize the model when loaded into the COMSOL Cosimulation block in Simulink you can select an image. Browse to select the image file to import, or click **Set from Graphics Window** to use a screenshot of what is currently displayed in the Graphics window.

In case you did specify a filename, click **Import**, to embed the image in the COMSOL model.

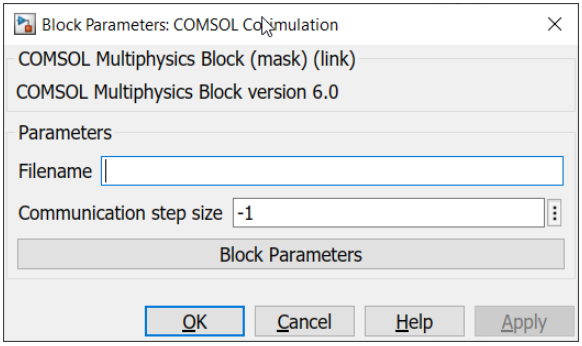
Adding a Cosimulation Block in Simulink

In the Simulink Library Browser you can find the COMSOL Cosimulation block in the section COMSOL 5.6.



The COMSOL Cosimulation block can be manipulated as any Simulink block, for instance to add it to the Simulink diagram, drag and drop the block.

In the Simulink diagram, to access the COMSOL Cosimulation block settings window, double-click the block.



In the **Filename** field enter the cosimulation file name. If you do not enter the full path for the cosimulation file, the COMSOL Cosimulation block searches in the following locations in the order they are listed below:

- The MATLAB current directory.
- The directory where the Simulink diagram is saved.
- The directories set as Model path using the function `mphsimsettings`.

In the **Communication step size** field you enter the time interval at which both programs communicate with each other.

Click **Block Parameters** if the cosimulation file is defined with block parameters and you want to modify the default value set during the cosimulation file export.

After you click **OK**, the block will change appearance to display the inputs and outputs that were defined when you exported the cosimulation file. If an image was set in the cosimulation file, it will be displayed in the block after the file is loaded.

After the Cosimulation

After running a cosimulation, you can extract data from Simulink based on the inputs and outputs of the COMSOL Cosimulation block and other blocks in the Simulink diagram. You can also perform postprocessing in COMSOL of the stored solution data.

In this section:

Saving the COMSOL Model

When performing a cosimulation the model is stored in memory on the COMSOL Multiphysics server. Using the command line in MATLAB it is possible to access and interact with this model.

To save the COMSOL model, enter the following command at the MATLAB prompt:

```
mphsave(<FILENAME>)
```

where *<FILENAME>* is the name of the COMSOL model to save.

The model can be loaded by COMSOL Multiphysics later for further analysis.

Editing the Model in the COMSOL Desktop

For a faster analysis, you can use the command `mphlaunch`. This command launches COMSOL Multiphysics, connects it to the COMSOL Multiphysics server that is used for the cosimulation, and loads the model into the user interface.

At this point, all postprocessing features available in COMSOL can be used to analyze the model. This is especially useful for verification of the simulation results in cases where accessing the inputs and outputs in Simulink is not sufficient for understanding the model's dynamics.

When opening COMSOL Multiphysics this way, the model becomes locked by the COMSOL user interface. Hence, before running another cosimulation in Simulink, you need to shut down the COMSOL user interface.

4

Reduced-Order Models

Reduced-order models can be useful for model analysis and control design as well as to improve the simulation performance. You may access the linearized state-space representation for a COMSOL model. Since finite-element models often have quite a high number of degrees of freedom it is desirable to extract a reduced-order model with fewer degrees of freedom.

In this chapter:

Working with Reduced-Order Models

COMSOL Multiphysics provides functionality to generate linearized reduced-order models based on eigenvalue solutions. The reduced system has much fewer degrees of freedom (DOFs) than the full model. See the Modal Reduced-Order Model and The Modal Solver Algorithm in the *COMSOL Multiphysics Reference Manual* for more information.

From the reduced-order model, use LiveLink™ *for* Simulink ® to retrieve the state-space matrices of the following system:

$$\begin{cases} \frac{dx}{dt} = Ax + Bu \\ y = Cx + Du \end{cases}$$

An alternative representation of the above dynamic system is:

$$M_C\dot{x} = M_CAx + M_CBu$$
$$y = Cx + Du$$

The latter form is more suitable for large systems because the matrices M_C and M_CA are usually sparse whereas A is dense.

Creating a Reduced-Order Model

In order to create a reduced-order model, three studies must be present in the model:

- Time-dependent study
- Eigenvalue/eigenfrequency study
- A study that includes the Model Reduction study step.

To enable reduced-order modeling in the COMSOL Desktop, click **Show Options...** , then under Study, select **Reduced-Order Modeling** .

For a detailed description for generating a reduced-order model in COMSOL Multiphysics see the tutorial *Thermal Controller, Reduced-Order Model* from the COMSOL Multiphysics Application Library.

Some physics interfaces require additional modules to support eigenfrequency analysis.

Extracting Reduced-Order State-Space Matrices

The tutorial in this section illustrates how to use the `mphreduction` function to extract the state-space matrices for a model. The studied heat transfer model contains a heat source, which is set as input in the state-space system. The temperature at a specific point is used as output. Follow the step-by-step instructions below to obtain the state-space matrices and to solve the system in Simulink.

MODEL DEFINITION IN THE COMSOL DESKTOP

1. Start the COMSOL Desktop.
2. In the **File** menu, select Application Libraries.
3. In the Application Libraries, browse to **LiveLink for Simulink>Tutorials**, and select `model_tutorial_llmatlab`. Click **Open**.

Model Parameters

The loaded model is used as base model for documentation tutorial. The geometry, mesh, and physics is set, but for this specific problem you need to edit the model to use parameters for the initial and external temperature as they will be used later in the reduced model.

1. Under **Global Definitions**, select **Parameters 1** node.
2. In the **Settings** window for **Parameters**, update the table according to the settings below:

NAME	EXPRESSION	DESCRIPTION
power	30[W]	Total power
Temp	300[K]	Base temperature
Text	300[K]	External temperature
T0	293.15[K]	Initial temperature

Physics Settings — Heat Transfer in Solids

In this section you will set up the Heat Transfer in Solids physics interface. If you do not have the Heat Transfer Module or MEMS Module skip the instructions below and go directly to Physics Settings — General Form PDE.

1. In the **Model Builder**, select **Heat Transfer in Solids > Initial Values 1** node. In the **Settings** window set the **Temperature** field to `T0`.
2. In the **Model Builder**, select **Heat Transfer in Solids > Heat flux 1** node. In the **Settings** window enter `Text` in **External temperature** field.

In this model the bottom temperature is imposed as a constraint, you will change it to heat flux condition instead.

3. Right-click **Heat Transfer in Solids > Temperature 1** node, and select **Disable**.
4. Right-click **Heat Transfer in Solids** node, and select **Heat Flux**.
5. Select boundary `3`.
6. Under **Heat Flux**, select **Convective heat flux**.
7. In the **Heat transfer coefficient** field enter `1e6`.
8. In the **External temperature** field enter `Temp`.

You have now set up the physics, and can go directly to Study Settings to continue.

Physics Settings — General Form PDE

In this section you will find the instruction to set up the physics using the General Form PDE interface. If you have the Heat Transfer Module or MEMS Module skip the instructions in this section and go directly to Study Settings.

1. In the **Model Builder**, right-click **Heat Transfer in Solids** node and select **Disable**.
2. In the **Physics** toolbar select **Add Physics**.
3. In the **Add Physics** window, expand **Mathematics>PDE Interfaces**. Select **General Form PDE (g)** and click **Add to Component 1**.
4. In the **Settings** window for **General Form PDE**, under **Units** section click **Select Dependent Variable Quantity**.
5. In the **Physical Quantity** window, expand **General**, and select **Temperature (K)**. Click **OK**
6. In the **Settings** window for **General Form PDE**, under **Units** section click **Select Source Term Quantity**.
7. In the **Physical Quantity** window, expand **Transport**, and select **Heat source (W/m^3)**. Click **OK**.

8 In the **Model Builder**, expand **General Form PDE** and select **General Form PDE 1** node.

9 In the **Settings** window for **General Form PDE**, under **Conservative Flux** section in the Γ table enter:

`-material.k11*ux`
`-material.k22*uy`
`-material.k33*uz`

10 In the **Settings** window for **General Form PDE**, under **Source Term** section in the *f* text field enter `0`.

11 In the **Settings** window for **General Form PDE**, under **Source Term** section in the d_a text field enter `material.rho*material.Cp`.

12 In the **Model Builder**, expand **General Form PDE** and select **Initial Values 1** node.

13 In the **Settings** window for **Initial Values**, under **Initial Values** section in the **Initial value for u** field enter `T0`.

14 In the **Model Builder**, right-click **General Form PDE** and select **Source**.

15 In the **Settings** window for **Source**, select domain `2`.

16 In the **Settings** window for **Source**, in the **Source Term** field enter `power/(1[cm]*1[cm]*1[mm])`.

17 In the **Model Builder**, right-click **General Form PDE** and select **Flux/Source**.

18 In the **Settings** window for **Flux/Source**, select boundaries `7 8 10 11 12`.

19 In the **Settings** window for **Flux/Source**, in the **Boundary Flux/Source** field enter `10[W/(m^2*K)]*(Text-u)`.

20 In the **Model Builder**, right-click **General Form PDE** and select **Flux/Source**.

21 In the **Settings** window for **Flux/Source**, select boundaries `3`.

22 In the **Settings** window for **Flux/Source**, in the **Boundary Flux/Source** field enter `1e6[W/(m^2*K)]*(Temp-u)`.

Study Settings

To create a reduced-order model, first add an unreduced model study and a training study to your model. In this section you will add a Time Dependent study and an eigenfrequency study that will be later be set as unreduced model and training studies, respectively.

1 In the **Study** toolbar, select **Add Study**.

2 In the **Add Study** window, select **Time Dependent** and click **Add Study**.

3 In the **Settings** window for **Time Dependent**, enter `range(0,1,50)` in the **Output times** field.

Add a domain point probe plot, which will define the output of the reduced-order model.

4 Under **Component 1** node, right-click **Definitions** and select **Probes>Domain Point Probe**.

5 In the **Settings** window for **Domain Point Probe**, locate the **Coordinates** field and enter: `1e-2 1e-2 1e-2`.

6 In the **Study** toolbar, select **Compute**.

7 In the **Add Study** window, expand **More Studies** and select **Eigenvalue**. Then click **Add Study**.

8 In the **Settings** window for **Eigenvalue**, select **Desired number of eigenvalues** and enter `60`.

9 In the **Settings** window for **Eigenvalue**, select **Search for eigenvalues around** and leave it with the default value.

10 In the **Study** toolbar, select **Compute**.

Reduced Order Model

Before adding a reduced-order study to your model you need to set the input, define the power variable `power`, the bottom temperature `Temp`, and the exterior temperature `Text`.

1 In the **Model Builder**, click **Show More Options**.

2 In the **Show More Options** window, under **Study**, select **Reduced-Order Modeling**.

3 Right-click **Global Definitions** and select **Reduced-Order Modeling>Global Reduced Model Inputs**.

4 In the **Settings** window for **Global Reduced Model Inputs**, edit the table as defined below:

CONTROL NAME	EXPRESSION
`power`	`30[W]`
`Temp`	`300[K]`
`Text`	`300[K]`

In the step below you will now add a model reduction study.

5 In the **Add Study** window, select **Empty Study**, and click **Add Study**.

6 Right-click **Study 3**, and select **Model Reduction**.

7 In the **Settings** window for **Model Reduction**, under **Model Reduction Settings** section enter the following settings:

NAME	VALUE IN LIST
Training study for eigenmodes	Study 2
Unreduced model study	Study 1
Defined by study step	Time Dependent

8 Under the **Outputs** section, enter the table as defined below:

VARIABLE	EXPRESSION	DESCRIPTION
Tout	comp1.ppb1	Temperature

9 Now click **Compute** to generate the reduced-order model.

10 The first part the modeling is done. You can now save your Model MPH-file and continue to follow the instruction from the MATLAB prompt.

EXTRACTING THE STATE SPACE MATRICES IN MATLAB

1 At the MATLAB prompt, to load the model you have just created, enter the command

```
model = mphopen(<modelname>);
```

where *<modelname>* is the name of the Model MPH-file. If the model file path is not set in MATLAB you can enter the full path of the model name.

2 A call to `mphreduction` creates the state-space matrices needed to simulate the reduced-order system. Enter:

```
MR = mphreduction(model, 'rom1', ...
    'out', {'MA' 'MB' 'C' 'D' 'Mc' 'x0'})
```

3 For later use, you can enter in MATLAB the value for the input of the dynamic system:

```
power = 30; Temp = 300; Text = 300;
```

RUN THE SIMULATION IN SIMULINK

1 At the MATLAB prompt, enter `simulink` to start Simulink.

2 In the **Simulink Start Page** window, select **Blank Model**.

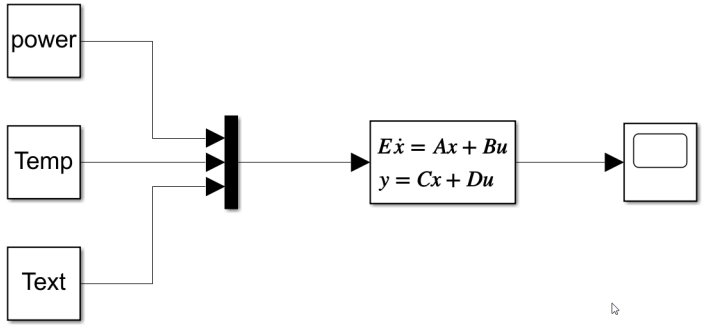
3 In the Simulink diagram add a **Descriptor State-Space** block.

4 Double-click the **Descriptor State-Space** block, and define the settings as in the table below:

PARAMETERS	VALUE
`E`	`MR.Mc`
`A`	`MR.MA`
`B`	`MR.MB`
`C`	`MR.C`
`D`	`MR.D`
`Initial conditions`	`MR.x0`

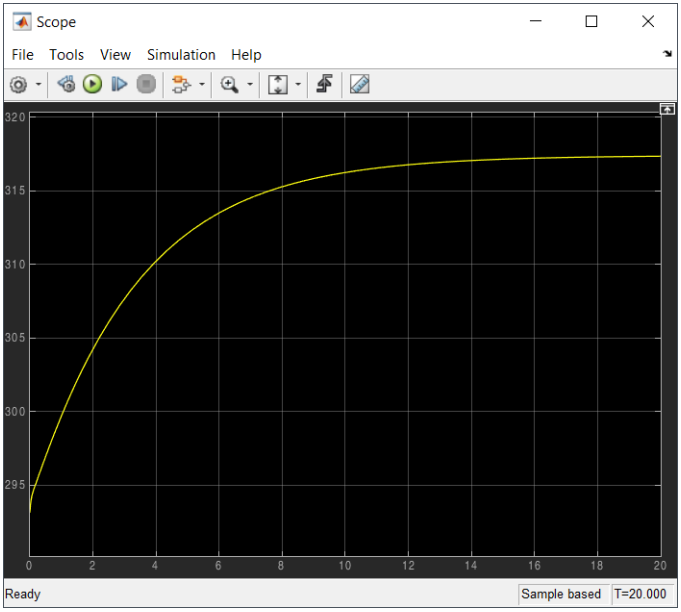
If you prefer to use a State-Space block in the Simulink diagram, you need to export the A, B, C, and D matrices. To proceed, in `mphreduction` replace the `out` property by `{'A' 'B' 'C' 'D' 'x0'}`.

5 Continue editing the Simulink diagram so it looks like in the figure below:



6 In the Simulation toolbar, locate Stop Time and enter 20.

7 You can now click Run to run the simulation.



5

Programming and Command Reference

In this chapter you find the additional interface functions that come with the LiveLink™ *for* Simulink®.

In this chapter:

Programming Reference

In this chapter you find detailed COMSOL® API reference information for the export to cosimulation file in LiveLink™ *for* Simulink®.

Adding an Export for Simulink Node with the COMSOL® API Syntax

To add an Export for Simulink® feature node to the COMSOL Multiphysics model using the COMSOL API, enter the command:

```
model.externalInterface.create(<simtag>, "SimulinkCosimulation")
```

where `<simtag>` is a string that set the tag of the feature node.

To set the cosimulation file name enter:

```
model.externalInterface(<simtag>).set("filename", <filename>)
```

where `<filename>` is a string that defines the name of the file.

To export the cosimulation file type:

```
model.externalInterface(<simtag>).run()
```

SETTING UP THE INPUTS

For each input you want in the cosimulation file you need to specify a name and an initial value. Note that input can only be chosen from the global parameters list.

A parameter can either be used as block parameter or input.

To create an input enter:

```
model.externalInterface(<simtag>).setIndex("inputname", <name>,
idx)
```

where `<name>` is a string that defines the parameter name used as input, and `idx` an integer that defines the input number.

Do not set parameters that define either geometry or mesh as inputs.

To set the initial value of an input enter:

```
model.externalInterface(<simtag>).setIndex("inputvalue", value,
idx)
```

where *value* is the desired initial value of input for the cosimulation.

To remove an input enter:

```
model.externalInterface(<simtag>).remove("inputname", idx)
model.externalInterface(<simtag>).remove("inputunit", idx)
model.externalInterface(<simtag>).remove("inputvalue", idx)
```

where *idx* is the index of the input to remove.

The order of the input corresponds of the order of the entry port in the COMSOL Cosimulation block in Simulink. To change the order of the input enter:

```
model.externalInterface(<simtag>).move("inputname", new int []
{idx}, relativeChange)
model.externalInterface(<simtag>).move("inputunit", new int []
{idx}, relativeChange)
model.externalInterface(<simtag>).move("inputvalue", new int []
{idx}, relativeChange)
```

where *idx* is the index of the input to move and *relativeChange* an integer corresponding to the relative position change (negative for upward change, positive for downward change).

SETTING UP THE BLOCK PARAMETERS

For each block parameter you want in the cosimulation file you need to specify a name and an initial value. Note that block parameter can only be chosen from the global parameters list.

A parameter can either be used as block parameter or input.

To create a block parameter enter:

```
model.externalInterface(<simtag>).setIndex("blockname", <name>,
idx)
```

where *<name>* is a string that defines the parameter name used as block parameter, and *idx* an integer.

To set the value of a block parameter enter:

```
model.externalInterface(<simtag>).setIndex("blockvalue", value,
idx)
```

To remove a block parameter enter:

```
model.externalInterface(<simtag>).remove("blockname", idx)
model.externalInterface(<simtag>).remove("blockunit", idx)
model.externalInterface(<simtag>).remove("blockvalue", idx)
```

where *idx* is the index of the block parameter to remove.

SETTING UP THE OUTPUTS

To export a file for cosimulation, you need to specify at least one output. The output can be define with any expression with a global scope.

To create an output enter:

```
model.externalInterface(<simtag>).setIndex("outputexpr", <expr>,
idx)
```

where *<expr>* is a string that defines the output expression, and *idx* the index position.

To set an output name enter:

```
model.externalInterface(<simtag>).setIndex("outputdescr",
<descr>, idx)
```

where *<descr>* is a string that set the output name at the position *idx*.

To remove an output enter:

```
model.externalInterface(<simtag>).remove("outputexpr", new int []
{idx})
model.externalInterface(<simtag>).remove("outputdescr", idx)
```

where *idx* is the index of the output to remove.

The order of the output corresponds of the order of the output port in the COMSOL Cosimulation block in Simulink. To change their order enter:

```
model.externalInterface(<simtag>).move("outputexpr", new int []
{idx}, relativeChange)
model.externalInterface(<simtag>).move("outputputunit", new int []
{idx}, relativeChange)
model.externalInterface(<simtag>).move("outputdescr", new int []
{idx}, relativeChange)
```

where *idx* is the index of the output to move and *relativeChange* an integer corresponding to the relative position change (negative for upward change, positive for downward change).

SETTING UP THE STUDY

To set the study to run in cosimulation enter:

```
model.externalInterface(<simtag>).set("study", <stdtag>)
```

where *<stdtag>* is a string that defines the tag of the study.

Only use study including a single study step (Stationary or Time Dependent).

For Time Dependent study you can specify how to store the solution.

To store the solution only at communication points enter:

```
model.externalInterface(<simtag>).set("storeacctostep", false)
```

To include the solution at steps defined by the study, enter:

```
model.externalInterface(<simtag>).set("storeacctostep", true)
```

To store the solution at the end of the communication step only enter:

```
model.externalInterface(<simtag>).set("store", "end")
```

To store the solution at both start and end of the communication step enter:

```
model.externalInterface(<simtag>).set("store", "both")
```

SETTING UP THE DEPENDENCIES

When the selected study is of type Time Dependent, you may want to specify the dependencies between outputs and inputs.

To set dependencies enter:

```
model.externalInterface(<simtag>).set("dependencies", idxIn,
idxOut)
```

where *idxIn* is the input index and *idxOut* is the output index.

SETTING UP THE BLOCK IMAGE

For better visualization of the COMSOL Cosimulation block you can include an image.

To set an image enter:

```
model.externalInterface(<simtag>).set("image", <filename>)
```

where *<filename>* is a string.

To embed the image inside the COMSOL model, enter:

```
model.externalInterface(<simtag>).importImage()
```

To discard the image, enter:

```
model.externalInterface(<simtag>).discard()
```

Summary of Commands

mphapplicationlibraries
mphlaunch
mphload
mphreduction
mphsave
mphsimsettings
mphtags
mphupdatesystem
mphversion

Commands Grouped by Function

INTERFACE FUNCTIONS

FUNCTION	PURPOSE
mphlaunch	Launch a COMSOL Multiphysics Client, connect it to the server, and load a model.
mphload	Load a COMSOL model MPH-file.
mphsave	Save a COMSOL model.
mphversion	Return the version number of COMSOL Multiphysics.
mphsimsettings	GUI for preferences settings for LiveLink for Simulink.
mphupdatesystem	Update COMSOL Cosimulation blocks in Simulink systems

UTILITY FUNCTIONS

FUNCTION	PURPOSE
mphreduction	Return reduced order state space matrices for a model.

MODEL INFORMATION AND NAVIGATION

FUNCTION	PURPOSE
mphapplicationlibraries	GUI for viewing the product Application Libraries.
mphtags	Get tags and names for nodes in a COMSOL model.

mphapplicationlibraries

Graphical user interface (GUI) for viewing the Application Libraries.

SYNTAX

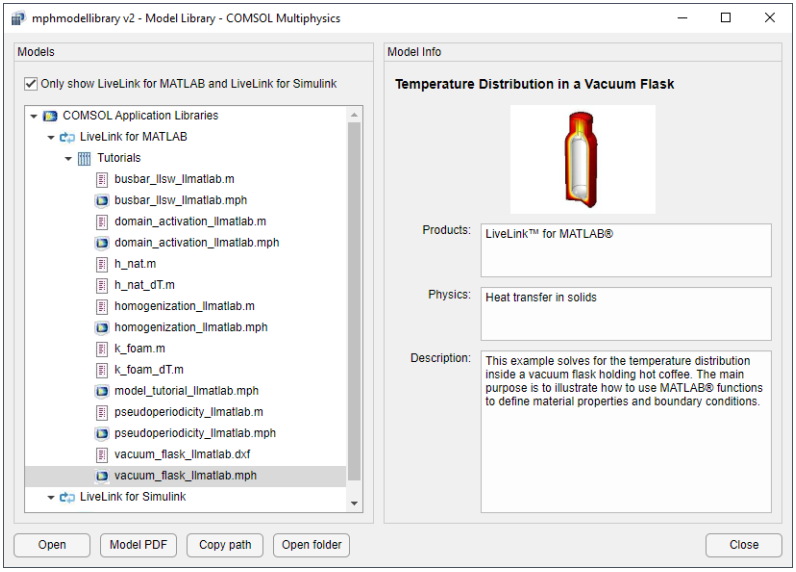
```
mphapplicationlibraries
```

DESCRIPTION

`mphapplicationlibraries` starts a GUI to visualize and access the example model available in the COMSOL Application Libraries. The model MPH-file can be loaded in MATLAB® and the model documentation PDF-file is accessible directly.

Models that are specific to LiveLink™ *for* MATLAB® also contains the script M-file.

Models that are specific to LiveLink™ *for* Simulink® also contains the simulation SLX-file and opens directly in Simulink.



mphlaunch

Launch COMSOL Desktop, connect it to the running server, and import an application.

SYNTAX

```
mphlaunch
mphlaunch(model)
mphlaunch ModelTag
mphlaunch(..., timeout)
```

DESCRIPTION

`mphlaunch` launches a COMSOL Multiphysics Client and connect it to the same server as MATLAB® is connected to. Then it imports the model on the server into the COMSOL Multiphysics Client.

`mphlaunch(model)` does the same as above, but uses the `model` argument to select which model is imported.

`mphlaunch ModelTag` uses the model with the tag `'ModelTag'` in the server to be imported. This can also be done using the syntax: `mphlaunch('ModelTag')`

`mphlaunch(..., tms)` uses the timeout `tms` (in milliseconds) to force MATLAB to wait until the COMSOL *server* is free again. The default timeout value is 500. A negative value results in no timeout.

EXAMPLE

Load the file `model_tutorial_llmatlab.mph`:

```
model = mphopen('model_tutorial_llmatlab');
```

Launch a COMSOL Multiphysics Client, connect it with the running server, import the model defined as `model`:, and set a timeout of 1 s:

```
mphlaunch(model,1000);
```

mphload

Load a COMSOL Multiphysics model MPH-file.

SYNTAX

```
model = mphload(filename)
model = mphload(filename, mtag)
model = mphload(filename, mtag, '-history')
model = mphload(filename, mtag, pwd)
[model, filename] = mphload(filename, ...)
```

DESCRIPTION

`model = mphload(filename)` loads a COMSOL model object saved with the name `filename` and assigns the default tag `Model` in the COMSOL *server*. If a model with tag `Model` already exists and is also open in a COMSOL Multiphysics client, the loaded model an index number is appended to the tag, for instance `Model1`. The model object is accessible at the MATLAB prompt using the variable `model`.

`model = mphload(filename, mtag)` loads a COMSOL model object and assigns the tag `mtag` in the COMSOL *server*.

`model = mphload(filename, mtag, '-history')` turns on model history recording.

`model = mphload(filename, mtag, pwd)` loads the COMSOL model object saved with the name `filename` protected with the password `pwd`.

`model = mphload(mtag)` link the model already loaded on the COMSOL *server* with the tag `mtag`. The model object is accessible at the MATLAB prompt using the variable `model`.

`[model, filenameloaded] = mphload(filename, ...)` also returns the full file name `filenameloaded` of the file that was loaded.

The model tag `mtag` and the password `pwd` are defined as string.

If the model tag is the same as a model that is currently in the COMSOL *server* the loaded model overwrites the existing one.

Note that MATLAB® searches for the model on the MATLAB path if an absolute path is not supplied.

`mphload` turns off the model history recording by default, unless the property `'-history'` is used.

The extension `mph` can be omitted.

`mphload` does not look for lock file when opening a model in the COMSOL *server*.

EXAMPLE

Load the file `model_tutorial_llmatlab.mph`:

```
model = mphload('model_tutorial_llmatlab');
```

Load the file `model_tutorial_llmatlab.mph` and set the model name in the COMSOL *server* to `Model2`:

```
model = mphload('model_tutorial_llmatlab','Model2');
```

Load `model_tutorial_llmatlab.mph` and return the filename:

```
[model, filename] = mphload('model_tutorial_llmatlab');
```

SEE ALSO

`mphsave`

mphreduction

Return reduced-order state-space matrices for a model.

SYNTAX

```
data = mphreduction(model, romtag, ...)
```

DESCRIPTION

`data = mphreduction(model, romtag, ...)` extracts the reduced order state-space matrices from the reduced order model `romtag`. The reduced order model can either be created in the COMSOL GUI or via the API.

The function `mphreduction` accepts the following property/value pairs:

TABLE 5-1: PROPERTY/VALUE PAIRS FOR THE MPHREDUCTION COMMAND

PROPERTY	VALUE	DEFAULT	DESCRIPTION
`out`	Cell array of strings	`'all'`	Names of output matrices
`return`	`struct` | `ss`	`struct`	Return type

The following values are valid for the `out` property:

TABLE 5-2: PROPERTY/VALUE PAIRS FOR THE PROPERTY OUT.

PROPERTY	EXPRESSION	DESCRIPTION
`out`	`Kr`	Stiffness matrix
	`Dr`	Damping matrix
	`Dra`	Damping ratio matrix
	`Er`	Mass matrix
	`Br`	Input matrix
	`Cr`	Output matrix
	`F`	Input feedback matrix
	`BOr`	Initial value input matrix
	`BOrdot`	Initial value time derivative input matrix
	`Brdot`	Time derivative input matrix
	`Brdotdot`	Second time derivative input matrix
	`Mc`	Damping matrix
	`MA | A`	Stiffness matrix
	`MB | B`	Input matrix
	`D`	Input feedback matrix
	`C`	Output matrix
	`L`	Load vector
	`YO`	Output bias
	`UO`	Initial value vector
	`UdotO`	Initial derivative vector
	`Kud`	Stiffness matrix times ud

The return type ss requires that the Control System Toolbox is installed.

mphsave

Save a COMSOL Multiphysics model.

SYNTAX

```
mphsave(model)
mphsave(model,filename,...)
mphsave(filename)
mphsave(mtag)
mphsave(mtag,filename,...)
```

DESCRIPTION

`mphsave(model)` saves the COMSOL model object `model`.

`mphsave(model,filename,...)` saves the COMSOL model object `model` to the file named `filename`.

`mphsave(filename)` saves the unique COMSOL model that is loaded in the COMSOL server to the file named `filename`.

`mphsave(mtag)` saves the COMSOL model loaded in the COMSOL server with the tag `mtag`.

`mphsave(mtag,filename,...)` aves the COMSOL model loaded in the COMSOL server with the tag `mtag` to the file named `filename`.

The function `mphsave` accepts the following property/value pairs:

TABLE 5-3: PROPERTY/VALUE PAIRS FOR THE MPHSAVE COMMAND

PROPERTY	VALUE	DEFAULT	DESCRIPTION
`component`	`on | off`	`off`	Save M-file using the component syntax
`copy`	`on | off`	`off`	Save a copy of the model
`description`	`String`		Set or append model description
`excludedata`	`on | off`		Exclude built, computed, an plotted data
`filenameis`	`fullpath | name | path`	`fullpath`	Use filename as the selected type
`optimize`	`size | speed`		Optimize for speed or file size
`store`	`on | off`	`on`	Store the filename in most recently used files

If the file name is not provided, the model has to be saved previously on disk.

If the file name does not provide a path, the file is saved relatively to the current path in MATLAB®.

The model can be saved as an MPH-file, Java file, or M-file. The file extension determines which format that is saved.

Note: Model created with older version than COMSOL 5.3 cannot be saved using the component syntax.

SEE ALSO

`mphload`

mphsimsettings

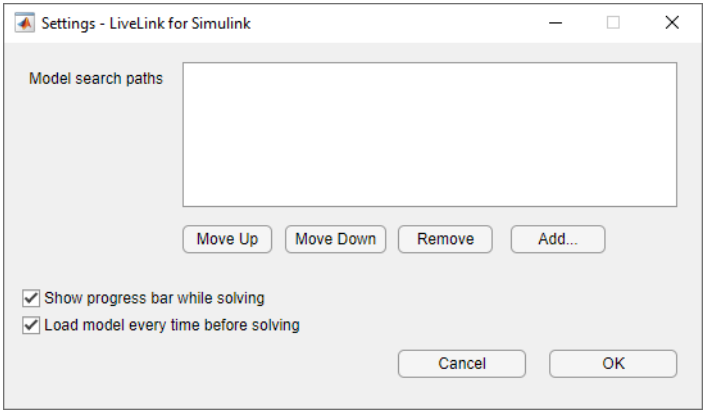
Graphical user interface (GUI) for preferences settings for LiveLink for Simulink.

SYNTAX

```
mphsimsettings
```

DESCRIPTION

`mphsimsettings` starts a GUI for preferences to LiveLink™ *for* Simulink®.



Use the **Add** and **Remove** buttons to update the MATLAB search path with the path of the model to run in cosimulation with Simulink.

Use the **Move Up** and **Move Down** buttons to change the position of the model path in the MATLAB search path.

Select the **Show progress bar while solving** check box to enable the COMSOL progress bar when running a cosimulation.

Note: The progress bar is not available on macOS.

Select the **Load model every time before solving** check box to ensure to run a cosimulation with the save version of the model.

mphtags

Get tags and names for nodes in a COMSOL Multiphysics model.

SYNTAX

```
mphtags(model)
mphtags(node)
mphtags(model, type)
[tags,labels,displaystrings] = mphtags(...)
mphtags
mphtags -show
[tags,filename,fullfilename] = mphtags
```

DESCRIPTION

`mphtags` is used to retrieve tags from nodes in a COMSOL Multiphysics model or tags from models that are loaded on the server.

When `mphtags` is called with a `model` or `node` variable the tags are returned form the model. `mphtags` also be called using a model variable and a type, where type can be one of these strings: `result, dataset, table, numerical`, and `export` to give easy access to the nodes under the result node. It is sufficient to use the first letter of the types.

If `mphtags` is called with output arguments it is possible to get both the tags as well as labels and display names used for the nodes. For example,

```
[tags,labels,displaystrings] = mphtags(model.geom)
```

If `mphtags` is called with the root model node as argument, the filename of the model can be returned:

```
[tag,filename,displaystring] = mphtags(model)
```

`mphtags` can be used to return a list of files currently loaded on the server. For example,

```
[tags,filename,fullfilename] = mphtags
```

In order to see this information quickly it is possible to call `mphtags` like this:

```
mphtags -show
```

that just produces output that is useful viewing on screen.

mphupdatesystem

Update COMSOL Cosimulation blocks in Simulink systems.

`mphupdatesystem(system)` updates all COMSOL Cosimulation blocks in `system` to the version included in the currently running LiveLink™ *for* Simulink®.

`mphupdatesystem(system,'librarypath',path,'updateversion',version)` updates all COMSOL Cosimulation blocks to `version` and provides an additional path to the libraries.

The function `mphupdatesystem` accepts the following property/value pairs:

TABLE 5-4: PROPERTY/VALUE PAIRS FOR THE MPHUPDATESYSTEM COMMAND

PROPERTY	EXPRESSION	DEFAULT	DESCRIPTION
`createbackup`	`off` | `on`	`on`	Creates a backup of any modified file
`librarypath`	String		Additional path to COMSOL Cosimulation block
`overwritebackup`	off | on	`off`	Value of eigenvalue linearization point
`postupdateaction`	open | save	save	Determines if the updated system should be opened or saved
`updateblock`	`off` | `on`	`on`	Update Cosimulation block
`updatefmu`	`off` | `on`	on	Update fmu
`updateversion`	`'5.6'`		Error for undefined operations

mphversion

Return the version number for COMSOL Multiphysics.

SYNTAX

```
v = mphversion
[v,vm] = mphversion(model)
```

DESCRIPTION

`v = mphversion` returns the COMSOL Multiphysics version number that MATLAB is connected to as a string.

`[v,vm] = mphversion(model)` returns the COMSOL Multiphysics version number that MATLAB is connected to as a string in the variable `v` and the version number of the model in the variable `vm`.

EXAMPLE

Load `model_tutorial_llmatlab.mph`:

```
model = mphopen('model_tutorial_llmatlab');
```

Get the version numbers:

```
[version, model_version] = mphversion(model)
```

SEE ALSO

mphload, mphsave

Index