Dymola

Dynamic Modeling Laboratory

Dymola Release Notes

Dymola 2021x

The information in this document is subject to change without notice.

Document version: 1

Dassault Systèmes AB
Ideon Gateway
Scheelevägen 27 – Floor 9
SE-223 63 Lund
Sweden

Support: https://www.3ds.com/support
URL: https://www.dymola.com/
Phone: +46 46 270 67 00

Contents

1 Important notes on Dymola 5
2 About this booklet 5
3 Dymola 2021x 6
3.1 Introduction 6
3.1.1 Additions and improvements in Dymola 6
3.1.2 New and updated libraries 7
3.2 Developing a model 8
3.2.1 New version of Modelica Standard Library 8
3.2.2 Handling the textColor attribute 9
3.2.3 Routing of connections 9
3.2.4 Spacing and aligning components automatically 12
3.2.5 GUI for displaying and setting global Boolean, Integer, Real and String variables 13
3.2.6 Display of browsers and other windows controlled from the status bar 15
3.2.7 Minor improvements 16
3.3 Simulating a model 19
3.3.1 Improved performance of static simulations 19
3.3.2 Simulation Analysis: Equation incidence graphs 19
3.3.3 Plot tab 24
3.3.4 Animation window 30
3.3.5 Scripting 31
3.3.6 Improved simulation logging 32
3.3.7 Display of browsers and other windows controlled from the status bar 35
3.3.8 Minor improvements 35
3.4 Installation 40
3.4.1 Support for Dymola in “dark mode” 40
3.4.2 Installation on Windows 42
3.4.3 Installation on Linux 42
3.4.4 Dymola license server on Windows and Linux 42
3.5 Model Experimentation 43
3.5.1 Improved GUI for sweeping parameters 43
3.6 Other Simulation Environments 45
3.6.1 Dymola – Matlab interface 45
3.6.2 Real-time simulation 45
3.6.3 DDE communication 46

3.6.4 OPC communication 46
3.6.5 Java, Python, and JavaScript Interface for Dymola 46
3.6.6 FMI Support in Dymola 46
3.7 Modelica Standard Library and Modelica Language Specification 52
3.8 New libraries 52
3.8.1 Aviation Systems Library 52
3.9 Documentation 54
3.10 Appendix – Installation: Hardware and Software Requirements 55
3.10.1 Hardware requirements/recommendations 55
3.10.2 Software requirements 55

1 Important notes on Dymola

Installation on Windows

To translate models on Windows, you must also install a supported compiler. The compiler is not distributed with Dymola. Note that administrator privileges are required for installation. Three types of compilers are supported on Windows in Dymola 2021x:

Microsoft Visual Studio C++

This is the recommended compiler for professional users. Both free and full compiler versions are supported. Refer to section "Compilers" on page 56 for more information. **Note** that from Dymola 2020x, Visual Studio C++ compilers older than version 2012 are no longer supported.

Intel

Dymola 2021x has limited support for the Intel Parallel Studio XE compiler. For more information about this compiler, see section "Compilers" on page 56; the section about Intel compilers.

Important. The support for Intel compilers will be discontinued in a future release.

GCC

Dymola 2021x has limited support for the MinGW GCC compiler, 32-bit and 64-bit. For more information about GCC, see section "Compilers" on page 56; the section about GCC compilers.

Installation on Linux

To translate models, Linux relies on a GCC compiler, which is usually part of the Linux distribution. Refer to section "Supported Linux versions and compilers" on page 58 for more information.

2 About this booklet

This booklet covers Dymola 2021x. The disposition is similar to the one in Dymola User Manuals; the same main headings are being used (except for, e.g., Libraries and Documentation).

3 Dymola 2021x

3.1 Introduction

3.1.1 Additions and improvements in Dymola

A number of improvements and additions have been implemented in Dymola 2021x. In particular, Dymola 2021x provides:

- Support for the new Modelica Standard Library version 4.0.0 (page 8), including handling of textColor attribute (page 9)
- Support for Dymola in "dark mode" (page 40)
- Routing of connections (page 9)
- Spacing and aligning components automatically (page 12)
- GUI for displaying and setting global Boolean, Integer, Real and String variables (page 13)
- Improved performance of static simulations (page 19)
- Support for displaying equation incidence graphs for simulation analysis (page 19)
- Improved simulation logging (page 32)
- Color-coding of tables containing numeric values (page 28)
- Improvements of plotting
    - Plot tab split into two subtabs (page 24)
    - Simplified creation of 2D plots (page 25)
    - Curve colors controlled by simulation result files (page 27)
    - Color-coded tables (page 28)
    - Improvements for predefined plots (page 29)
    - Re-calculation of plot expressions after simulation (page 29)
- Improved lighting model in the animation window (page 30)
- Improved control of display of browsers and other windows (page 15)
- FMI support improvements
    - Filtering signals when importing an FMU (page 48)
- Dassault Systèmes License Server (DSLS) on Windows and Linux for Dymola (page 42)
- Discontinued OPC support (page 46)
- Documentation: A new "*Dymola Full User Manual*" containing all chapters of Dymola User Manual 1A – 2C (page 54)

3.1.2 New and updated libraries

New libraries

In this Dymola version, the Aviation Systems Library has been added.

For more information about this new library, please see the section “New libraries” starting on page 52.

Updated libraries

The following libraries have been updated:

- Battery Library, version 2.2.0
- Brushless DC Drives Library, version 1.1.2
- ClaRa DCS Library, version 1.4.0
- ClaRa Grid Library, version 1.4.0
- ClaRa Plus Library, version 1.4.0
- Claytex Library, version 2020.3
- Claytex Fluid Library, version 2020.3
- Cooling Library, version 1.4.1
- Dassault Systemes Library, version 1.5
- DataFiles, version 1.1.0
- Design, version 1.1.0
- Dymola Commands Library, version 1.10
- Dymola Models Library, version 1.2
- Electric Power Systems Library, version 1.4
- Electrified Powertrains Library (ETPL), version 1.3.3
- Flexible Bodies Library, version 2.3.1
- Flight Dynamics Library, version 1.0.4
- Fluid Dynamics Library, version 2.10.0
- Fluid Power Library, version 2020.3
- FTire Interface Library, version 1.1.0
- Human Comfort Library, version 2.10.0
- HVAC (Heating, Ventilation, and Air Conditioning) Library, version 2.10.0
- Hydrogen Library, version 1.3.3
- Model Management, version 1.2.0
- Modelica Standard Library, version 4.0.0
- Modelica_DeviceDrivers, version 2.0.0
- Modelica_LinearSystems2, version 2.4.0
- Modelica_StateGraph2, version 2.1.0
- Optimization, version 2.2.5

- Plot3D, version 1.1.0
- Pneumatic Systems Library, version 1.4.1
- PowerTrain Library, version 2.6.0
- Testing Library, version 1.3.1
- Thermal Systems Library, version 1.6.1
- Thermal Systems Mobile AC Library, version 1.6.1
- UserInteraction, version 0.70
- Vehicle Interfaces, version 2.0.0
- VeSyMA (Vehicle Systems Modeling and Analysis) Library, version 2020.3
- VeSyMA - Engines Library, version 2020.3
- VeSyMA - Powertrain Library, version 2020.3
- VeSyMA - Suspensions Library, version 2020.3
- VeSyMA2ETPL Library, version 2020.3
- Visa2Base, version 1.9
- Visa2Paper, version 1.9
- Visa2Steam, version 1.9
- Wind Power, version 1.1.2

For more information about the updated libraries, please see the Release Notes section in the documentation for each library, respectively.

3.2 Developing a model

3.2.1 New version of Modelica Standard Library

The new major version 4.0.0 of Modelica Standard Library (MSL) is included in this Dymola 2021x distribution.

By default, Dymola 2021x will start with MSL 4.0.0 if installing Dymola for the first time.

All libraries and demos included in the Dymola distribution support MSL 4.0.0.

Notes:

- If the new version MSL 4.0.0 should be used as default version when earlier versions of Dymola have been used previously, apply the command **Tools > Options**, select the **Version** tab, select Modelica version 4.0.0 and tick **Force upgrade of models to this version**.
- If you need to work with an older Modelica model that has not been converted to the new MSL version, you can install MSL 3.2.3 libraries separately. They are included in the Dymola media as `extra\CompatibilityLibraries MSL 3.2.3.zip`. To use them, unpack them in, typically, `Program Files\Dymola 2021x\Modelica\Library`. Note the above item to select proper version.

Some notes about MSL 4.0.0:

- MSL 4.0.0 is not backward compatible to previous MSL versions. A tested conversion script is provided to transform models and libraries of previous versions 3.x.y to the new version.
- Two new libraries have been added:
    - Modelica.Clocked to precisely define and synchronize sampled data systems with different sampling rates.
    - Modelica.Electrical.Batteries offers simple battery models.
- MSL 4.0.0 is based the recent Modelica Language Specification 3.4.

3.2.2 Handling the textColor attribute

The attribute `textColor` was introduced in the Modelica Language Specification 3.3 in text annotations. You could still use the attribute `lineColor` in text annotations, but that use was stated as to be deprecated. In Modelica Standard Library (MSL) version 4.0.0, this is implemented.

To be able to select how to handle these attributes when, for example, wanting to work with older models, libraries, and MSL versions, there is in Dymola 2020x a new Boolean flag `Advanced.Editor.NewTextAttributes` available.

The flag is by default `false`, meaning that existing attributes `lineColor` in text annotations are preserved, but for MSL 4.0.0 and later, attributes `textColor` are created.

If you set the flag to `true`, the attribute `textColor` is always used in such cases.

You can automatically convert `lineColor` to `textColor` in text annotations for a library, use (replace "Modelica" by the name of your library):

```
updateModelicaAnnotations("Modelica",
removeTextDeprecated=true, renameTextColor=true);
```

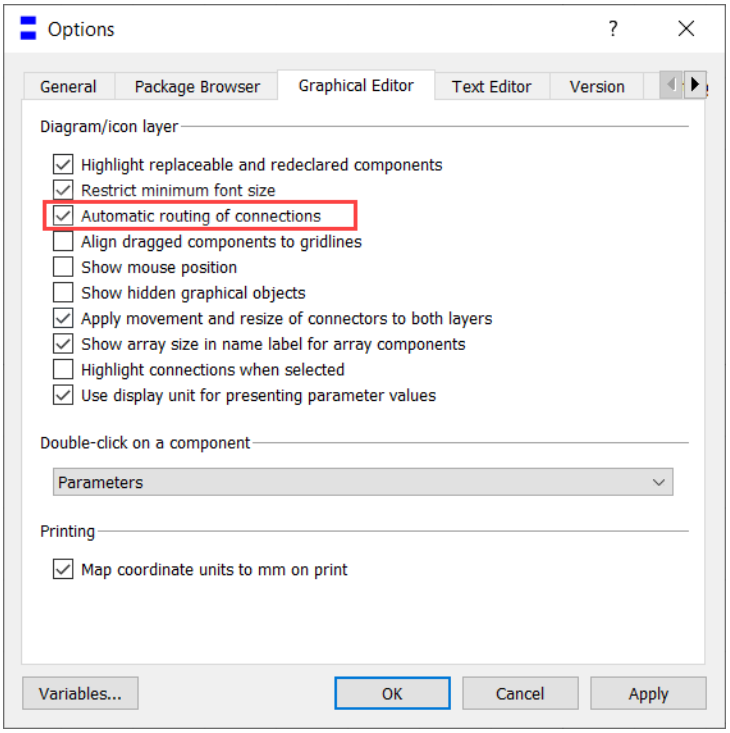
3.2.3 Routing of connections

Automatic routing when creating connections

In Dymola 2021x routing of connections are by default applied when creating them or when moving the connected components more than half a visible grid unit. The creation of a new connection is performed in two steps. The first step (applying e.g. Manhattanize) was done also in previous versions of Dymola. In the second step, automatic routing is performed. Note that if you added intermediate connection points, the automatic routing is not performed.

The command **Undo** is applied to each step, this means that after having created a connection, if you undo once, you undo the routing, if you undo again, the connection is deleted.

You can control if automatic routing should be used by the setting **Automatic routing of connections**, reached by the command **Tools > Options**, the **Graphical Editor** tab:

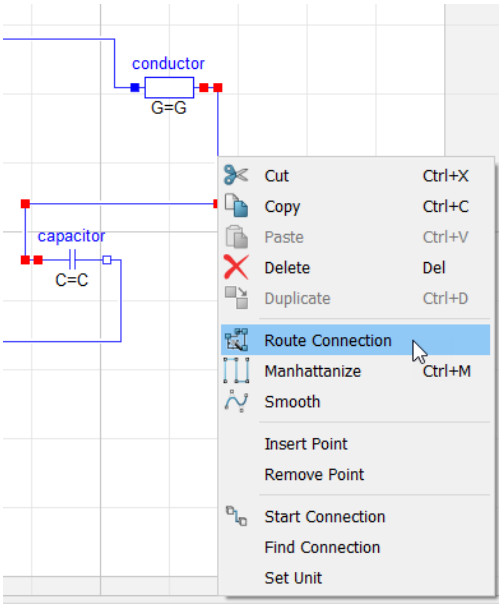


This setting is by default activated. This setting corresponds to the flag `Advanced.Editor.Routing.Automatic = true`.

Commands for routing present connections

Connection context menu command “Route Connection”

A new command **Route Connection** is available in the context menu of a connection:

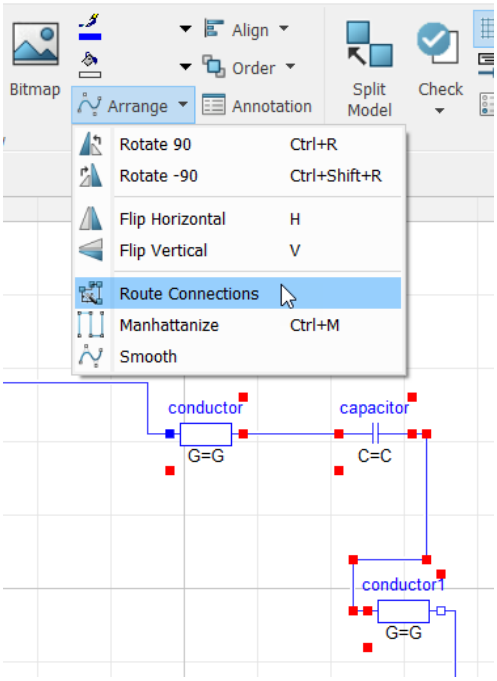


Notes:

- You can select several connections using **Shift+click** before right-clicking to display the context command.
- Routing is also applied if you move the selected components more than half a grid unit.

Command "Arrange > Route Connections"

A new command **Arrange > Route Connections** is available in the **Graphics** tab:



Notes:

- You can select a part of an open class by framing that part, and then applying the command to route all connections selected in the framed part.
- If you have nothing selected, the command works on all connections selectable in that class.

Avoiding crossing existing (unrelated) connections when routing connections

By default, when performing routing, crossing other (unrelated) connections is avoided, if possible.

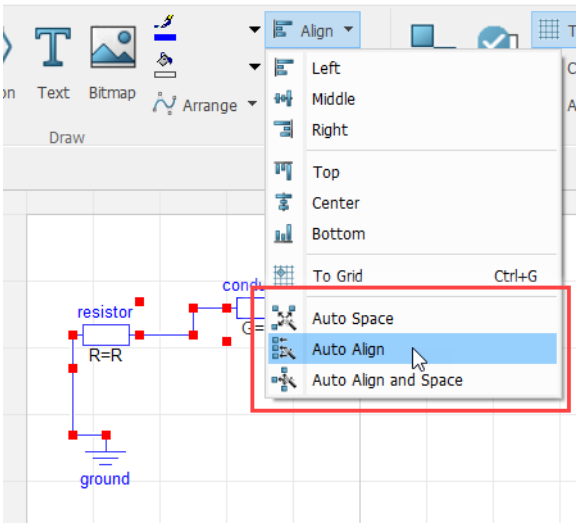
The feature is controlled by the flag `Advanced.Editor.Routing.AvoidCrossing`. The default value of the flag is `true`.

3.2.4 Spacing and aligning components automatically

You can space and align components using three new commands.

To work with specific components, you can multi-select them, otherwise, if no component is selected, the commands work on all selectable components. Note that the commands work only on components, not on graphical primitives like lines or rectangles.

The commands are:

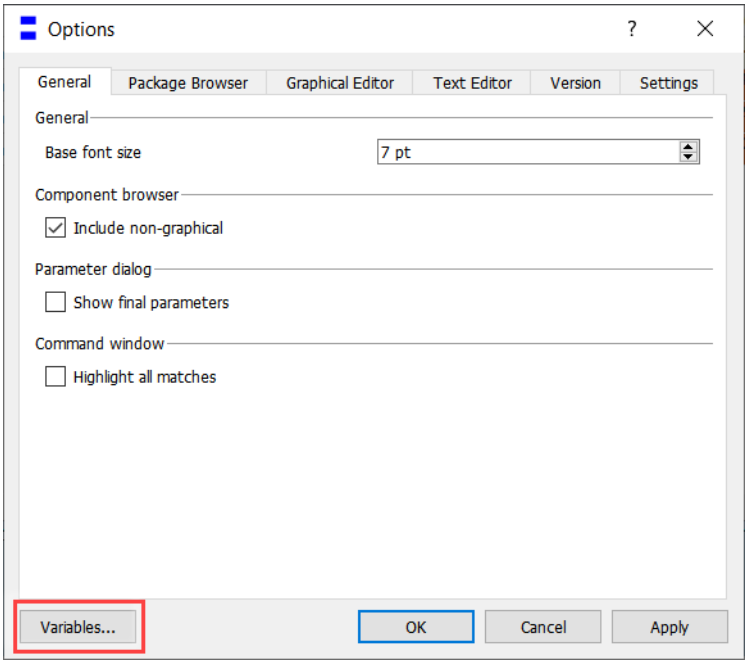


- **Graphics > Align > Auto Space** to space the components. Note that the size of the components are not modified.
- **Graphics > Align > Auto Align** to align the components, in both x and y direction. Notes:
    - Components that you have previously aligned by other commands may not be aligned to each other anymore: they may be aligned with other components depending on the algorithms.
    - For manually well-aligned classes, you may not improve the layout by this command
- **Graphics > Align > Auto Align and Space** to both align and space the components corresponding to using both the above commands. The notes of these commands applies for this command as well.

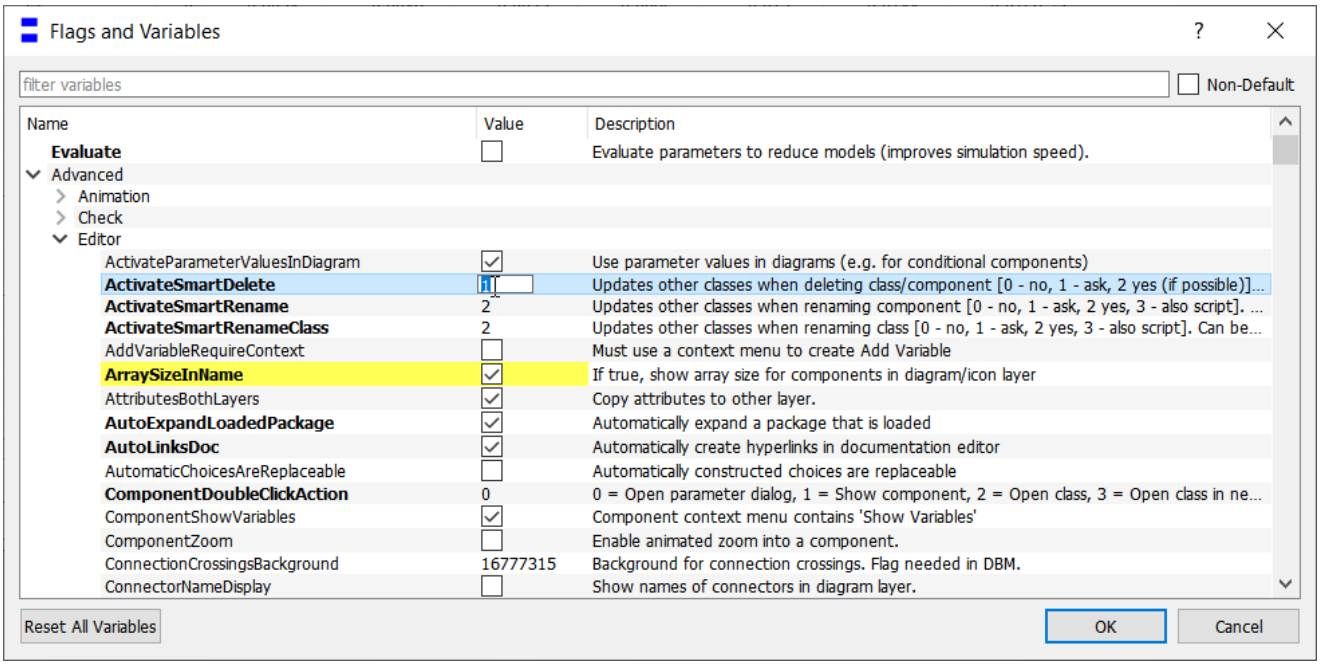
3.2.5 GUI for displaying and setting global Boolean, Integer, Real and String variables

The former GUI for displaying and setting Boolean flags, reached by the command button **Flags…** in the **Tools > Options** command dialog, has been extended. It now also covers global settings for integers, reals, and strings, in addition to Boolean flags.

Because of this extension, the name of the command button is changed to **Variables…**:



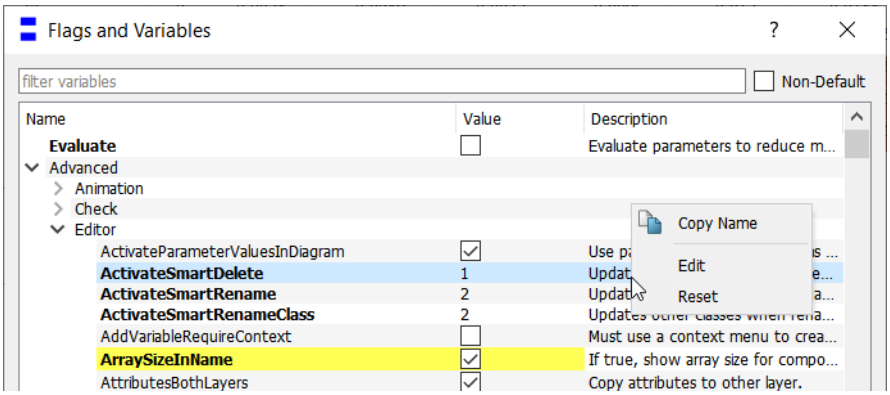
The dialog that opens when clicking this button may look like:



To accommodate the new types, a **Value** column has been added, and the checkbox for setting flags has been moved to that column.

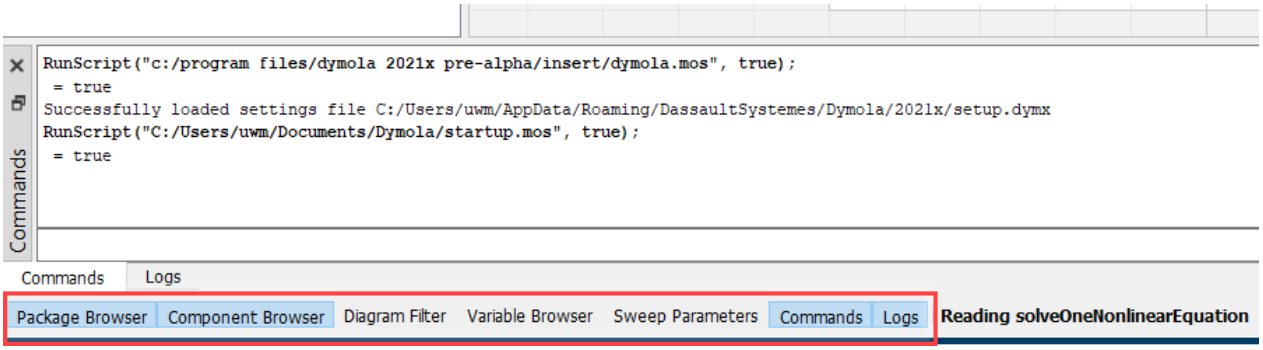
For other kind of variables than Boolean, clicking twice in the value field of a selected variable starts the editor, as in the example above. (Note that double-click has no effect.)

The context menu of a flag/variable has been extended with two new choices, **Edit** to start editing a variable (for a Boolean flag it toggles the value), and **Reset** which sets the variable to its default value.



3.2.6 Display of browsers and other windows controlled from the status bar

What browsers and other windows that should be displayed are in previous Dymola versions controlled by the command **Windows > Docked Windows**. In Dymola2021x, this command is replaced by buttons on the status bar at the left bottom of the Dymola main window:



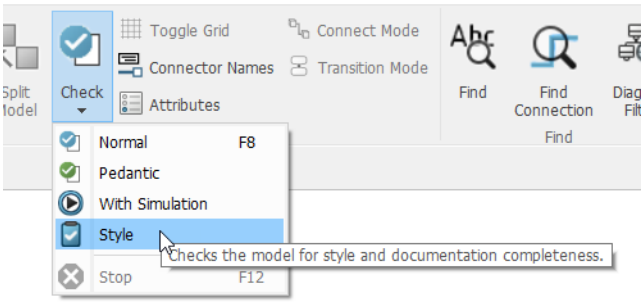
You can click on the buttons to display/hide the browser/windows. The buttons work in toggle mode. You can have different selections for different ribbon tabs (for example, one for set of displayed browser/windows for the **Graphics** tab, another set for the **Text** tab). The selections are stored between sessions.

Note the status message after the command buttons, in bold to make it more visible.

3.2.7 Minor improvements

Extended Check with style checking

The function `checkStyle()` has been added to the ModelManagement.Check package. This is the default style checker, implemented by calling `checkLibrary()` using the default style check setup. The style checker is typically run from the **Check** menu, available in the **Graphics** tab and the **Text** tab.



The user can instead use a custom style check function by setting the flag `Advanced.Check.StyleCheckFunction` to the full path of that Modelica function. The user-defined style checker must be a Modelica function, but it can of course invoke external tools. The style check function should take a single argument, a string with the path name of the model to check. The default value of the flag is

```
Advanced.Check.StyleCheckFunction = "ModelManagement.Check.checkStyle"
```

If the style checker produces a file in the current directory called `<model name>_StyleCheckLog.html`, this file is automatically displayed by the **Graphics > Check > Style** or the **Text > Check > Style** menu command.

Improved display unit handling

The following display units have been added in `displayunit.mos` (inserted in the corresponding groups):

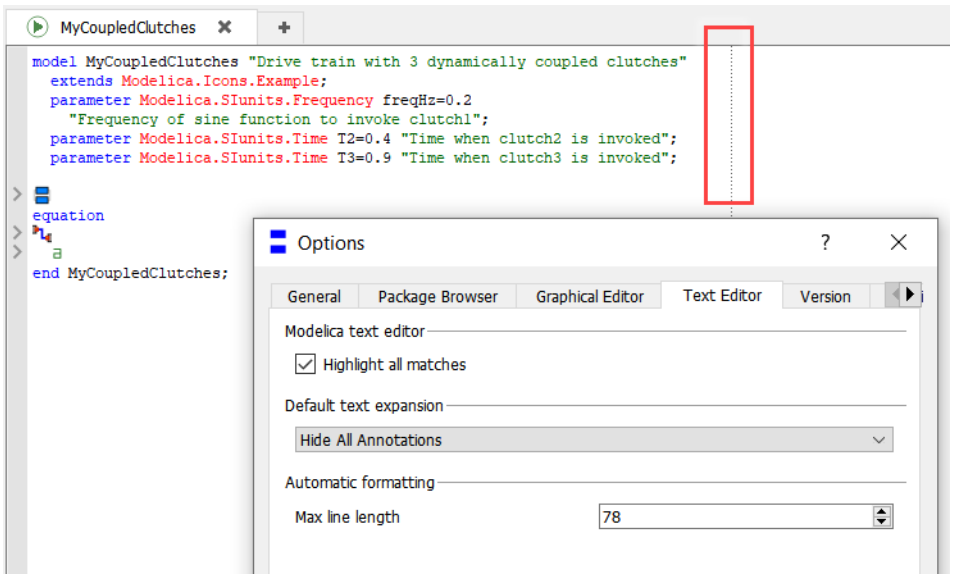
```
// Work, Energy
defineUnitConversion("J", "Wh", 1/3600);
defineUnitConversion("1/J", "1/kWh", 3.6e6);
defineUnitConversion("1/J", "1/MWh", 3.6e9);

// Charge
defineUnitConversion("C", "As", 1);

// Reactive power
```

```
defineUnitConversion("var", "kvar", 1e-3);
defineUnitConversion("var", "Mvar", 1e-6);

// Apparent power
defineUnitConversion("VA", "kVA", 1e-3);
defineUnitConversion("VA", "MVA", 1e-6);

// For Rectification and Chemical engineering
defineUnitConversion("mol/s", "mol/h", 3600);
defineUnitConversion("J/(mol.K)", "J/(kmol.K)", 1000);
defineUnitConversion("s/m", "h/m", 1/3600);
defineUnitConversion("W/(m2.K)", "kW/(m2.K)", 1e-3);
defineUnitConversion("m2/s", "m2/h", 3600);

// Surface-area-to-volume ratio (for reactivity in chemical
reactions)
defineUnitConversion("m2/m3", "1/m", 1);

// various
defineUnitConversion("1", "%", 100);
```

Improved adjustment of maximum line length in the Modelica text editor

The maximum line length in the Modelica text editor is indicated with a dotted line in the editor, framed in the image below. In Dymola 2021x it is easier to change this line length:

- If you move the dotted line, the setting **Max line length** in the **Text Editor** tab, also shown in the image below, is automatically updated with the new value.
- If you change the **Max line length** setting in the mentioned tab and click **OK**, the dotted line is automatically adjusted to that value.



Finding unused parameters

It is possible to find unused parameters when checking the model, translating it, or simulating it. This is done by setting the following flag before performing any of those actions:

```
Advanced.Check.WarnAboutUnreferenced = true
```

(By default, the flag is false.)

Changed highlighting of replaceable and redeclared components

Redeclared components are now only marked if they are *modifiable*, either because they are replaceable, or because the redeclare is in the current model. (Whether the current model is editable is ignored.)

The setting Automatic Manhattanize of connections only available as a flag

Previously a setting **Automatic manhattanize of connections** was available from the command **Tools > Options**, in the **Graphical Editor** tab. This setting has now been replaced by the setting **Automatic routing of connections** (see "Automatic routing when creating connections" on page 9). However, you can still use the flag `Advanced.ManhattanizeConnection` to control if automatic Manhattanize of connections should be active.

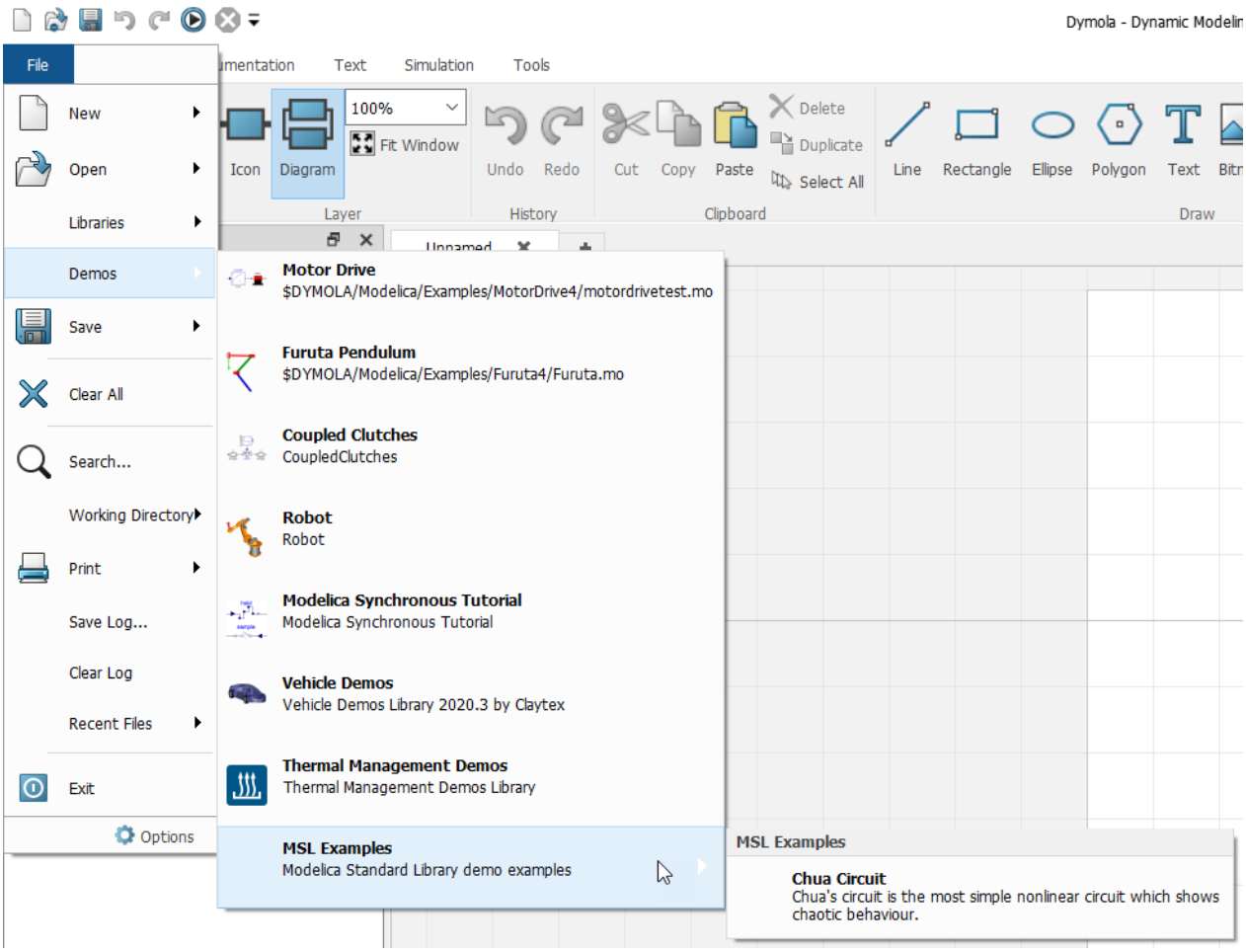
Adding subcategories in the menus File > Library and File > Demos

You can now add subcategories when you define the menus **File > Libraries** or **File > Demos**. This is done by using the optional argument subcategory in the function `LibraryInfoMenuCommand`. This argument can contain a vector of zero to two strings. The first string is the submenu name; the second is a longer description text.

An example of function:

```
LibraryInfoMenuCommand(category="demos",
  subCategory={"MSL Examples", "Modelica Standard Library demo examples"},
  text="Chua Circuit",
  reference="Modelica.Electrical.Analog.Examples.ChuaCircuit",
  isModel=true,
  description="Chua's circuit is the most simple nonlinear circuit which
     shows chaotic behaviour.",
  pos=9999);
```

The result of the command **File > Demos** when using this function in `libraryinfo.mos` is:



3.3 Simulating a model

3.3.1 Improved performance of static simulations

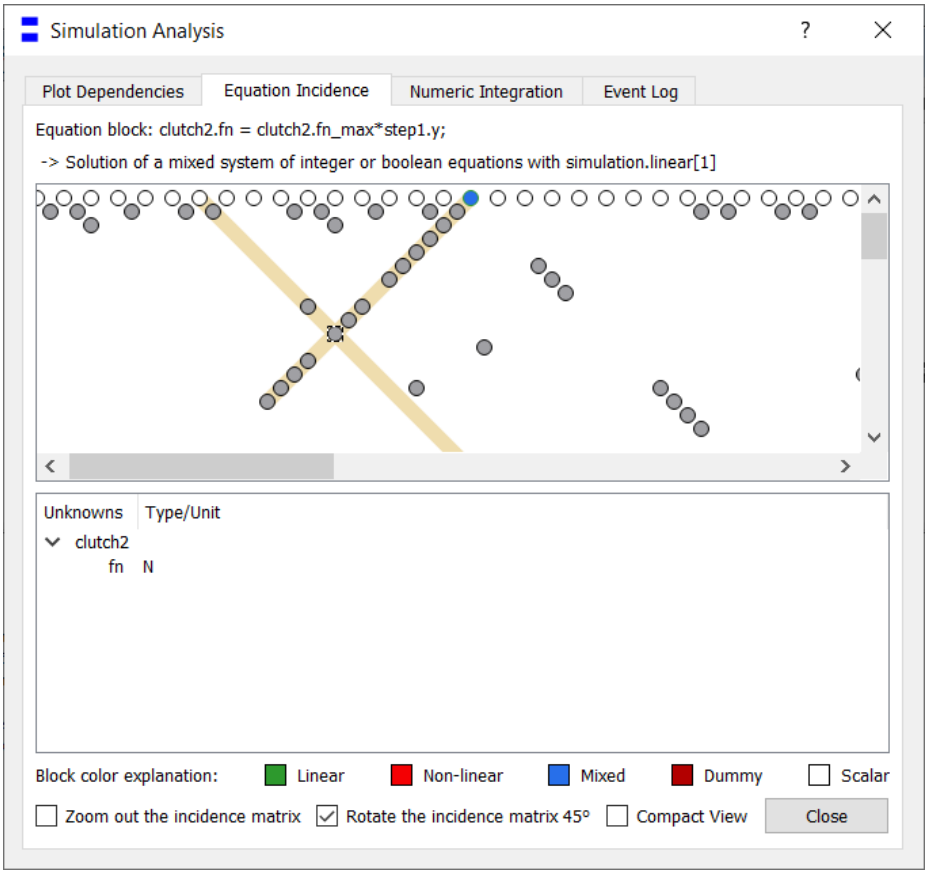
The performance of static simulations has been significantly improved in Dymola 2021x. To enable this, select one of the solvers Lsodar, Dassl, Euler, or any Rkfix solver, and set the stop time to the same value as the start time in the simulation setup (typically 0 to 0). Depending on the model, up to twice as fast simulations may be expected.

3.3.2 Simulation Analysis: Equation incidence graphs

A new tab in the **Simulation Analysis** window displays the equation incidence graphs for the systems of equations in the simulated model.

To be able to display equation incidence graphs, you must activate the setting **Provide variable dependencies and equation incidence for plotting** in the simulation setup. This setting is found using the command **Simulation > Setup**, the **Debug** tab.

The incidence graph is displayed by the command **Simulation > Simulation Analysis**, the **Equation Incidence** tab. An example:



The **Equation Incidence** tab shows the equation blocks. This is the sorted Block Lower Triangular (BLT) form of equation blocks, turned 45 degrees counter-clockwise to make it easy to scroll vertically to follow the diagonal.

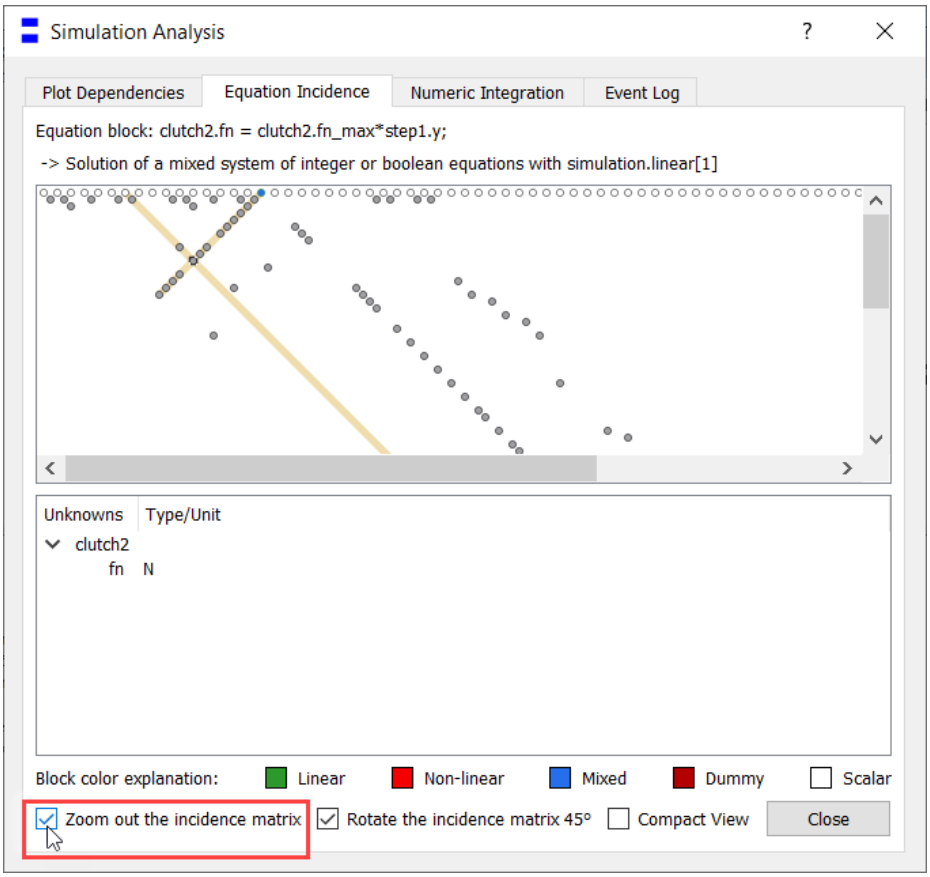
Selecting a “diagonal” block shows the equation blocks, and the variables solved from that block (“Unknowns”), as in the example above. Selecting a “non-diagonal” block shows the two blocks and the “relevant” variables solved from the first block.

The block color has the following meaning:

- Green (**Linear**): Linear system of equations
- Red (**Non-linear**): Non-linear system of equations
- Blue (**Mixed**): Mixed system of equations, border define if linear or non-linear

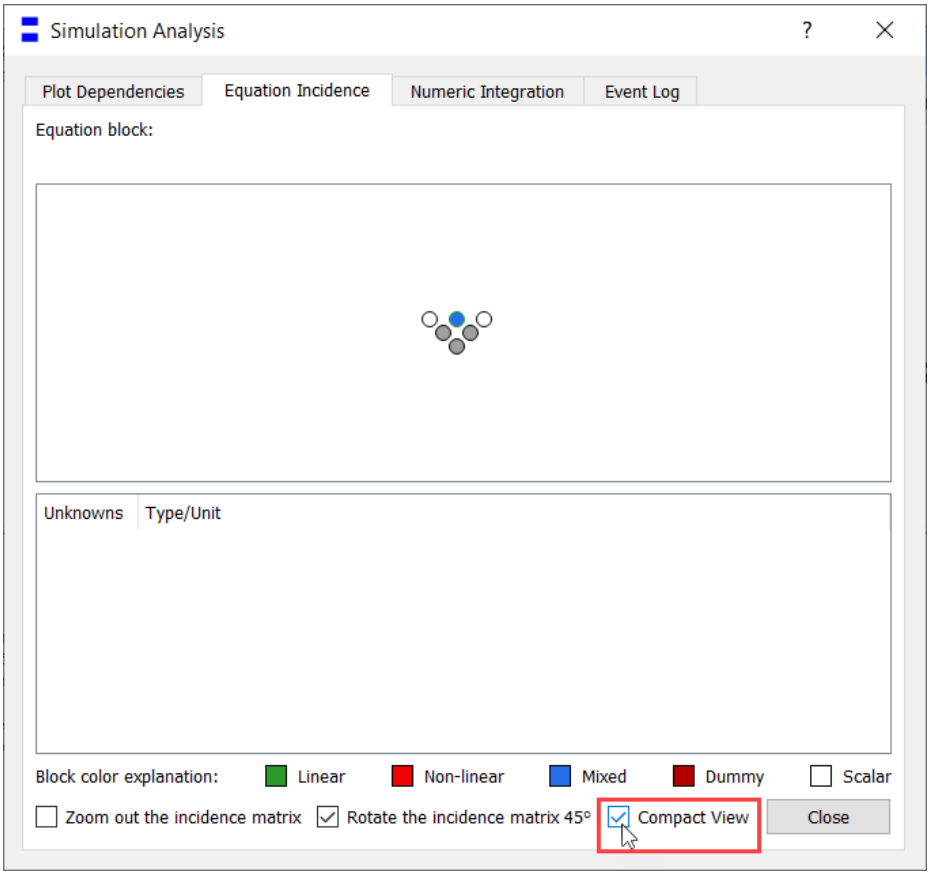
- Dark red (**Dummy**): Dummy derivative system of equations for dynamic state selection
- White (**Scalar**): Simple scalar equations, indicating alias equations

You can zoom out the incidence matrix by activating the option **Zoom out the incidence matrix** at the bottom of the window. The result for the above example is:



By default, the incidence matrix is rotated 45 degrees. You can reset this by deactivating the setting **Rotate the incidence matrix 45**. The result is:

Finally, you can display the incidence matrix in a compact view, that is, displaying the incidence matrix with all simple systems merged, by activating the option **Compact View**. Doing this for the current example gives:

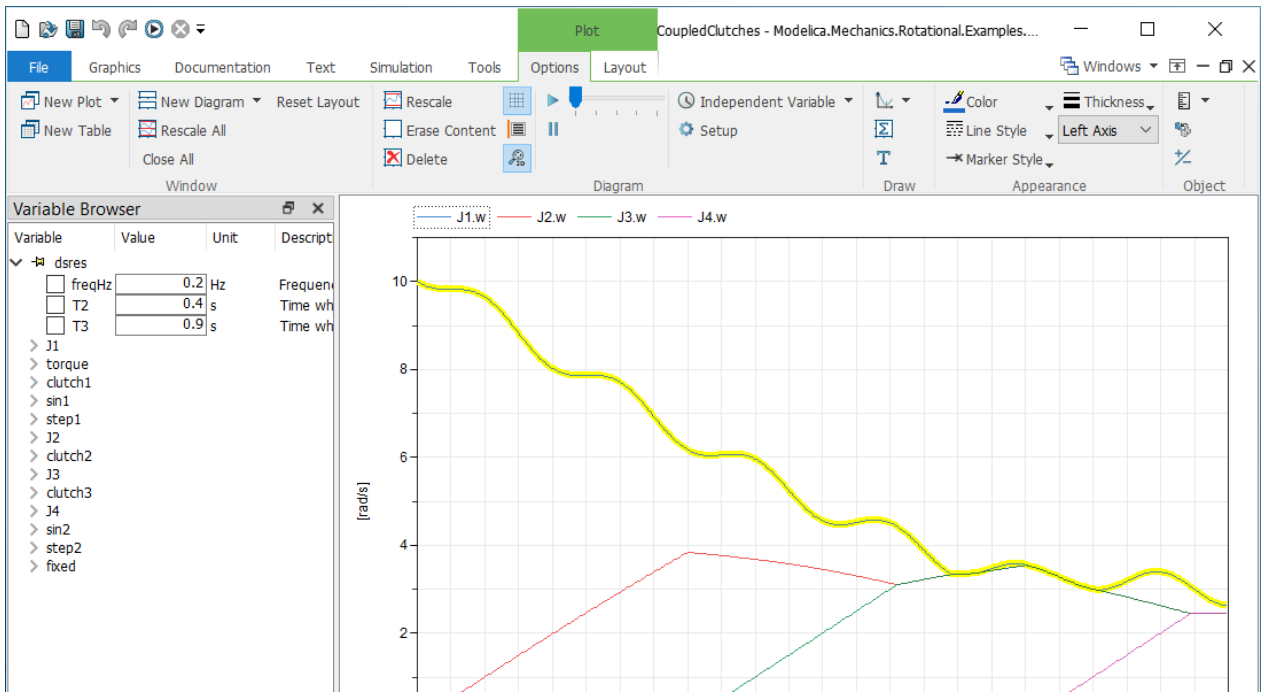

Simulation Analysis
Plot Dependencies
Equation Incidence
Numeric Integration
Event Log
Equation block:
Unknowns
Type/Unit
Block color explanation:
Linear
Non-linear
Mixed
Dummy
Scalar
Zoom out the incidence matrix
Rotate the incidence matrix 45°
Compact View
Close

3.3.3 Plot tab

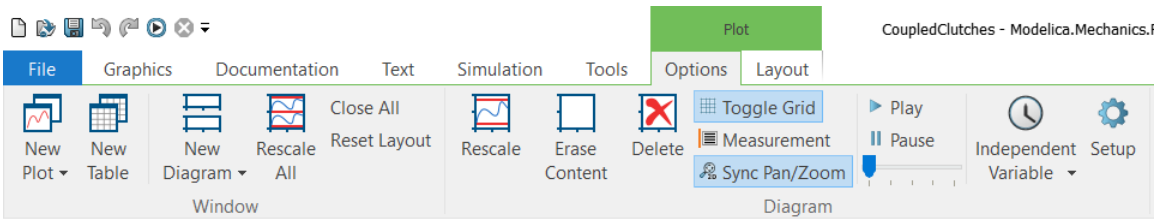
The Plot tab split into two subtabs

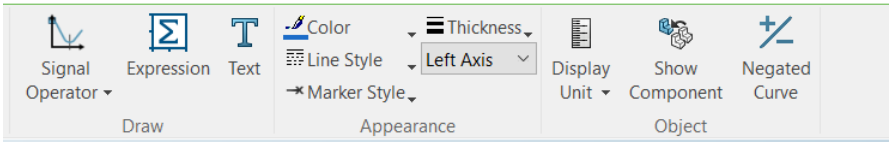
In Dymola 2021x, the plot tab is split into two subtabs. In short, the commands to manage 2D plot layout changes have been moved to a separate subtab.

The Plot: Options subtab

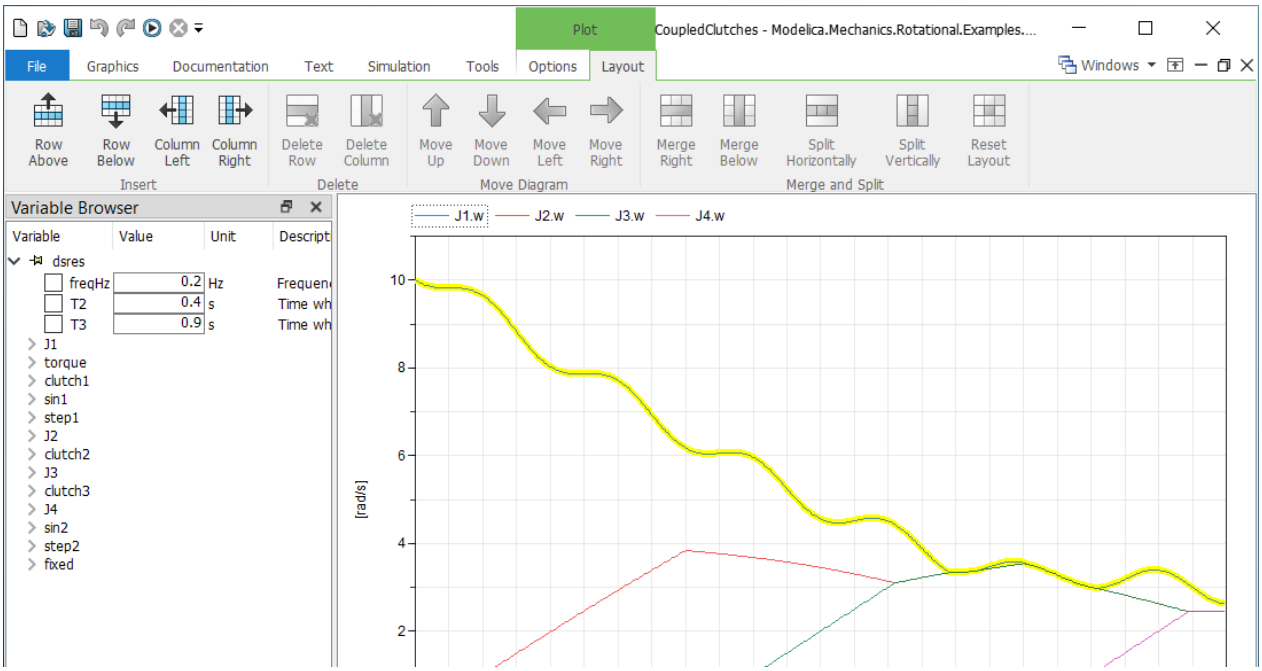


This tab, which is the default tab to enter, contains all plot commands except the ones for managing 2D plot layout changes. Using full screen, the tab looks like (it is divided in two images to have it reasonably large):





The Plot: Layout subtab



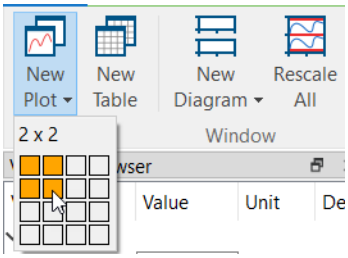
This tab, which you have to select from the **Plot: Options** tab, contains commands for managing 2D plot layout changes.

Simplified creation of 2D plots

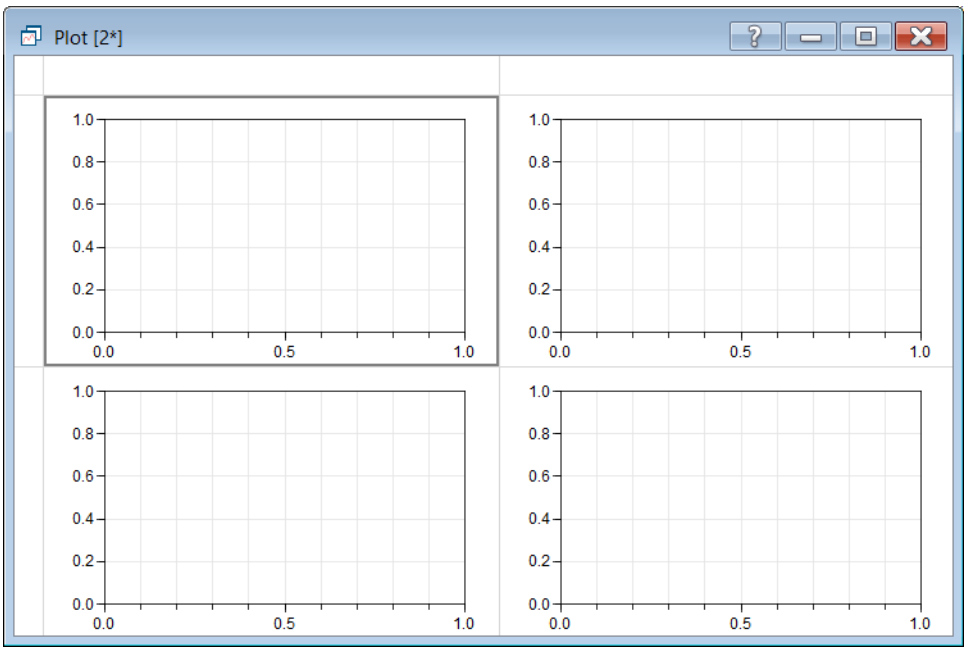
Already in previous version of Dymola, you could you could create a 2D plot layout in two new ways:

Creating a new plot window with 2D plot layout

Using the arrow down on the command **Plot: Options > New Plot**:



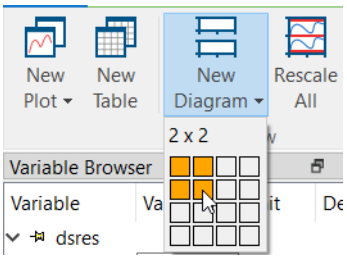
The command displayed will create a new plot window with four diagrams like:



Note that the previous behavior, creating a new window, is still present by clicking the button instead of the arrow.

Changing an existing plot window to a 2D plot layout

Using the arrow down on the command **Plot: Options > New Diagram**:



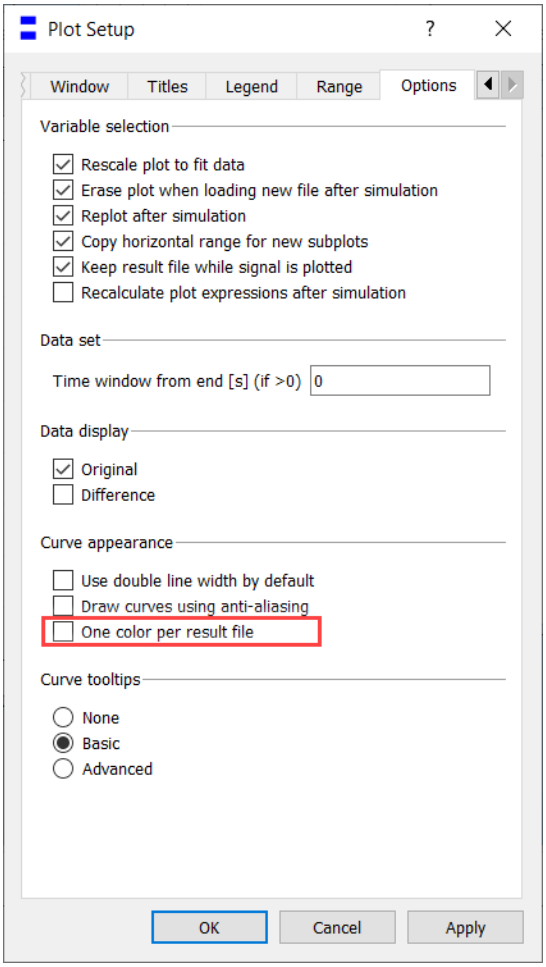
In this case, you change the layout of the currently active plot window to the selected 2D layout.

Note that the previous behavior, creating a vertical split, is still present by clicking the button instead of the arrow.

Curve colors controlled by simulation result files

In Dymola 2021x you can apply a color scheme that assigns a unique color for curves based on the result file, so all curves from one result file gets one color, and all signals from the next result file get another color.

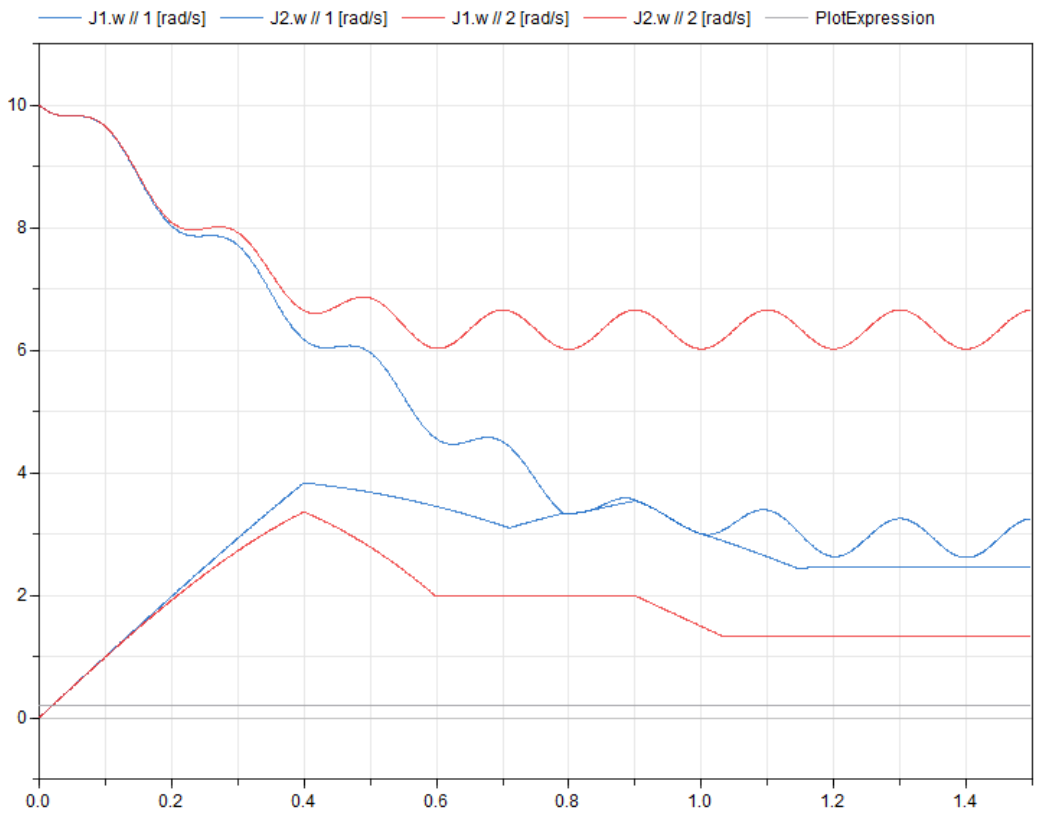
The feature is controlled by the setting **One color per result file**. The setting can be reached by the command **Plot: Options > Setup**, the **Options** tab:



The setting is by default not activated. This corresponds to the flag `Advanced.Plot.ResultColors = false`. The setting is saved between sessions.

Note! This feature is an option when displaying curves, but when using the command **Compare Results** in the variable browser, the feature is always applied, independent of if the feature is activated or not for other cases.

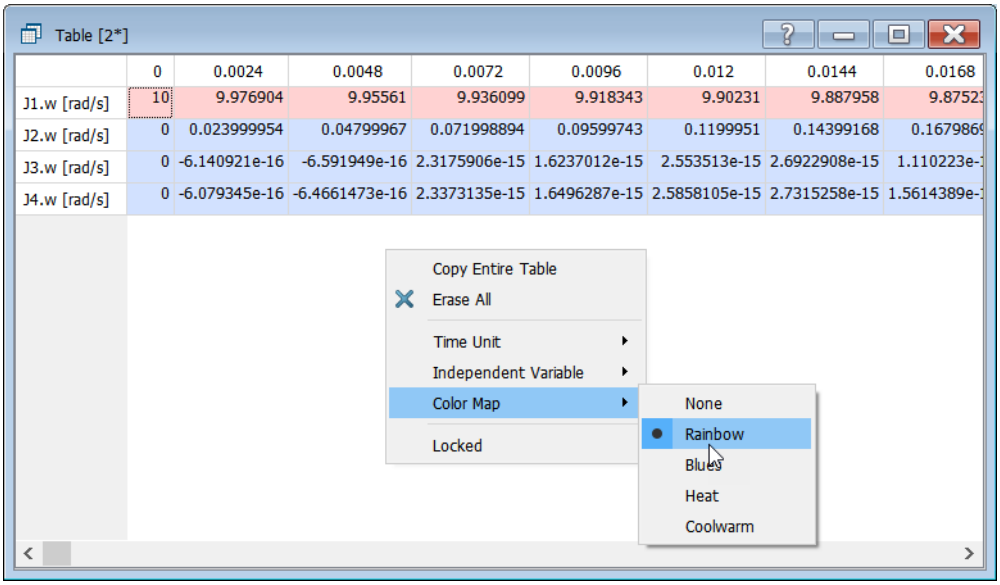
An example of the feature:



An example of a plot expression is added in the image above; curves that does not belong to a result file, for example plot expressions, are displayed in gray color.

Applying Color-coding to result file tables

You can apply color-coding of table cells to result file tables to indicate approximately the values. To activate the feature, right-click inside the table, in a cell or in empty space like below, and select **Color Map > <theme>**. An example:



By default, this feature is not active (**None** is the default selection).

Improvements for predefined plots

Dymola 2021x supports **Annotations for Predefined Plots**, which are proposed as an extension of the Modelica Specification (MCP-0033), but the following should be noted:

- Figure captions are placed in Dymola´s Plot Documentation (the "?" button in the plot window title bar)
- Empty legends are not supported
- Variable replacements and links in text are not supported

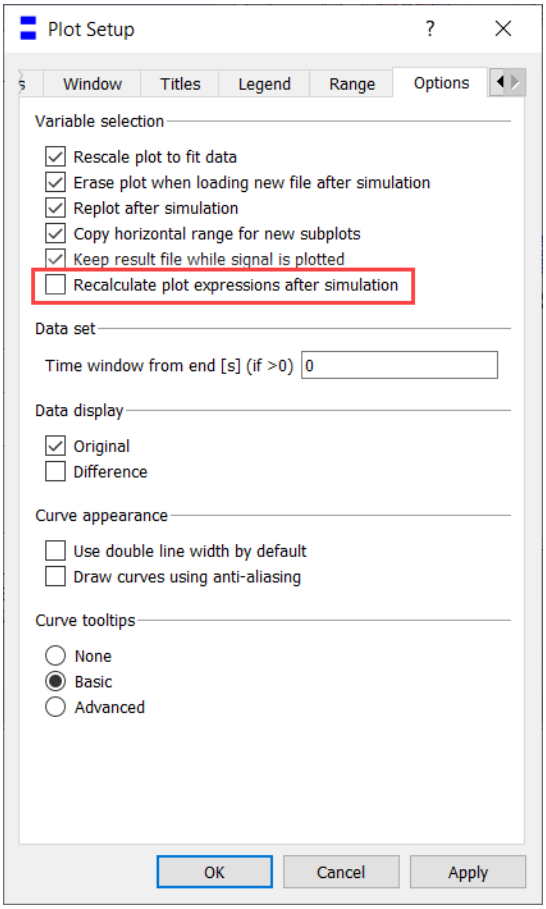
Setting units on plot expression plots

To be able to set unit/display unit on plot expressions, there is now a new parameter for the built-in function `plotExpression`. The new parameter is:

```
input String unit = "" "Unit.
   If the string has the format 'unit|displayunit' it sets the
   display unit too. For example, 'rad|deg'.";
```

Re-calculations of plot expressions after simulation

There is a new option available in the plot setup to select if plot expression curves should be automatically recalculated after simulation:



The option is reached by the command **Plot: Options > Setup**, the **Options** tab. The option is by default not active. The default setting of the option corresponds to the flag

```
Advanced.Plot.AutoUpdatePlotExpressions = false
```

The option is saved between sessions.

3.3.4 Animation window

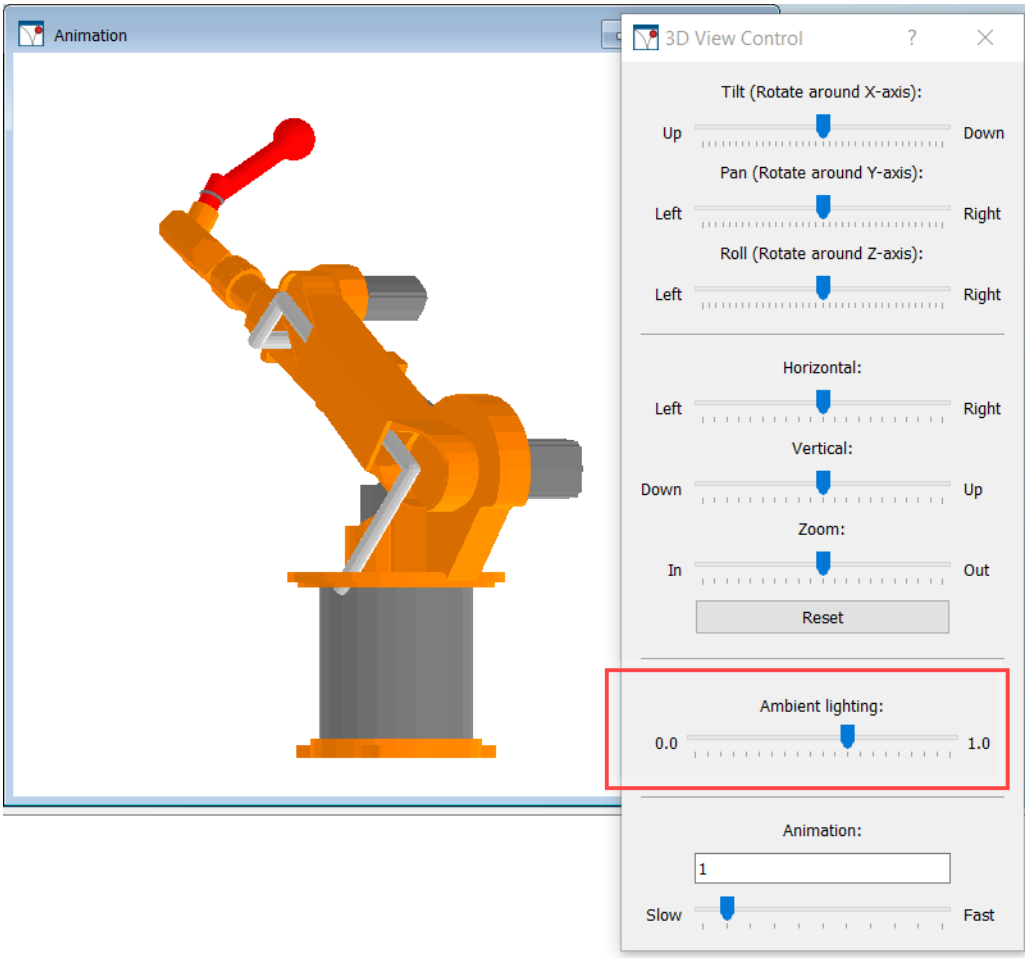
Improved lighting model

The lighting model of the animation window has been improved by extended lighting. The extended lighting is by default activated, but the previous lighting model can be implemented by setting the flag

```
Advanced.Animation.ExtendedLighting = false
```

(The flag is by default `true`.)

If you use extended lighting or if the model has been created by the built-in function `animationLightning` you can use a new slider in the command **Animation > 3D View Control** to set the ambient lighting:



3.3.5 Scripting

Improved built-in function plotExpression

See section “Setting units on plot expression plots” on page 29.

Improved built-in function importFMU

The built-in function `importFMU` now contains a new argument for handling filtering of signals when importing an FMU:

```
input String includeVariables[:] = fill("", 0) "FMU variables
to be included in the fmuWrapper, an empty list will include
all variables";
```

The default value is an empty array. If this is not changed, the argument `includeAllVariables` in the built-in function will be used to specify what variables are included in the generated FMU, as in previous versions.

If however variables are added in this array these variables will be included in the generated FMU. In this case, the argument `includeAllVariables` in the built-in function is ignored.

These improvements of the built-in function `importFMU` corresponds to the new GUI when importing an FMU, see section "Filtering signals when importing an FMU" on page 48.

3.3.6 Improved simulation logging

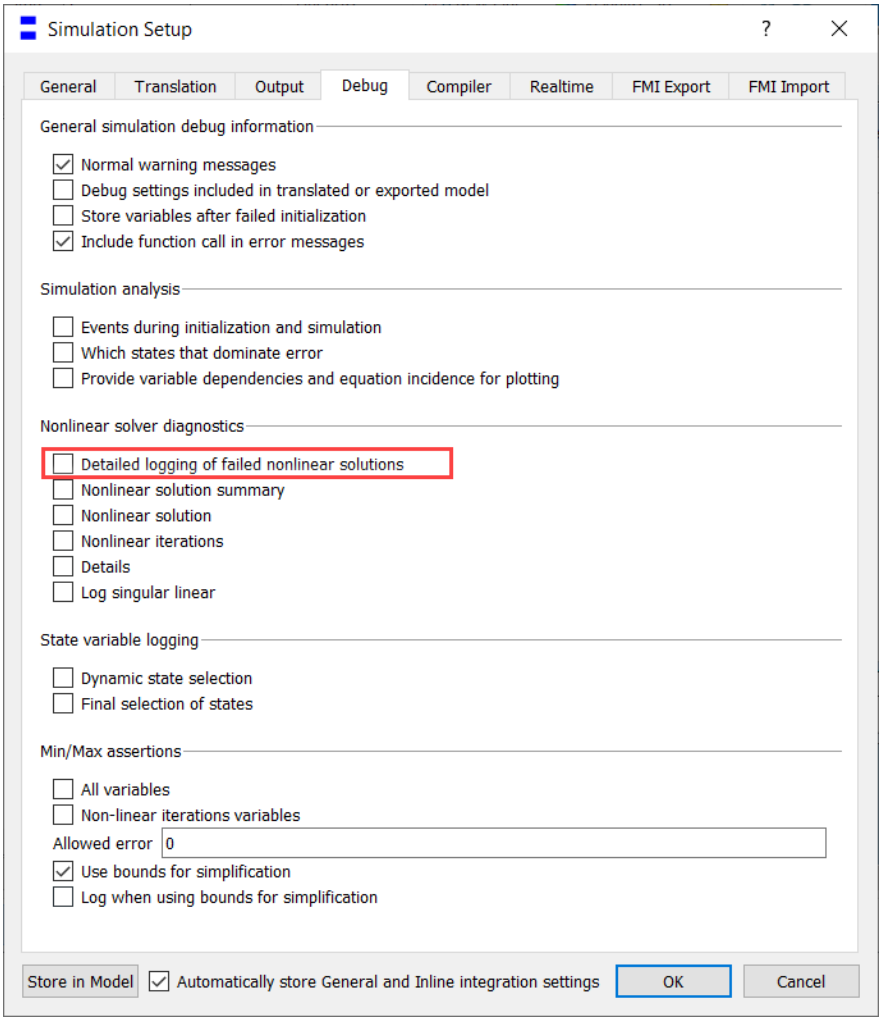
More general information

At the end of the simulation log, a message is now printed, clearly stating whether the simulation was successful or if it failed. The message also states if the simulation was terminated `terminate()` by the operator. The message also contains the full path to the model being simulated.

The full path to the model being simulated is also given when the simulation is started.

Improved nonlinear solver diagnostics

You can now select how detailed the logging of failed nonlinear solutions should be, by using the new option **Detailed logging of failed nonlinear solutions**:



The setting is reached by the command **Simulation > Setup**, in the **Debug** tab.

If you activate the setting, you get the same detailed information about failed nonlinear solutions as in previous versions.

The setting is by default not activated, corresponding to the flag `Advanced.Debug.LogNonlinearFailedSolutions=false`.

The setting is saved between sessions.

An example of the first part of a simulation log in Dymola 2021x with the default (not activated) setting:

```
Log-file of program ./dymosim
(generated: Wed Aug 26 10:49:02 2020)

dymosim started
... "NonlinearEquationSolve" simulating
... "dsin.txt" loading (dymosim input file)
... "NonlinearEquationSolve.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dassault Systemes))

Warning: Failed to solve nonlinear system using Newton solver.
  Time: 0.4996100837250252
  Tag: simulation.nonlinear[1]
  For debugging information enable
  Simulation/Setup/Debug/Nonlinear solver diagnostics/Detailed logging of failed nonlinear solutions.
Solver will attempt to handle this problem.

Warning: Failed to solve nonlinear system using Newton solver.
  Time: 0.4939881859511698
  Tag: simulation.nonlinear[1]
Solver will attempt to handle this problem.


Warning: Failed to solve nonlinear system using Newton solver.
  Time: 0.4911772370642422
  Tag: simulation.nonlinear[1]
Solver will attempt to handle this problem.
```

If the setting **Detailed logging of failed nonlinear solutions** is instead activated, the framed message above will instead be:

```
Warning: Failed to solve nonlinear system using Newton solver.
  Time: 0.4996100837250252
  Tag: simulation.nonlinear[1]

  Common causes:
   * The system of equations has no solution - the residual will be above zero.
     - In some cases the event-logic can cause this.
   * Starting values are too far from the solution.
     - In rare cases this could occur at events.
   * The equations are too discontinuous for the nonlinear solver - the residual will have knees.
     - Likely caused by over-using noEvent.

  To get more information consider the options:
   * Simulation/Setup/Translation/Generate listing of translated Modelica code in dsmodel.mof
   * Simulation/Setup/Translation/List non-linear iteration variables
   * The options under the group Simulation/Setup/Debug/Nonlinear solver diagnostics

  Jacobian inverse norm estimate: 516840
  Condition number estimate: 213378
  1-norm of the residual = 0.0133969

  Last value of the solution:
    der(z) = -0.358241
    der(x) = -1.09532
  Last value of the residual:
    { -0.00363467, -0.00976219 }

Solver will attempt to handle this problem.
```

Notes:

- In Dymola 2021x, messages about failed nonlinear solutions are displayed as warnings; previously they were displayed as errors.
- The log of the first failed attempt of solving nonlinear equations always contain tips on how to display more information.

3.3.7 Display of browsers and other windows controlled from the status bar

Please see the corresponding section on page 15.

3.3.8 Minor improvements

Improved event logging – selecting what events to log

By default, all events are logged when the event logging option is enabled. You can now disable logging of time events, state events, or step events, during the simulation, by setting the following flags:

- Set `Advanced.Simulation.Debug.LogTimeEvents = false` to disable logging of time events during the simulation.

- Set `Advanced.Simulation.Debug.LogStateEvents = false` to disable logging of state events during the simulation.
- Set `Advanced.Simulation.Debug.LogStepEvents = false` to disable logging of step events during the simulation.

(All the flags are by default `true`.) You must set the flags before simulating. The choice affects both the simulation log and the event log in the Simulation Analysis feature. Changing any of these flags forces re-translation of the model.

An example of a use-case is simulations with frequent sample events resulting in large logs. Disabling time event logging makes it easier to find the state events.

Note. The events during initialization are not filtered with any of the above flags; to filter out those events you can set the flag `Advanced.Debug.LogEventsInitialization=false`. (The idea with the three flags above is to exclude based on what triggers an event during simulation: if the triggering event is a time, state, or step event. As initialization is none of these three categories, events during initialization cannot be excluded by these flags.)

Improvements of the command Save in Model

When you perform the command **Simulation > Save in Model**, you are asked if you want to save modifications of start values having a prefix `final` and start values of protected variables. If final start values are modified, Dymola may issue an error when translating the resulting model as the modifications may not be conforming to the Modelica specification. To turn this error message off set

```
Advanced.Translation.Log.FinalStartModifierError = false.
```

Changing the start values of protected variables only gives a warning by default.

The following model can be used to test this:

```
model ProtectedAndFinalVariables
  Real x(final start=1);
protected
  Real y;
initial equation
  x + exp(x) = 2;
  y + exp(y) = 2;
equation
  der(x) = 1;
  der(y) = 1;
end ProtectedAndFinalVariables;
```

Dragging multiple result files into Dymola window to load them

You can now drag multiple result files into the Dymola window to load them.

Calling functions without displaying the function dialog for functions in the Commands menu or similar

It is possible in Dymola 2021x to execute a function without displaying the function dialog for functions in the **Commands** menu (or similar menus, like **Tools > Linear Analysis**) if the

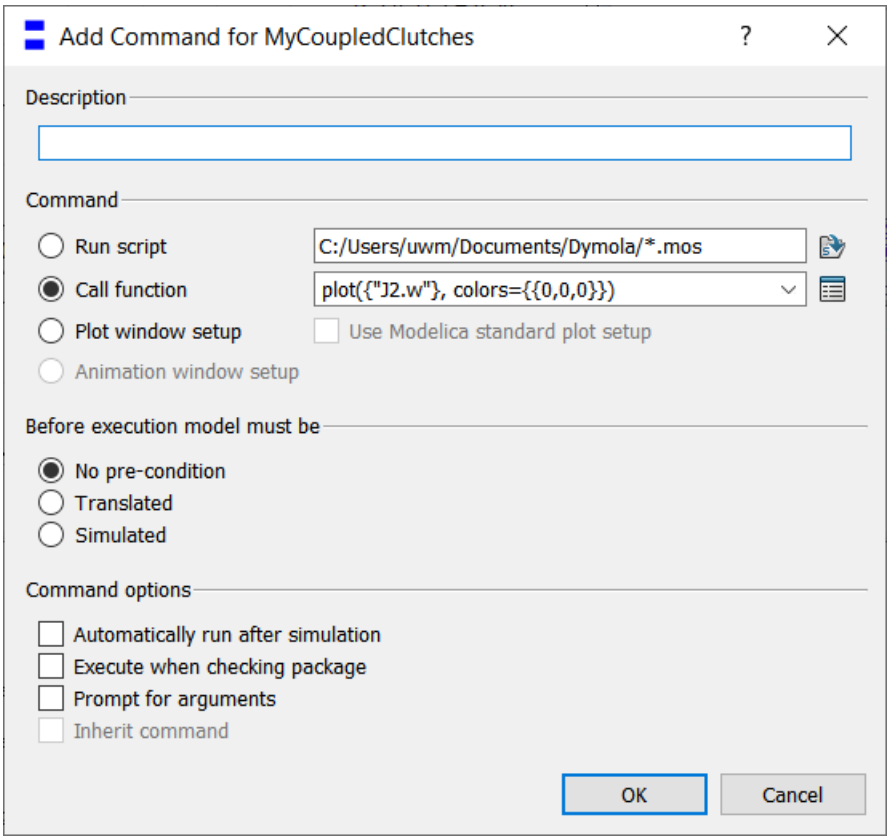
function has no input parameters, or if all input parameters have default values. This is obtained by adding the annotation `__Dymola_autoExecute=true` to the function.

Notes:

- This functionality is not supported when calling functions in any other way.
- The annotation is best seen as "pressing **OK** automatically" rather than "skip launching the function call window" – the adding of the current model-name to the function is still handled.
- For advanced users, note that you can override the annotation by going to the command log, right-click the function call, and select **Edit Function Call**.

Revised layout of Add Commands dialog

The layout of the **Simulation > Add Command...** dialog has been revised:

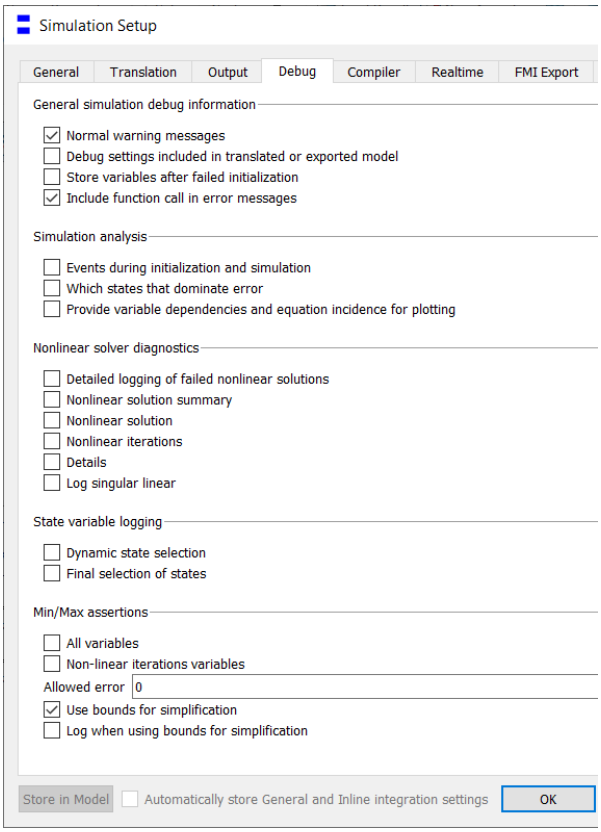


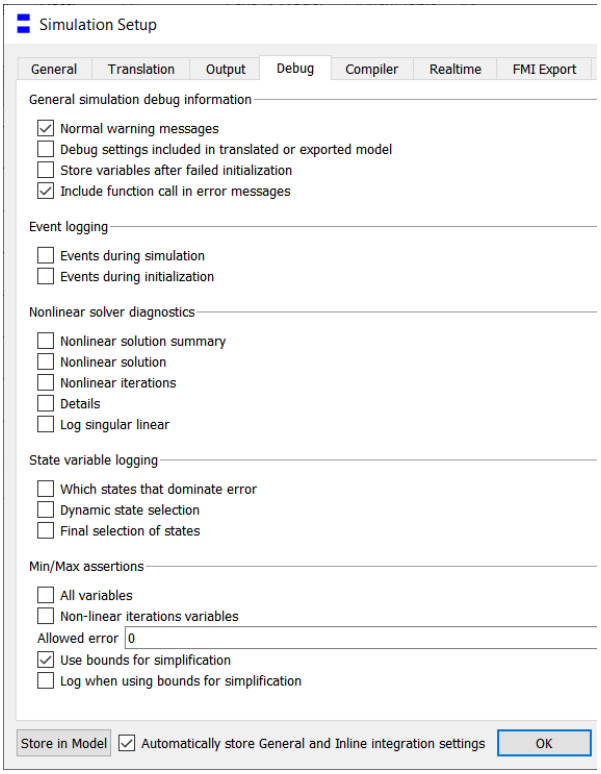
The changes, which only affect the layout of the dialog, and not its functionality, are:

- The description field has been moved to the top of the dialog, because it is required to provide a reasonable menu choice.
- The command options have been grouped at the end, to clarify that they are of general nature.

Revised Debug tab of the Simulation setup dialog

[389] The **Debug** tab of the simulation setup dialog has been revised. (The dialog can be reached by the command **Simulation > Setup**, the **Debug** tab.) Below to the left, the new dialog, and, for comparison, to the right, the dialog in Dymola 2021.





The changes are:

- The former group **Event Logging** has been renamed to **Simulation Analysis**
- In this group, the former two settings **Events during simulation** and **Events during initialization** have been merged to **Events during initialization and simulation**. Note that if you want to change some of them separately, you can still use the corresponding flags.
- The setting **Which states that dominate error** has been moved from the **State variable logging** group to the **Simulation analysis** group
- The former setting **Provide variable dependencies for plotting** that was located in the **Translation** tab in Dymola 2021 has been moved to the **Debug** tab, the **Simulation analysis** group in Dymola 2021x. It has also been renamed to **Provide variable dependencies and equation incidence for plotting** due to being a requisite also for

the new feature of displaying equation incidence graphs (see section “Simulation Analysis: Equation incidence graphs” on page 19).

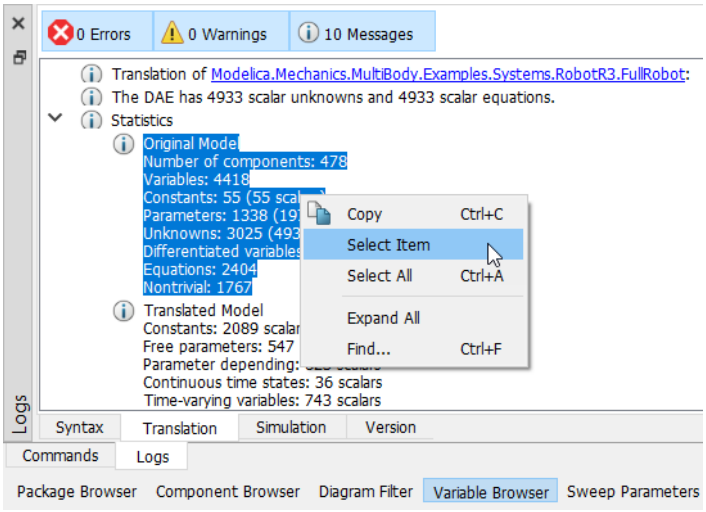
- In the group **Non-linear solver diagnostics**, a new setting **Detailed logging of failed nonlinear solutions** has been added. See section “Improved nonlinear solver diagnostics” starting on page 32.

Improving parallel code simulation outside of Dymola

If you run a parallel code simulation (parallelizing the model using the flag `Advanced.ParallelizeCode=true`) outside of Dymola by using, for example, the command prompt, you can speed up the parallel simulation by setting the flag `OMP_WAIT_POLICY=PASSIVE` before running the simulation. This flag was available already in Dymola 2020.

Context menu command for selecting the active node in the translation log

A new context command **Select Item** in the translation log selects the active node:



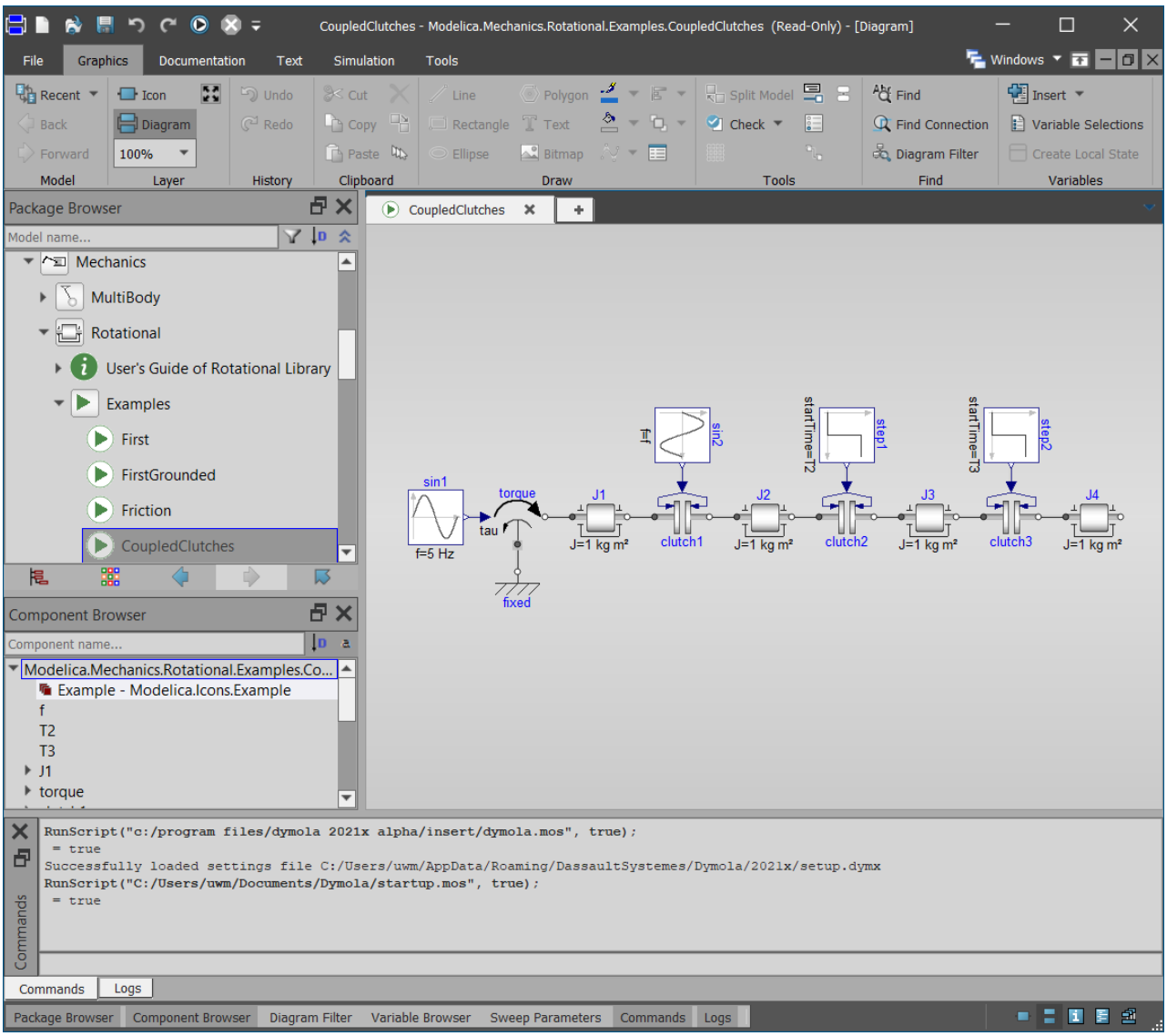
This can be useful when wanting to copy a node that might be large, for sharing information.

3.4 Installation

For the current list of hardware and software requirements, please see chapter “Appendix – Installation: Hardware and Software Requirements” starting on page 55.

3.4.1 Support for Dymola in “dark mode”

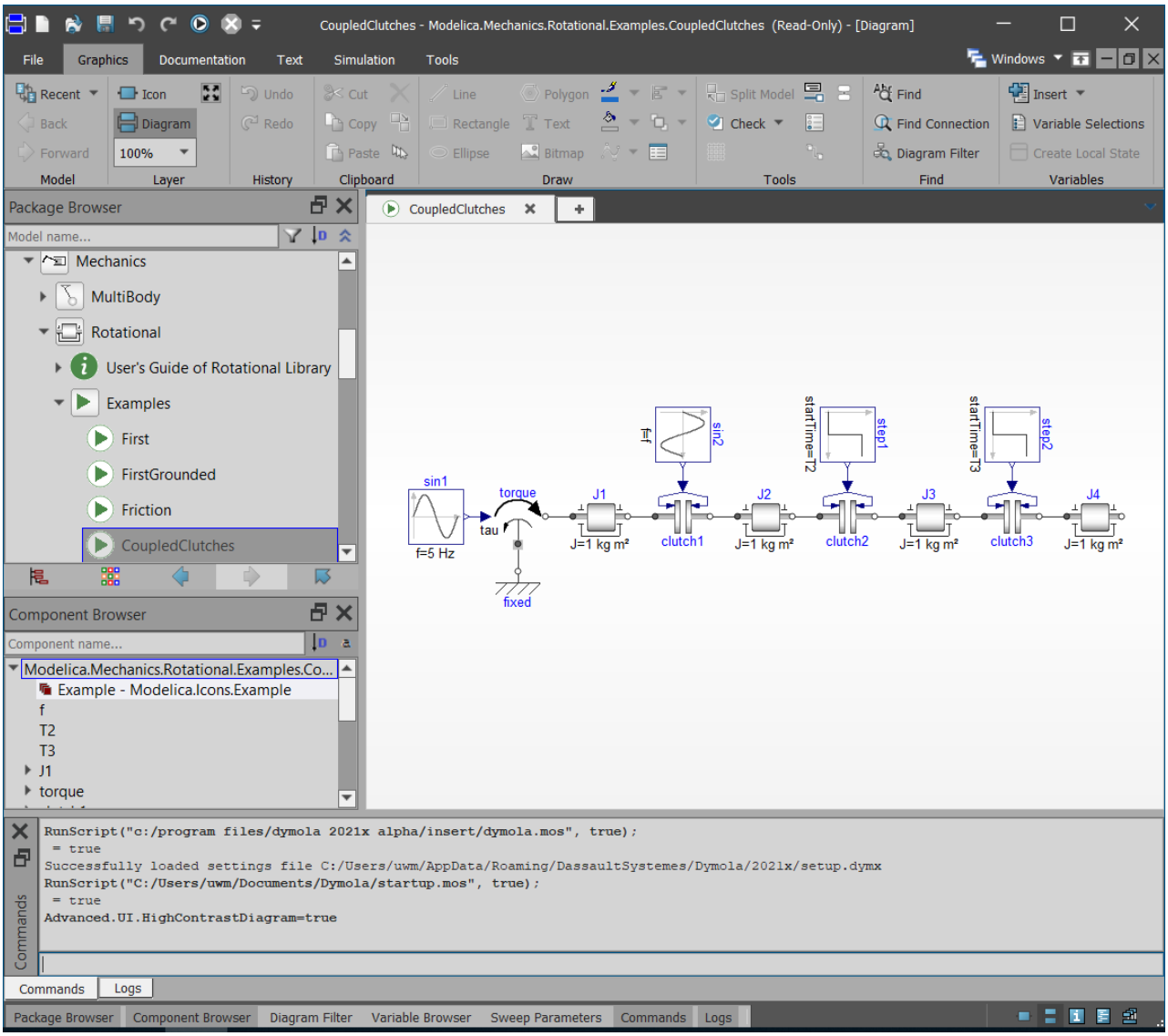
Dymola 20201x can be started with the command line switch `-dark` or `/dark`. This will start Dymola with an alternative darker color theme:



To enable the traditional white (high contrast) background in the graphical editor also in "dark" mode, you can set the flag

```
Advanced.UI.HighContrastDiagram = true
```

(The flag is by default `false`.) This can help if a model is hard to see against the grey background:



3.4.2 Installation on Windows

For the full list of supported compilers, see “Compilers” on page 56.

Updated Qt version

Dymola 2021x is built with Qt 5.15.0.

GCC compilers

In Dymola 2021x, the support is discontinued for MinGW GCC compilers with versions lower than 5. For supported and tested versions, see “Compilers” on page 56.

3.4.3 Installation on Linux

Updated Qt version

Dymola 2021x is built with Qt 5.15.0.

3.4.4 Dymola license server on Windows and Linux

Dassault Systèmes License Server (DSLS) for Dymola

As an alternative to the FLEXnet license server, Dassault Systèmes License Server (DSLS) is now also supported for Dymola, for sharable and nodelocked licenses.

Some notes about installation and use:

- To generate a target (host) id to order a license key, start Dymola with the command `dymola.exe /DSLS` (without license) and then, in Dymola, use the command **Tools > License Setup > Details** to copy the target id.
- A nodelocked DSLS license key should be saved in:
    - On Windows: `C:\ProgramData\DassaultSystemes\Licenses`.
    - On Linux: `var/DassaultSystemes/Licenses`
- The DSLS license server can be downloaded from the same location as the Dymola media. After downloading, you can install it as described in the DSLS documentation.
- To start Dymola with DSLS, use the command `dymola.exe /DSLS`
- To connect to an existing DSLS license server, use, in Dymola, the command **Tools > License Setup > Setup**. Type in the server name and press **OK**. The default port number can be used in almost any case.

Limitations:

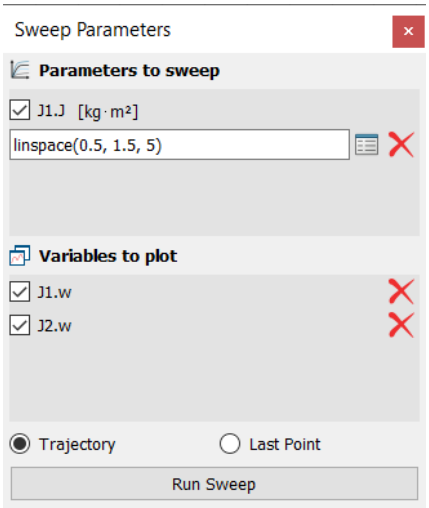
- Borrowing of licenses (the Dymola command **Tools > License Setup > Borrow**) is not supported.

For more details, see the Knowledge Base article QA00000061268. Note that a DS Passport is needed to see this article.

3.5 Model Experimentation

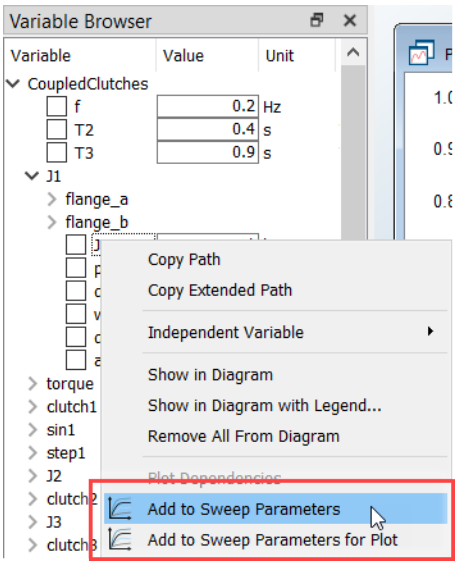
3.5.1 Improved GUI for sweeping parameters

The dialog for selecting which variables to sweep/plot now looks like:

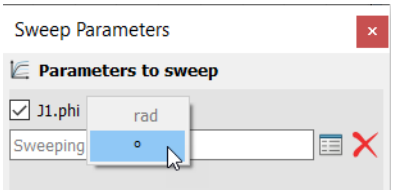


The panel is now divided into two sections, one for parameter to sweep and one for parameters to plot. A parameter can be in both sections.

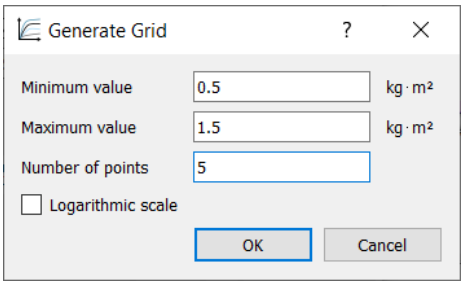
According to this, there are now two commands to add a variable to the Sweep Parameters panel (dragging them also works):



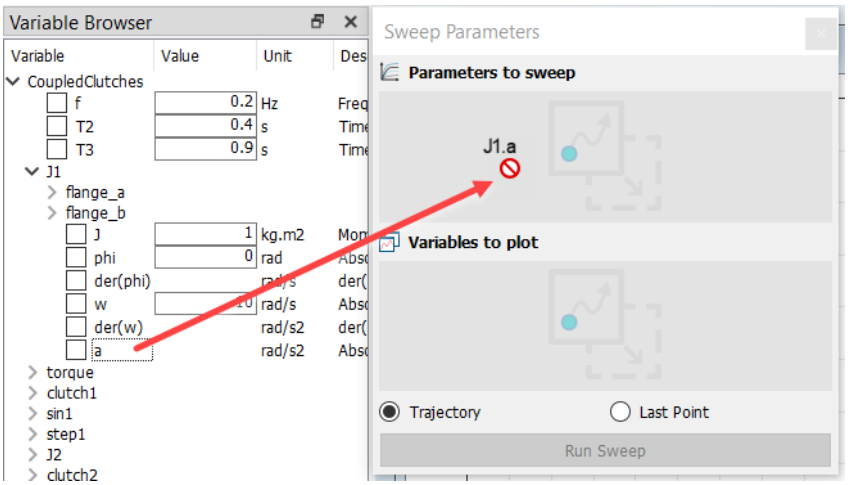
Units are displayed for parameters to sweep. You can change the display units:



The units are also displayed, and changeable if possible, when specifying how to generate the grid:



In Dymola 2021x, if you cannot sweep a variable, it cannot be added to the Parameters to sweep pane. As an example, trying to drag J1.a from the package browser to this pane gives:



The corresponding context menu in the package browser (see above) is grayed out.

3.6 Other Simulation Environments

3.6.1 Dymola – Matlab interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R2015a (ver. 8.5) up to R2020a (ver. 9.8). On Windows, only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. On Linux, the gcc compiler is supported. The LCC compiler is not supported, neither on Windows nor on Linux.

3.6.2 Real-time simulation

Compatibility – dSPACE

Dymola 2021x officially supports the DS1005, DS1006, MicroLabBox, and SCALEXIO systems for HIL applications. For these systems, Dymola 2021x generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases:

- dSPACE Release 2015-A with Matlab R2015a
- dSPACE Release 2015-B with Matlab R2015b
- dSPACE Release 2016-A with Matlab R2016a
- dSPACE Release 2016-B with Matlab R2016b
- dSPACE Release 2017-A with Matlab R2017a
- dSPACE Release 2017-B with Matlab R2017b
- dSPACE Release 2018-A with Matlab R2018a
- dSPACE Release 2018-B with Matlab R2018b
- dSPACE Release 2019-A with Matlab R2019a
- dSPACE Release 2019-B with Matlab R2019a and R2019b
- dSPACE Release 2020-A with Matlab R2019a, R2019b, and R2020a

The selection of supported dSPACE releases focuses on releases that introduce support for a new Matlab release and dSPACE releases that introduce a new version of a cross-compiler tool. In addition, Dymola always support the three latest dSPACE releases with the three latest Matlab releases. Although not officially supported, it is likely that other combinations should work as well.

New utility functions – dym_rti_build2 and dym_rtimp_build2

Dymola 2021 introduced a new function, `dym_rti_build2`, that replaces `dym_rti_build` for building dSPACE applications from models containing DymolaBlocks. The new function uses the new dSPACE RTI function `rti_build2` instead of the old function `rti_build`.

A corresponding new multi-processor build function, `dym_rtimp_build2`, is also introduced.

These functions are supported with dSPACE Release 2019-B and later.

Note on dym_rti_build and dSPACE Release 2017-A and later

The function `rti_usrtrcmerge` is no longer available in dSPACE Release 2017-A and later. As a consequence, it is required to run the standard `rti_build` function (with the 'CM' command) after `dym_rti_build` to get your _usr.trc content added to the main .trc file. For example:

```
>> dym_rti_build('myModel', 'CM')
>> rti_build('myModel', 'Command', 'CM')
```

Note that this note applies the new functions `dym_rti_build2` and `rti_build2` as well.

Compatibility – Simulink Real-Time

Compatibility with Simulink Real-Time has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R2015a (Simulink Real-Time ver. 6.2) to R2020a (Simulink Real-Time ver. 6.12). Only Microsoft Visual C compilers have been tested.

3.6.3 DDE communication

Extended solver support for DDE server

In Dymola 2021x, DDE server is supported for all solvers.

3.6.4 OPC communication

Discontinued OPC communication support

From this version, Dymola 2021x, OPC communication is not supported. For alternatives, please contact support: https://www.3ds.com/support.

3.6.5 Java, Python, and JavaScript Interface for Dymola

A number of new and improved built-in functions are available in the interfaces.

For more information, see the corresponding section in "Scripting" on page 31.

3.6.6 FMI Support in Dymola

Unless otherwise stated, features are available for both FMI version 1.0 and version 2.0.

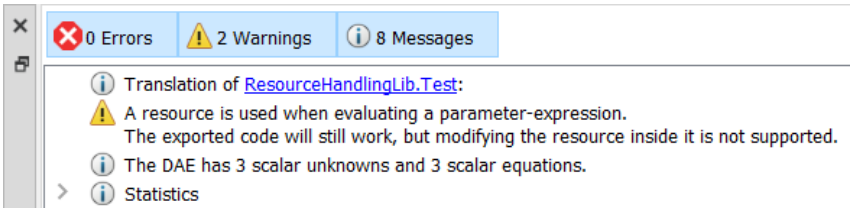
FMU Export

Improved information in the translation log about handling of resources

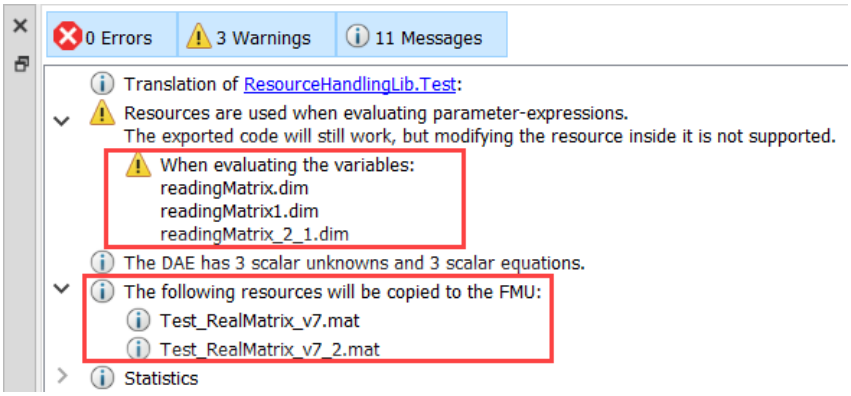
Even if you select to copy resources to the FMU when exporting it, you cannot change structural parameters depending on those resources after the FMU is created. The reason is that Dymola must evaluate the structural parameters in the beginning of the translation of the model to the FMU, by reading the resources. Structural parameters involve for example sizes

of vectors and matrices. The only way to change such sizes is to recreate the FMU after changing the resources.

The information about such calculations was already previously listed in the translation log:



In Dymola 2021x, the names of the structural parameters that have been evaluated in such a way are also listed when expanding the message:



Additionally a message is given in the translation log listing all files that will be copied to the FMU (the second framed message above)

Appending unique id to file name all.c

The extra source file `all.c` that includes all other C files (needed when compiling multiple FMUs source code as one unit) can now be created with a unique id (modelidentifier) appended to the name, to avoid multiple instances of `all.c` files. As an example, having a model identifier `MyModel`, the corresponding sorce file will be `all_MyModel.c`.

You activate this by setting the flag

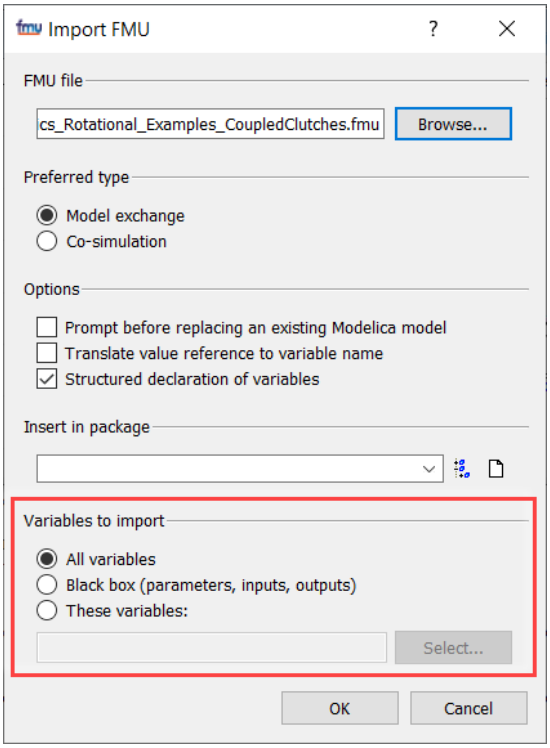
```
Advanced.FMI.FMUSourceCodeUniqueNaming = true
```

The flag is by default `false`. The flag value is stored between sessions.

FMU Import

Filtering signals when importing an FMU

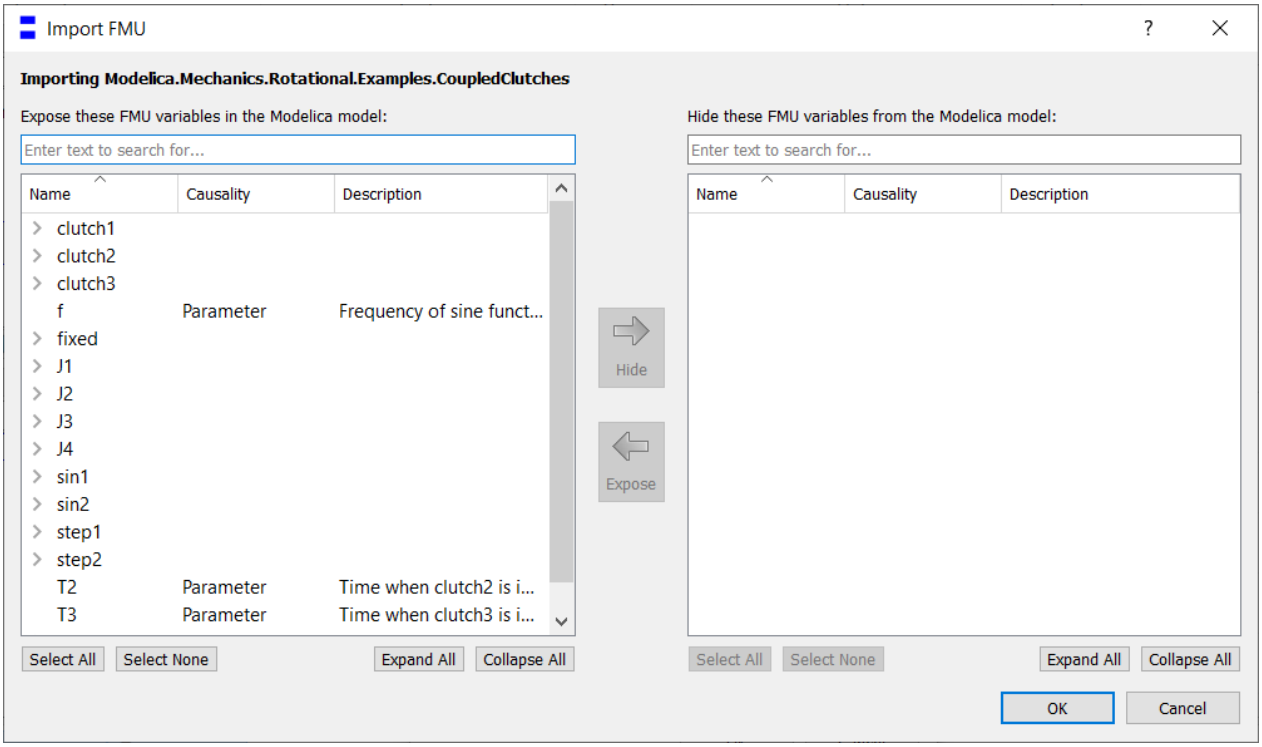
You can filter signals when importing an FMU. An example of a dialog, with default settings, to import an FMU created from the demo Coupled Clutches:



There are three alternatives:

- **All variables**: This corresponds to the previous option **Include all variables** being activated. This setting corresponds to the parameter `includeAllVariables=true` in the built-in function `importFMU`. This is the default setting.
- **Black box (parameters, inputs, outputs)**: This corresponds to the previous option **Include all variables** being deactivated. This setting corresponds to the parameter `includeAllVariables=false` in the built-in function `importFMU`.
- **These variables**: This is a new option in Dymola 2021x, you can select what variables to expose or hide from the FMU in detail. Selecting this alternative, you can then click **Select…** to have a dialog for what variables to select, and you will have the resulting variables in the variable field.

If you select **These variables:** and then click **Select…**, an example of the dialog that appears may be (in this case the demo Coupled Clutches is imported as an FMU):



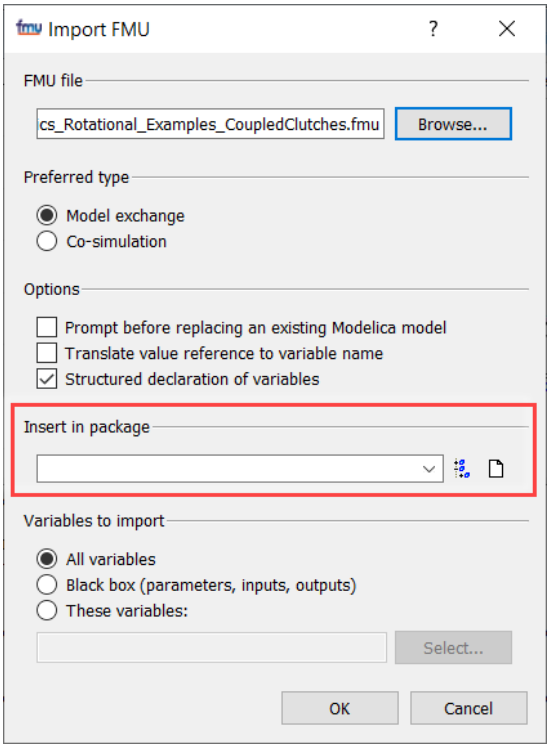
You select the variables you want to move to the other pane, and then click the arrow **Hide** or **Expose** depending on in which pane you want to have the variables, if they should be exposed or hidden.

For each pane, you have a search field on top, and four buttons at the bottom: **Select All**, **Select None**, **Expand All**, and **Collapse All**. Note that it might be handy to, in the above figure, to click **Select All** and then click the **Hide** arrow, if you want to select which variables to expose rather than the ones to hide.

The new filtering feature can also be used when scripting, see section “Improved built-in function importFMU” on page 31.

Selecting the package where the imported FMU should be inserted

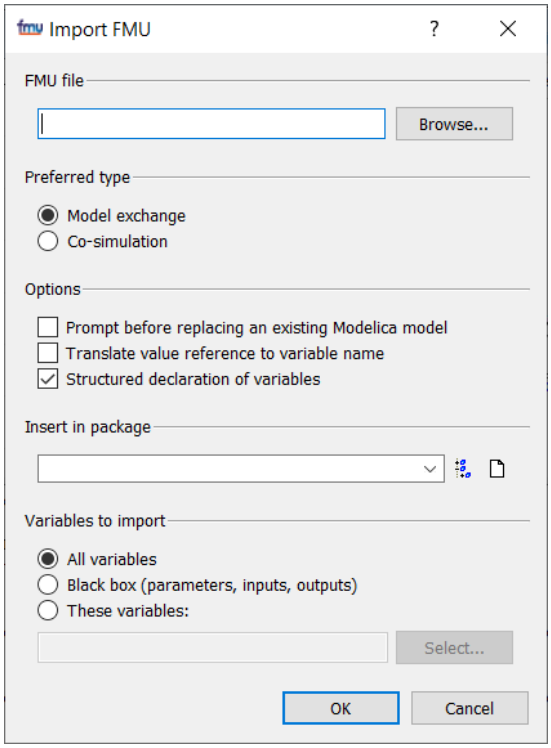
In Dymola 2021x you have a new group **Insert in package** to specify where the imported FMU should be inserted:

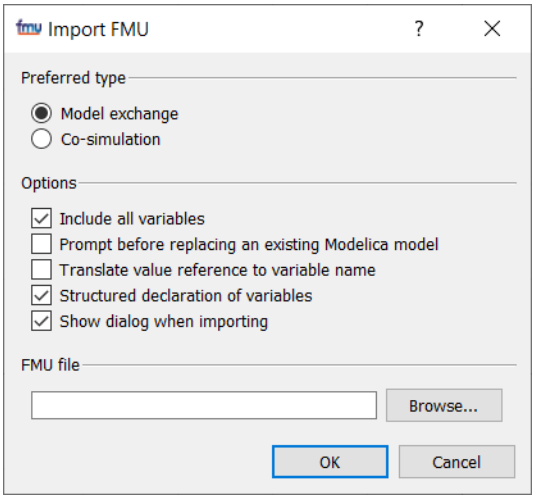


You can use the arrow to select from current packages; you can also browse the trees in the package browser, and finally create a new package.

Reorganization of the FMU import menu

As can be seen from previous sections, new features are available when importing FMUs. Due to this, the menu for importing an FMU has been reorganized. Below the menu in Dymola 2021x (to the left), and in previous version (to the right):





Except for the two new groups **Insert in package** and **Variables to import**, described in previous sections, the group **FMU file** is now put as first group, and two changes are seen in the group **Options**:

- The option **Include all variables** is now present as the **All variables** setting in the new **Variables to import** group. (The option **Include all variables** is however still present in the simulation setup, in the **FMI Import** tab, reached by the command **Simulation > Setup**.)
- The option **Show dialog when importing** has been removed from this menu, but is still available in the simulation setup, the **FMI Import** tab, reached by the command **Simulation > Setup**.

Enhanced functionality when importing Co-simulation FMUs supporting fmi2GetDirectionalDerivative

If you import a Co-simulation FMU and it supports the function `fmi2GetDirectionalDerivative`, Dymola will generate parameter arrays listing the input, output, exposed state, and exposed state derivative value reference and names.

In addition there is a function for the FMU, `linearizeFMU` that will take the valueReference as an argument and return the A,B,C,D matrices for the state space representation of the linearized system by calling `fmi2GetDirectionalDerivative` on the FMU.

If the flag `Advanced.FMI.LinearizeCSFMU` is set to `true` when you import the FMU, you will get a parameter `fmi_linearizeTime` that will allow you to select a time when the linearize function will be called. The function assigns the A,B,C,D matrices to the variables `fmi_AMatrix`, `fmi_BMatrix`, `fmi_CMatrix`, and `fmi_DMatrix`, respectively, at that given time.

3.7 Modelica Standard Library and Modelica Language Specification

The current version of the Modelica Standard Library is version 4.0.0. The current version of the Modelica Language Specification is 3.4.

Note that the Modelica Standard Library version 4.0.0 is compliant with the Modelica Language Specification 3.4.

3.8 New libraries

Below is a short description of new libraries. For a full description, please refer to the libraries documentation.

The libraries are presented in alphabetical order. If not stated as free, the library is commercial.

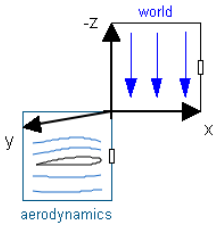
3.8.1 Aviation Systems Library

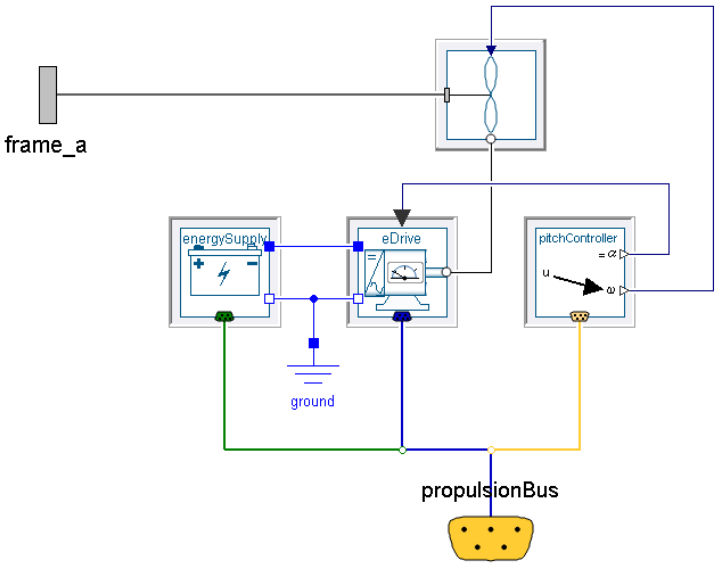
The new Aviation Systems Library (ASL) offers models for the aerodynamic behavior of aircrafts like airplanes or multi-copters. It uses standard MSL physical connectors to allow the design of propulsion systems with other libraries like the Electrical Powertrains Library (EPTL) considering the effects of the flight mission on the performance. Being fully integrated into the MSL MultiBody environment, all potential of Modelica libraries can be utilized.

In this first release, two main topics are covered:

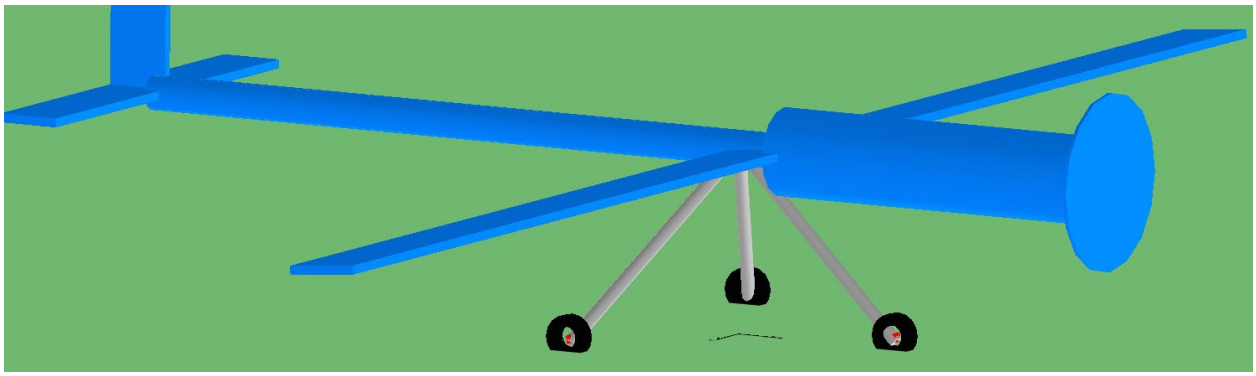
- Longitudinal dynamics of airplanes (fixed-wing) for designing a propulsion system considering the flight mission
- Full dynamics of multi-copter drones (rotating-wing) including the control system for mission planning and design of the propulsion system

The two areas share common models like propulsion or landing gear systems. Components of the electric drive are structured in a reasonable way to allow replacing by custom drive systems from other libraries:





The aerodynamic forces of wings or propeller blades are calculated by table-based profile polar curves. Geometrical arrangement of the aerodynamic objects to model an aircraft is conducted in the Modelica MultiBody environment. A simple tire model allows take-off from ground and landing:



3.9 Documentation

One single Dymola Full User Manual

To facilitate search and indexing of all Dymola User Manuals, a new manual, "*Dymola Full User Manual*" is present in Dymola 2021x. The manual contains all chapters in the Dymola User Manuals 1A – 2C. The order of the chapters are the same, except that the chapter "Appendix – Installation" has been moved to be the last chapter of this manual.

Manual structure

In parallel with the new Dymola Full User Manual, the content is as well present divided into a number of different manuals:

In the previous versions of Dymola, the former Dymola User Manual Volume 1 was split in two manuals:

- "*Dymola User Manual 1A: Introduction, Getting Started, and Installation*", containing the following chapters:
    - What is Dymola?
    - Getting started with Dymola
    - Introduction to Modelica
    - Appendix - Installation
- "*Dymola User Manual 1B: Developing and Simulating a Model*", containing the following chapters:
    - Developing a model
    - Simulating a model

The former Dymola User Manual Volume 2 was split into three manuals:

- "*Dymola User Manual 2A: Model Development Tools*", containing the following chapters:
    - Model Experimentation
    - Model Calibration
    - Design Optimization
    - Model Management
- "*Dymola User Manual 2B: Simulation Interfaces and Export*", containing the following chapters:
    - FMI Support in Dymola
    - Simulation Environments
    - Scripting and Reporting
- "*Dymola User Manual 2C: Advanced Concepts*", containing the following chapters:
    - Advanced Modelica Support
    - Visualize 3D
    - User-defined GUI

- Appendix - Migration

In the software distribution of Dymola 2021x Dymola User Manuals of version “September 2020” will be present; these manuals include all relevant features/improvements of Dymola 2021x presented in the Release Notes.

3.10 Appendix – Installation: Hardware and Software Requirements

Below the current hardware and software requirements for Dymola 2021x are listed.

3.10.1 Hardware requirements/recommendations

Hardware requirements

- At least 2 GB RAM
- At least 400 MB disc space

Hardware recommendations

At present, it is recommended to have a system with an Intel Core 2 Duo processor or better, with at least 2 MB of L2 cache. Memory speed and cache size are key parameters to achieve maximum simulation performance.

A dual processor will be enough if not using multi-core support; the simulation itself, by default, uses only one execution thread so there is no need for a “quad” processor. If using multi-core support, you might want to use more processors/cores.

Memory size may be significant for translating big models and plotting large result files, but the simulation itself does not require so much memory. Recommended memory size is 6 GB of RAM.

3.10.2 Software requirements

Microsoft Windows

Dymola versions on Windows and Windows operating systems versions

Dymola 2021x is supported, as 64-bit application, on Windows 8.1, and Windows 10. Since Dymola does not use any features supported only by specific editions of Windows (“Home”, “Professional”, “Enterprise” etc.), all such editions are supported if the main version is supported.

Compilers

Please note that for the Windows platform, a Microsoft C/C++ compiler, an Intel compiler, or a GCC compiler, must be installed separately. The following compilers are supported for Dymola 2021x on Windows:

Microsoft C/C++ compilers, free editions:

Note. When installing any Visual Studio, make sure that the option "C++/CLI support…" is also selected to be installed.

- Visual Studio 2012 Express Edition (11.0)
- Visual Studio 2013 Express Edition for Windows Desktop (12.0)
- Visual Studio 2015 Express Edition for Windows Desktop (14.0)
- Visual Studio 2017 Desktop Express (15) **Note!** This compiler only supports compiling to Windows 32-bit executables.
- Visual Studio 2017 Community 2017 (15)
- Visual Studio 2017 Build Tools **Notes:**
    - The recommended selection to run Dymola is the workload "Visual C++ build tools" + the option "C++/CLI Support…"
    - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
    - This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2017 alternative: **Visual Studio 2017/Visual C++ 2017 Express Edition (15)**.
    - For more information about installing and testing this compiler with Dymola, see www.Dymola.com/compiler.
- Visual Studio 2019 Community (16)
- Visual Studio 2019 Build Tools **Notes:**
    - The recommended selection to run Dymola is the workload "C++ build tools" + the option "C++/CLI Support…"
    - Installing this selection, no IDE (Integrated Development Environment) is installed, only command line features
    - This installation is not visible as a specific selection when later selecting the compiler in Dymola, the alternative to select is the same as for any Visual Studio 2019 alternative: **Visual Studio 2019/Visual C++ 2019 (16)**.
    - For more information about installing and testing this compiler with Dymola, see www.Dymola.com/compiler.

Microsoft C/C++ compilers, professional editions:

Note. When installing any Visual Studio, make sure that the option "C++/CLI support…" is also selected to be installed

- Visual Studio 2012 (11.0)

- Visual Studio 2013 (12.0)
- Visual Studio 2015 (14.0)
- Visual Studio Professional 2017 (15)
- Visual Studio Enterprise 2017 (15)
- Visual Studio Professional 2019 (16)
- Visual Studio Enterprise 2019 (16)

Intel compilers

The Intel compilers Intel Parallel Studio XE 2016, XE 2017, and XE 2018 are currently supported.

Note!

Important. The support for Intel compilers will be discontinued in a future release.

Note that you must also select a Visual Studio compiler when selecting any Intel compiler.

Currently supported combinations of Intel compilers and Visual Studio compilers:

- Intel Parallel Studio XE 2016: Visual Studio 2012, 2013, or 2015
- Intel Parallel Studio XE 2017: Visual Studio 2012, 2013, 2015, or 2017
- Intel Parallel Studio XE 2018: Visual Studio 2013, 2015, or 2017

Current limitations:

- Embedded server (DDE) is not supported.
- Export DLL is not supported.

GCC compilers

Dymola 2021x has limited support for the MinGW GCC compiler. The following versions have been tested and are supported:

- For 32-bit GCC: version 5.3, 6.3, and 8.2
- For 64-bit GCC: version 5.3, 7.3, and 8.1

Hence, at least the versions in that range should work fine.

To download any of these free compilers, please visit http://www.Dymola.com/compiler where the latest links to downloading the compilers are available. Needed add-ons during installation etc. are also specified here. Note that you need administrator rights to install the compiler.

Also, note that to be able to use other solvers than Lsodar, Dassl, and Euler, you must also add support for C++ when installing the GCC compiler. Usually, you can select this as an add-on when installing GCC.

Current limitations with 32-bit and 64-bit GCC:

- Embedded server (DDE) is not supported.
- Support for external library resources is implemented, but requires that the resources support GCC, which is not always the case.

- FMUs must be exported with the code export option[1] enabled. **Note!** When migrating to Modelica Standard Library (MSL) version 4.0, MinGW gcc versions older than 5 are not guaranteed to work for source code export.
- For 32-bit simulation, parallelization (multi-core) is currently not supported for any of the following algorithms: RadauIIa, Esdirk23a, Esdirk34a, Esdirk45a, and Sdirk34hw.
- Compilation may run out of memory also for models that compile with Visual Studio. The situation is better for 64-bit GCC than for 32-bit GCC.

In general, 64-bit compilation is recommended for MinGW GCC. In addition to the limitations above, it tends to be more numerically robust.

Dymola license server

For a Dymola license server on Windows, all files needed to set up and run a Dymola license server on Windows using FLEXnet, except the license file, are available in the Dymola distribution. (This includes also the license daemon, where Dymola presently supports FLEXnet Publisher version 11.14. This version is part of the Dymola distribution.)

As an alternative to FLEXnet, Dassault Systèmes License Server (DSLS) can be used. For more information, see “Dassault Systèmes License Server (DSLS) for Dymola” on page 42.

Linux

Supported Linux versions and compilers

Dymola 2021x runs on openSUSE 42.2, 64-bit, with gcc version 5.3.1, and compatible systems. (For more information about supported platforms, do the following:

- Go to https://doc.qt.io/
- Select the relevant version of Qt, for Dymola 2021x it is Qt 5.15
- Select Supported platforms)

Any later version of gcc is typically compatible. In addition to gcc, the model C code generated by Dymola can also be compiled by clang.

You can use a dialog to select compiler, set linker flags, and test the compiler by the **Verify Compiler** button, like in Windows. This is done by the command **Simulation > Setup**, in the **Compiler** tab.

You can however still change the compiler by changing the variable `CC` in `/opt/dymola-<version>-x86-64/insert/dsbuild.sh`. As an example, for a 64-bit Dymola 2021x application:

```
/opt/dymola-2021x-x86_64/insert/dsbuild.sh
```

Dymola 2021x is supported as a 64-bit application on Linux.

Notes

[1] Having the code export options means having any of the license features **Dymola Binary Model Export** or the **Dymola Source Code Generation**.

- 32-bit compilation for simulation might require explicit installation of 32-bit libc. E.g. on Ubuntu: `sudo apt-get install g++-multilib libc6-dev-i386`
- Dymola is built with Qt 5.15.0 and thereby inherits the system requirements from Qt. This means:
    - Since Qt 5.15 no longer supports embedding of the XCB libraries, these must now be present on the platform running Dymola. See the table in https://doc.qt.io/qt-5.15/linux-requirements.html for the list of versions of the ones starting with "libxcb". Note that the development packages ("-dev") mentioned outside the table are not needed.
    - The library `libxcb-xinput.so.0` and `libevent-2.0.so.5` might require explicit installation.
- For FMU export/import to work, zip/unzip must be installed.

Note on libraries

- The library UserInteraction is not supported on Linux.

Dymola license server

For a Dymola license server on Linux, all files needed to set up and run a Dymola license server on Linux, except the license file, are available in the Dymola distribution. (This also includes the license daemon, where Dymola presently supports FLEXnet Publisher 11.14.)

As an alternative to FLEXnet, Dassault Systèmes License Server (DSLS) can be used. For more information, see "Dassault Systèmes License Server (DSLS) for Dymola" on page 42.