

Dymola

Dynamic Modeling Laboratory

Dymola 7 Release notes

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

Document version: 1

© Copyright 1992-2011 by Dassault Systèmes AB. All rights reserved.
Dymola® is a registered trademark of Dassault Systèmes AB.
Modelica® is a registered trademark of the Modelica Association.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Dassault Systèmes AB
Ideon Science Park
SE-223 70 Lund
Sweden

E-mail: Dymola.Support@3ds.com
URL: <http://www.Dymola.com>
Phone: +46 46 2862500
Fax: +46 46 2862501

Contents

1	Important notes on Dymola 7	7
2	About this booklet	7
3	Dymola 7.4 FD01	8
3.1	Introduction	8
3.2	Developing a model	8
3.2.1	New version of Modelica Standard Library used	8
3.2.2	Modelica Text editor	8
3.2.3	Checking packages	9
3.2.4	Documentation.....	9
3.3	Simulating a model	10
3.3.1	Improved handling of initialization and start values.....	10
3.3.2	Improved diagnostics.....	10
3.4	Installation.....	11
3.4.1	Installation on Windows	11
3.5	Model Management	11
3.5.1	Licensing libraries	11
3.6	Other simulation environments	11
3.6.1	Dymola – Simulink interface.....	11
3.6.2	Real-time simulation.....	12
3.6.3	DDE Communication	12
3.6.4	FMI for model exchange	14
3.7	Advanced Modelica Support.....	15
3.7.1	Functions as formal input to functions supported.....	15
3.7.2	Operator overloading improved.....	15
3.7.3	New homotopy operator supported.....	15
3.7.4	Licensing	16
3.7.5	Additional minor improvements.....	16
3.8	Code and Model Export	16
3.8.1	Source Code Generation	16
3.9	Libraries	16
3.9.1	Modelica Standard Library	16
3.9.2	AirConditioning Library.....	17
3.9.3	Hydraulics Library.....	17
3.9.4	Vehicle Dynamics Library.....	18

3.9.5	Modelica_LinearSystems2	18
3.9.6	The ModelManagement package.....	19
3.10	Documentation	19
4	Dymola 7.4.....	20
4.1	Introduction.....	20
4.2	Developing a model	20
4.2.1	Search functionality improved.....	20
4.2.2	Improved display of redeclared classes	20
4.2.3	Update dialog extended	21
4.2.4	Modelica Text editor	21
4.2.5	Modelica Language support	22
4.2.6	Documentation.....	22
4.3	Simulating a model	26
4.3.1	Improved simulation performance.....	26
4.3.2	Default displayUnit in parameter dialog.....	26
4.3.3	Command window.....	26
4.3.4	Debugging models.....	29
4.4	Installation.....	30
4.4.1	Simplified installation of license file for all users	30
4.4.2	Remote installation of Dymola	30
4.5	Other simulation environments	31
4.5.1	Dymola – Simulink interface.....	31
4.5.2	Real-time simulation.....	31
4.5.3	FMI for Model Exchange	31
4.5.4	Importing Simulink models (using FMI).....	33
4.5.5	Interfacing Dymola and SIMULIA software.....	33
4.6	Migration.....	33
4.6.1	Conversion of location of class-level annotations	33
4.6.2	Conversion of URI's.....	33
4.7	Libraries	34
4.7.1	Modelica Standard Library	34
4.7.2	Hydraulics Library.....	34
4.7.3	Modelica_LinearSystems2	34
4.8	Documentation	35
5	Dymola 7.3.....	36
5.1	Introduction.....	36
5.2	Libraries	36
5.2.1	Modelica Standard Library	36
5.2.2	Hydraulics Library.....	36
5.2.3	Air Conditioning Library.....	37
5.2.4	Vehicle Dynamics Active Safety Library	37
5.2.5	Modelica_LinearSystems2	37
5.2.6	Modelica_StateGraph2	37
5.3	New information browser	38
5.4	Developing a model	38
5.4.1	New version of Modelica Standard Library used	38
5.4.2	Window handling	39

5.4.3	Modelica text editor.....	39
5.4.4	Documentation.....	42
5.5	Simulating a model	43
5.5.1	Window handling	43
5.5.2	Plot window.....	43
5.5.3	Animation window	44
5.5.4	Command window.....	45
5.5.5	Scripting	46
5.6	Linux.....	49
5.6.1	New Linux versions supported	49
5.7	Installation.....	49
5.7.1	New installation process on Linux.....	49
5.7.2	Dymola License Server	49
5.8	Other simulation environments	50
5.8.1	Dymola – Simulink interface.....	50
5.8.2	Real-time simulation.....	51
5.9	Advanced Modelica Support.....	51
6	Dymola 7.2.....	52
6.1	Introduction.....	52
6.2	Developing a model	52
6.2.1	Graphical user interface.....	52
6.2.2	Unary Connectors.....	54
6.2.3	Annotation to define an inverse of a function.....	55
6.3	Simulating a model	55
6.3.1	Graphical user interface.....	55
6.3.2	Parameter values.....	56
6.3.3	Surface-objects with transparency	56
6.3.4	Scripting	57
6.4	Installation.....	58
6.4.1	Installing a node-locked license.....	58
6.5	Other simulation environments	58
6.5.1	Dymola – Simulink interface.....	58
6.5.2	Real-time simulation.....	59
6.5.3	DDE communication	60
6.6	Libraries	60
6.6.1	Vehicle Dynamics Library.....	60
6.6.2	Modelica_Fluid Library.....	61
6.6.3	Automotive Demos Library.....	62
6.6.4	Hydraulics Library and Pneumatics Library.....	62
6.7	Advanced Modelica Support.....	62
6.7.1	User-defined derivatives – improvements	62
7	Dymola 7.1.....	63
7.1	Introduction.....	63
7.2	Developing a model	63
7.2.1	Graphical user interface.....	63
7.2.2	Expandable connectors	64
7.2.3	Modelica language support.....	64

7.3	Simulating a model	65
7.3.1	Improved simulation initialization.....	65
7.3.2	Scripting language	65
7.4	Installation.....	66
7.4.1	C compiler notes.....	66
7.4.2	Dymola License Server	66
7.5	Other simulation environments	66
7.5.1	Dymola – Simulink interface.....	66
7.5.2	Real-time simulation.....	66
7.6	Code and model export	67
7.6.1	Real-time simulation.....	67
7.6.2	Binary model export	67
7.6.3	Source code generation.....	67
7.6.4	Dymola run-time.....	68
7.7	Libraries	68
7.7.1	VehicleDynamics.....	68
7.8	Documentation	68
8	Dymola 7.0.....	69
8.1	Introduction	69
8.2	Developing a model	69
8.2.1	New version of Modelica Standard Library used	69
8.2.2	Graphical user interface.....	69
8.2.3	Unit checking and unit deduction	71
8.2.4	Modelica language support.....	72
8.3	Simulating a model	73
8.3.1	Plot window	73
8.3.2	Diagram layer in Simulate mode – improved usage	74
8.4	Installation.....	75
8.4.1	New installation process on Windows.....	75
8.5	Model Management	76
8.5.1	Encryption in Dymola	76
8.5.2	Model and library checking	76
8.6	Other Simulation Environments.....	76
8.6.1	Dymola – Matlab interface	76
8.6.2	Real-time simulation.....	77
8.7	Advanced Modelica Support.....	78
8.7.1	Symbolic solution of nonlinear equations in Dymola.....	78
8.8	Migration.....	79
8.9	Libraries	79
8.9.1	Modelica Standard Library	79
8.9.2	Modelica_LinearSystems	79
8.9.3	VehicleInterfaces	79
8.9.4	FlexibleBodies	79
8.10	Documentation	79

1 Important notes on Dymola 7

Installation on Windows

To translate models in Dymola 7.x you must also install a supported C compiler. The C compiler is not distributed in Dymola. Note in particular that GCC is no longer supported. Also note concerning **free** Microsoft compilers that earlier versions than Microsoft Visual Studio Express 2008 are not supported (concerning **full** versions, some earlier versions are supported). Please read corresponding sections in this booklet for more information.

Administrator privileges are required for installation.

The installation procedure on Windows has been changed compared to Dymola 6. Dymola 7.0 and later will create directories of their own; there is no need to remove previous version(s). Please see “Dymola User Manual 1”, chapter “Appendix – Installation” for more information.

2 About this booklet

This booklet covers Dymola 7.0 to Dymola 7.4 FD01. For versions earlier than Dymola 7.4 FD01 the release notes have been somewhat shortened.

The disposition for each Dymola version is similar to the one in Dymola User Manual Volume 1 and 2; the same main headings are being used (except for e.g. Libraries and Documentation).

3 Dymola 7.4 FD01

3.1 Introduction

A number of improvements and additions have been implemented in Dymola 7.4 FD01. In particular, Dymola 7.4 FD01

- Supports and includes Modelica Standard Library 3.2.
- Enhances the AirConditioning, Hydraulics and Vehicle Dynamics libraries.

3.2 Developing a model

3.2.1 New version of Modelica Standard Library used

By default Dymola will start with Modelica Standard Library 3.2. If the user wish to use Modelica Standard Library 2.2.2 this can be selected using the command **Edit > Options...** selecting the **Versions** tab. Concerning new features in Modelica Standard Library 3.2 please see section “Modelica Standard Library” on page 16 in this booklet.

Note! If you have previously been using Dymola 7.4 or earlier, please use the command above and change Modelica version to 3.2 after installation.

3.2.2 Modelica Text editor

Improved handling of collapsed blocks

Editing is not allowed if a collapsed block is selected, in order to prevent unwanted actions; e.g. deletion of unexpanded blocks.

Improved indenting

The automatic indenting now automatically adjusts indenting for e.g. end statements before entering Return.

Improved Find functionality

The GUI for the **Find** functionality is improved. Please see the corresponding section in the section “Documentation” below.

3.2.3 Checking packages

Checking of sizes

The reporting of size checking has been improved. When checking a package the following output is the best result:

```
The model has the same number of unknowns and equations.
```

However, any of the following two results are also acceptable:

```
The model has the same number of unknowns and equations for the  
given bindings of parameters.
```

```
The model has the same number of unknowns and equations for the  
given numerical settings of parameters.
```

When checking a model or a connector the message also includes the number or symbolic expression for the size.

By setting the flag

```
Advanced.LogSymbolicSizeCheck=true
```

you also get the numbers when checking a package. Moreover, there might be warnings that parameters were already evaluated before the check started.

3.2.4 Documentation

Link handling improved



The handling of links using the button **Create Link** in the documentation editor has been improved:

- Links with mixed text and images are supported.
- Links with embedded formatting (e.g. bold) are supported.

Improved Find functionality

The GUI for the **Find** functionality has been improved.

- The color marking of a selected text has been changed to blue to improve the visibility.
- The search menu is not closed when entering Return.
- Incremental search is applied; the search starts from the position of the cursor.

3.3 Simulating a model

3.3.1 Improved handling of initialization and start values

New homotopy methods

Dymola supports the new homotopy operator introduced in Modelica 3.2. Please see section “New homotopy operator supported” on page 15 for more information.

Improved manipulation to be invariant to renaming of components and variables

The symbolic manipulation done during translation includes parts such as selection of alias variables, selection of continuous time states and selection of iteration variables for algebraic loops. The outcome may depend on which order variables and equations were treated. Previously Dymola used alphabetic order for the variables. It had, unfortunately, the consequence that renaming a component or variable could give other iteration variables and Dymola could fail to solve the algebraic loop because the start values of the new iteration variables were not good enough.

In order to be invariant with respect to renaming of components and variables, Dymola now uses the order of declaration for the variables and equations when eliminating alias variables, selecting continuous time states and selecting iteration variables for algebraic loops. In order to get good start values for the numerical solver, Dymola tries, as previously, first to eliminate variable with less reliable start values, see the manual “Dymola User Manual Volume 1”, section 5.8.3 for more information.

3.3.2 Improved diagnostics

Nonlinear systems of equations

If Dymola fails to solve a nonlinear system during simulation, the error message will include

```
Nonlinear system of equations number = xxx
```

where xxx will be a number.

In order to facilitate the search of the location of the system, exactly the same line is included in the file containing the C code, dsmodel.c. If flat Modelica code is generated in a dsmodel.mof file, the line will be present also in that file.

3.4 Installation

3.4.1 Installation on Windows

Compilers

New compiler supported

The new compiler Microsoft Visual Studio C++ 2010 is now supported, both the Professional edition and the Express edition.

To download the free Express edition, the link

<http://www.microsoft.com/express/Downloads/#2010-visual-CPP>

can be used. Note that you need administrator rights to install the compiler.

Discontinued support

Official support for the compilers Visual Studio 6, Visual Studio .NET 2002 (7.0) and Visual Studio .NET 2003 (7.1) has been discontinued with Dymola 7.4 FD01.

3.5 Model Management

3.5.1 Licensing libraries

Dymola 7.4 FD01 supports the Modelica feature of licensing “external” Modelica libraries, i.e. libraries which are not sold and licensed through the Dassault Systèmes channels.

The basic feature of licensing according to the Modelica Language Specification is supported. Licensing is defined, according to the Modelica Language Specification, as “restrict the use of an encrypted package for particular users for a specified period of time”. (Licensing of “external” Modelica libraries was actually introduced in Dymola 7.4, but has been extended in Dymola 7.4 FD01.)

3.6 Other simulation environments

3.6.1 Dymola – Simulink interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R14SP3 (ver. 7.1) up to R2010b (ver. 7.11). Support for Matlab R14SP2 (ver. 7.0.4) has been discontinued with

Dymola 7.4 FD01. Only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. The LCC compiler is not supported.

3.6.2 Real-time simulation

Dymola 7.4 FD01 generated code has been verified for compatibility with the following combinations of dSPACE and Matlab releases on the DS1005 and DS1006 platforms

- dSPACE Release 5.0 (RTI 5.3.5) with Matlab R14SP3
- dSPACE Release 5.1 (RTI 5.3.6) with Matlab R2006a
- dSPACE Release 5.2 (RTI 5.3.7) with Matlab R2006b
- dSPACE Release 5.4 (RTI 5.5) with Matlab R2007a
- dSPACE Release 6.0 (RTI 5.6) with Matlab R2007b
- dSPACE Release 6.1 (RTI 6.0) with Matlab R2007b
- dSPACE Release 6.2 (RTI 6.1) with Matlab R2008a
- dSPACE Release 6.3 (RTI 6.2) with Matlab R2008b
- dSPACE Release 6.4 (RTI 6.3) with Matlab R2009a
- dSPACE Release 6.5 (RTI 6.4) with Matlab R2009b
- dSPACE Release 6.6 (RTI 6.5) with Matlab R2009b and R2010a

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, we 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.

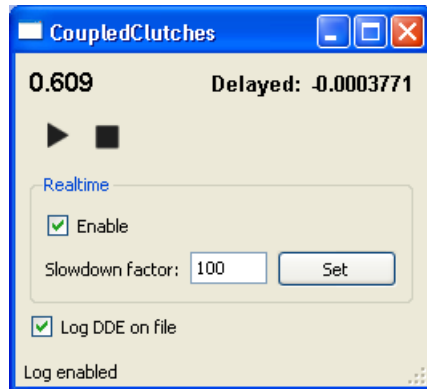
Compatibility – xPC Target

Compatibility with Matlab xPC Target has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, which means R14SP3 (xPC Target ver. 2.8) to R2010b (xPC Target ver. 4.4). Only Microsoft VisualC compilers have been tested.

3.6.3 DDE Communication

Improved Dymosim Windows application

The user interface of the feature of compiling Dymosim as a Windows application has been improved:



- The design is more intuitive.
- The name of the simulated model is shown.
- Short key for changing between run and pause: **Ctrl+P**
- A status line is displayed.

The example above shows the GUI for the model CoupledClutches, simulated with realtime (synchronization) using a slowdown factor of 100. The slowdown factor can be modified anytime. The simulation has been paused at 0.609 seconds by the user. The time the simulation is delayed at the accepted output point, relative real time is displayed. A negative value indicates that the simulation is faster than real time, i.e., that there is spare time for additional computations. (If so, the simulation is actually delayed by the system in order not to accumulate the time difference.)

The simulation can be resumed by clicking on the “play” button or using the short key **Ctrl+P**. (When the simulation is running, a “pause” button is shown instead of the “play” button.) Logging of DDE events to file has been activated. The status bar shows the latest DDE related activity.

There are some limitations of the feature:

- It is not possible to build Dymosim as a DLL when DDE is selected.
- Currently only the solvers Lsodar, Dassl, Euler, Rkfix2, Rkfix3 and Rkfix4 are supported.
- When clients request data from the DDE server they specify the desired data format. Dymosim currently only support the following formats:
 - Plain text, used by e.g. Matlab.
 - XITable, used by e.g. Microsoft Office Excel.

3.6.4 FMI for model exchange

FMI XML description extended to include declared type

Previously when generating the XML file according to the Model Description Schema, Dymola inlined the complete type definition individually for each variable, except for enumeration types.

Now Dymola outputs Type elements for appearing types in the element TypeDefinitions, e.g.

```
<TypeDefinitions>
  <Type
    name="Modelica.Blocks.Interfaces.RealInput">
    <RealType/>
  </Type>
  <Type
    name="Modelica.SIunits.Angle">
    <RealType
      quantity="Angle"
      unit="rad"
      displayUnit="deg"/>
    </RealType>
  </Type>
```

These types are referenced by the declaredType attribute of the type modifiers of the ScalarVariable elements, e.g.

```
<ScalarVariable
  name="motor.emf.phi"
  valueReference="905969676"
  description="Angle of shaft flange with respect to support
(= flange.phi - support.phi)">
  <Real
    declaredType="Modelica.SIunits.Angle"/>
</ScalarVariable>
```

It is possible retain the old style where Dymola inlined the type definition for each variable by setting the flag

```
Advanced.FMI.InlineTypeDefinitions = true;
```

For the example above, the TypeDefinitions will be empty and the variable will be declared as

```
<ScalarVariable
  name="motor.emf.phi"
  valueReference="905969676"
  description="Angle of shaft flange with respect to support (=
flange.phi - support.phi)">
  <Real
    quantity="Angle"
    unit="rad"
    displayUnit="deg"/>
</ScalarVariable>
```

FMU Export of Simulink Models

A new version (1.1) is provided for the Simulink FMU export package (rtwscfnfmi.zip in the m-files directory of the Dymola 7.4 FD01 distribution). This new version has been updated to support Matlab R2010a. There is also support to generate FMUs from Simulink models containing custom S-function blocks written in C.

3.7 Advanced Modelica Support

3.7.1 Functions as formal input to functions supported

Functional input arguments to functions in order to e.g. pass criteria functions to generic optimization functions are supported. The following functionality is currently supported:

- Sending a function as an input argument to another function, e.g. `function sin()` as argument.
- Calls through a functional input argument, e.g. `integrand(x)`.
- Propagating a functional input argument, either directly, e.g. `function integrand(x)`, or with partial binding, e.g. `function integrand(x=1)`.
- Partial interface functions.
- Functional input arguments to functions can be used interactive or in models.

3.7.2 Operator overloading improved

Dymola supports the improvements in operator overloading defined for Modelica 3.2.

- The new special class `operator record` is supported. Overloaded operators can only be defined inside such a class.
- Inheritance of `operator record` is allowed if defined via a short class definition.
- New overloaded element “0” is supported. This element enables operator record classes to be used as flow variables in connectors.

3.7.3 New homotopy operator supported

The new Modelica homotopy operator `homotopy` is supported in Dymola 7.4 FD01

The operator is a powerful feature in improving the convergence of iterative solvers by providing an alternative, simplified version of the model that is not so dependent of accurate initial guess values for the unknown variables, and then continuously transforming that simplified model to the original more complicated model. The homotopy is normally applied when the usual initialization fails; it is however also possible to directly enable the homotopy iterations by setting the flag

```
Advanced.OnlyUseHomotopyMethod = true;
```

this will activate homotopy iterations for initializing all models.

3.7.4 Licensing

Licensing of packages is supported. Please see section “Licensing libraries” on page 11 for more information.

3.7.5 Additional minor improvements

Better handling of “Include” and “Library”

Dymola supports the default location for include and library files inside the package storage location now implemented, and the corresponding annotation to override this location if needed.

The “modelica://” URI scheme can be used in this annotation.

Use of “modelica://” URI scheme in annotations for scripts

The “modelica://” URI scheme can be used in annotations for scripts. Such annotations are used e.g. when implementing run script commands or running conversion scripts.

Annotation groupImage supported

The standard Modelica annotation `annotation(Dialog(groupImage))` for showing an image in e.g. a parameter dialog is supported in Dymola 7.4 FD01.

3.8 Code and Model Export

3.8.1 Source Code Generation

Improved tracing information

The comments in the generated C code created by the flag `Advanced.OutputEquationTrace` now also contain aliases, substitutions and substituted constants for improved traceability.

3.9 Libraries

3.9.1 Modelica Standard Library

The Dymola 7.4 FD01 distribution contains two versions of Modelica Standard Library (MSL):

- version 3.2 released in October, 2010 (build 5).
- version 2.2.2 released in August, 2007 (build Rev 1128, June 2008).

MSL 3.2 is default in Dymola 7.4 FD01. If the user wish to use MSL 2.2.2 this can be selected using the command **Edit > Options...** selecting the **Versions** tab. When an old model using MSL 2.2.1 is opened, a conversion is made to the default MSL version.

MSL version 3.2

MSL 3.2 is the latest Modelica Standard Library and builds on the MSL 3.1. Version 3.2 is backward compatible to version 3.1, 3.0 and 3.0.1; models developed with any of those versions will work without any changes in version 3.2.

Some highlights of the new version:

- A number of new sub-libraries have been added for e.g. handling of input/output control blocks with Complex signals, very fast simulations of electrical circuits, detailed simulations of electrical circuits, magnetic fundamental waves in electric machines and computation of convective heat transfer and pressure loss characteristics.
- All physical models that dissipate heat have now an optional heat port to which the dissipated energy is flowing if activated. This will significantly improve design studies about the thermal efficiency of technical systems.
- Handling of electrical machines has been improved by inclusion of a possibility of modeling of losses and the possibility of summarizing converted power and losses.
- MSL 3.2 uses three new language elements compared to MSL 3.1 (operator records/overloaded operators, functions as input arguments to functions, and improved expandable connectors).

For further details, please see the release notes of the library.

MSL version 2.2.2

MSL 2.2.2 can be used for older models when conversion to Modelica 3.x is not wanted. It is backward compatible with version 2.2.1/2.2 with a few exceptions in Modelica.Media.

3.9.2 AirConditioning Library

Important additions:

- Support for engine cooling applications has been added, with improved performance for detailed heat exchanger models.
- The refrigerant R1234yf is now part of the standard distribution, with new, more accurate equation of state.
- The streams concept is now used in connectors.

3.9.3 Hydraulics Library

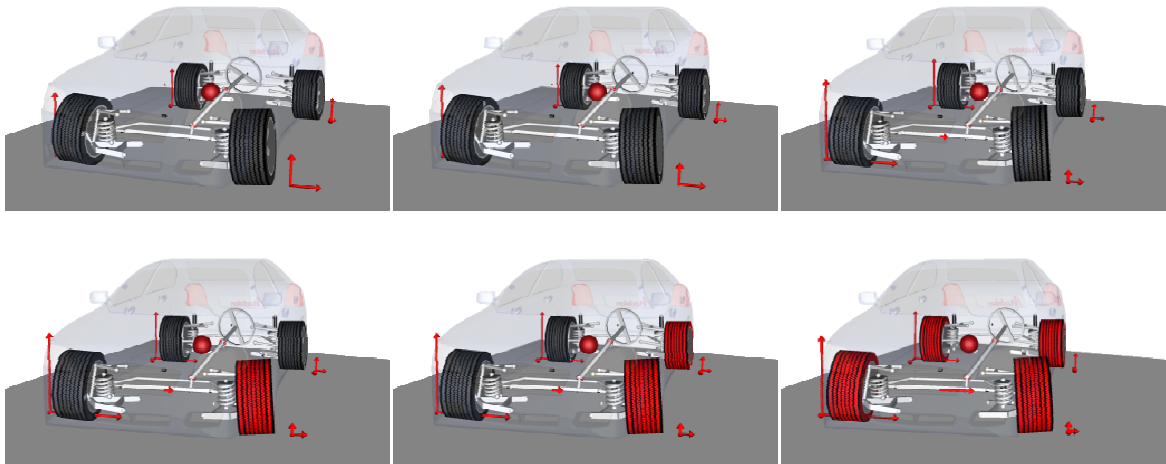
Important additions:

- New models for accumulators, using an accurate model for high pressure Nitrogen.
- Possibility to use appropriate fluids from Modelica.Media as hydraulic fluids.
- Several new directional control valves.
- Many customer requests for usability improvements are included.

3.9.4 Vehicle Dynamics Library

The library has been further developed; some highlights in version 1.5:

- Enhanced support for steady state and quasi steady state driving situations, e.g. cornering, corner entry and exit, and braking. The images below illustrate the quasi-steady-state solutions at corner exit for different percent of maximum performance.



- Improved table based suspensions that capture coupling effects between bump and steer, as well as steering system forces.
- Improved configurability of suspensions, with extended support to conveniently change mount points, as well as adding shims and other tuning elements.

3.9.5 Modelica_LinearSystems2

The library has been further developed; some improvements:

- Two new records `DiscreteZerosAndPoles` and `DiscreteTransferFunction` have been added to operate with discrete zeros-and-poles-transfer functions and transfer functions.
- Constructors and basic operators for the records `DiscreteStateSpace`, `DiscreteZerosAndPoles` and `DiscreteTransferFunction` have been added.
- Functions to construct, convert, analyze and plot are provided for the records above.
- Examples for the above records are provided.

3.9.6 The ModelManagement package

The ModelManagement package version 1.1 has been updated with new features for structural validation (translation statistics). The structural validation now also compares the

- Selected states
- Nonlinear iteration variables of the initialization problem
- Nonlinear iteration variables of the simulation problem

In order to use the new features new reference files must be generated.

The option **Variable unit checking** has been removed from the parameter dialog of the function ModelManagement.Check.checkLibrary. The option is obsolete since it has been part of the regular check of models in Dymola since several releases. Note that this is a non-backward compatible change for users calling the function from e.g. scripts.

3.10 Documentation

In the software distribution of Dymola 7.4 FD01 Dymola User Manuals of version “March 2011” will be present; these manuals include all relevant features/improvements of Dymola 7.4 FD01 presented in the Release notes.

4 Dymola 7.4

4.1 Introduction

A number of improvements and additions have been implemented in Dymola 7.4. In particular, Dymola 7.4 has a focus on the simulation capabilities whereas Dymola 7.3 had a focus on graphical user interface. Sparse techniques for the Jacobians have been introduced in the solvers. This gives considerably faster simulations for models with many state variables and complex model equations, such as discrete 1D fluid systems. For example, many models of the AirConditioning Library run 2-6 times faster.

Another important new feature is the support for Functional Mock-up Interface (FMI) which enables execution of model equations in other software environments. Other features such as improved tools for investigation of models, improved documentation tools and simplified installation will increase the usability of Dymola.

4.2 Developing a model

4.2.1 Search functionality improved

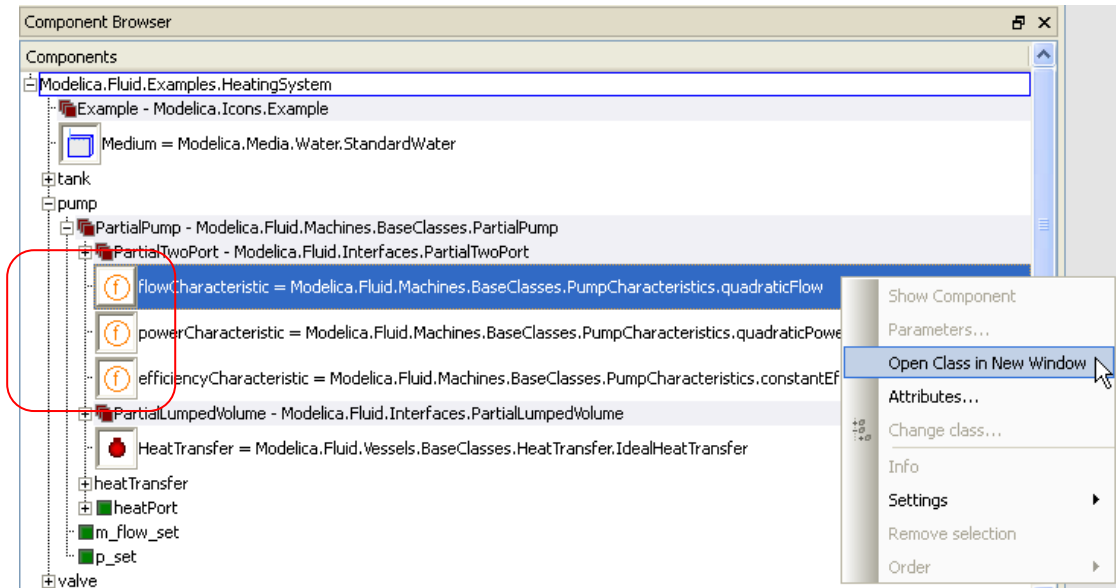
The command **File > Search...** has been improved for Modelica Text. It can now also be used for non-hidden information in encrypted models. Annotations (documentation etc) are also searched.

It is now also possible to use the search functionality of Microsoft Windows to search in the content of Modelica files (.mo).

4.2.2 Improved display of redeclared classes

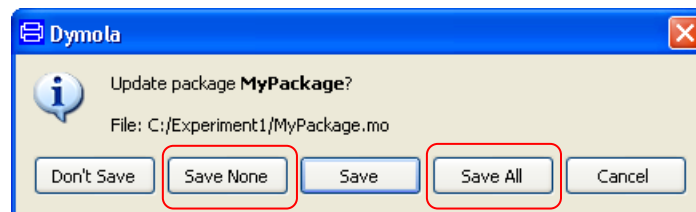
In the component browser, class parameters (i.e. replaceable classes in sub-components) now show the actual redeclared class in the same way as in the Diagram layer. Replaceable components with the default value are shown sunken, as a socket. Re-declared components are shown raised, as mounted in the socket.

Redeclared classes can now also be opened from the component browser using the context command **Open Class in New Window**. These features make it easier to understand complex models involving class parameters.



4.2.3 Update dialog extended

The dialog that appears for changed packages when saving or closing Dymola has been improved to contain more options.



The new options are:

- **Save None** corresponds to **Don't Save** for all packages that have been changed.
- **Save All** corresponds to **Save** for all packages that have been changed.

In particular the **Save All** option is very handy in this version because of e.g. the automatic conversion that moves class-level annotations to the right place according to the specification; this will change a large number of files of a library.

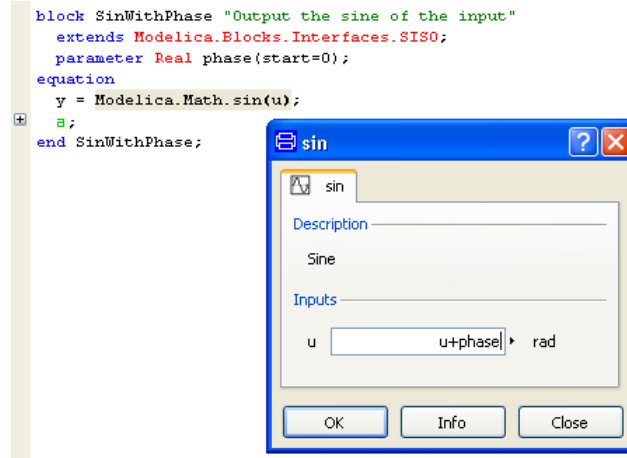
4.2.4 Modelica Text editor

Comments in equation blocks displayed using mathematical notation

Comments among equations are now displayed when using Mathematical notation in the Modelica Text layer.

More intuitive handling of context menu

When popping the context menu by right-clicking, the text cursor automatically will be placed at the click before popping of context menu, which will lead to popping of relevant context menu. The example below illustrates right-clicking on the sinus function call, popping the context menu and selecting **Edit Function Call** to edit that call.



4.2.5 Modelica Language support

Some features/improvements in Dymola:

- Dymola supports automatic rendering of the new Modelica 3.1 annotations *versionDate*, *versionBuild* and *dateModified* in the documentation layer. The annotations are rendered as:
“Version: *version*, *versionDate*, build *versionBuild* (*dateModified*)”
Note that the *dateModified* is only displayed if *versionBuild* has a value.

4.2.6 Documentation

Modelica URI scheme supported for images and links

General

Dymola now supports the Modelica URI (Uniform Resource Identifier) ‘modelica://’ scheme when handling images and links (to e.g. pdf documents). New references created will use this scheme automatically.

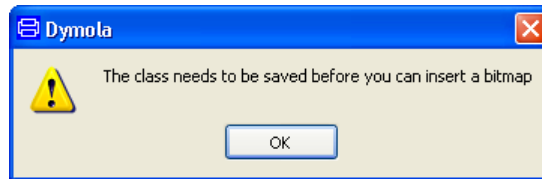
Inserting images



Inserting an image in the diagram or icon layer using the button **Bitmap** or inserting an image in the documentation layer using the button **Insert image** will automatically use the modelica:// URI scheme.

The starting point for interpretation of the `modelica://` URI will be the folder where the corresponding Modelica package resides.

In order to be able to generate correct links, the `model/package` has to be saved (using e.g. **File > Save**) before an image can be inserted. If initial saving has not been done, a warning will be displayed:



We recommend storing images that should be used in Dymola documentation in a folder "Images" located in the same folder as the relevant top package. Using this recommendation for an image "Illustration.png" for a top package "MyPackage", the URI

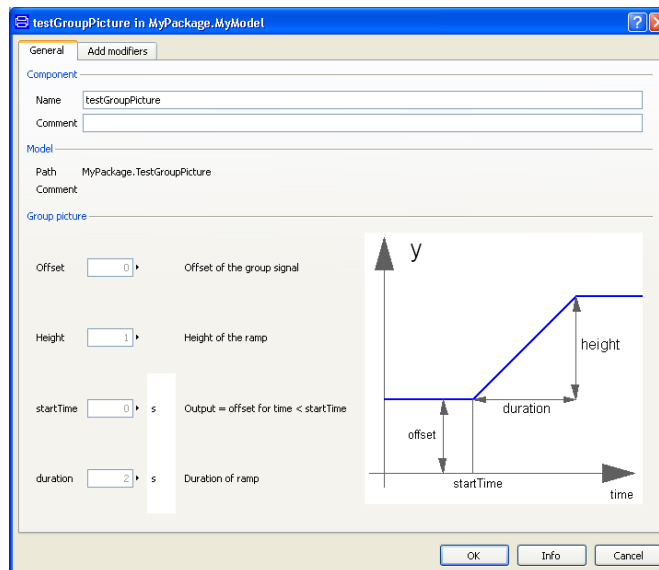
```
modelica://MyPackage/Images/Illustration.png
```

will be the resulting URI when using the **Bitmap** or **Insert image** button. This URI can be used in all sub-packages of `MyPackage` to refer to the image. It will also be intuitive to move the folder "Images" together with the corresponding packages if needed, which will preserve the image references.

Also the annotation `__Dymola_Images` for inserting images in e.g. parameter dialogs supports this scheme. An example taken from the Dymola User Manual Volume 2, chapter "Modelica Data Structures and GUI":

```
annotation (__Dymola_Images(Parameters(group="Group picture",
    source="modelica://MyPackage/Images/ramp.png")));
```

Example of image in parameter dialog.



Conversion of older code

For conversion of older code, the conversion function

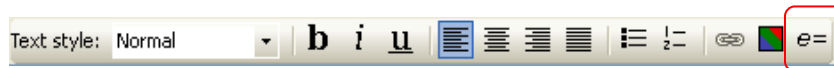
```
updateModelicaURIreference ("packagename")
```

can be called using the command input line in the command window.

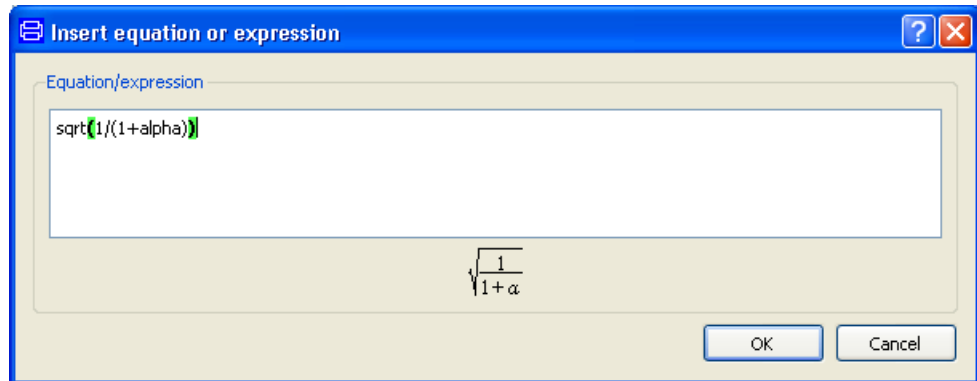
Improved handling of expressions and equations in the documentation layer

Creation of an equation or expression

The toolbar of the documentation editor has been extended with a button **Insert equation or expression**.



When clicking the button an editor is opened for the creation of an equation/expression according to Modelica syntax.



The result of the editing is shown below the editing pane. If an equation/expression is not concluded, the text *Incomplete equation/expression* is displayed. Clicking **OK** the equation/expression is inserted in the documentation editor where the cursor was located when pressing the **Insert equation or expression** button in the first place.

Equations or expressions created this way are displayed in italics. They are in fact images; nothing inside can be selected, copying a section containing such items and pasting it into e.g. Microsoft Word will display these items as pictures. The equations/expressions can however still be edited in Dymola, please see below.

In order to be able to generate correct links, the model/package has to be saved (using e.g. **File > Save**) before an image can be inserted. If initial saving has not been done, a warning will be displayed.

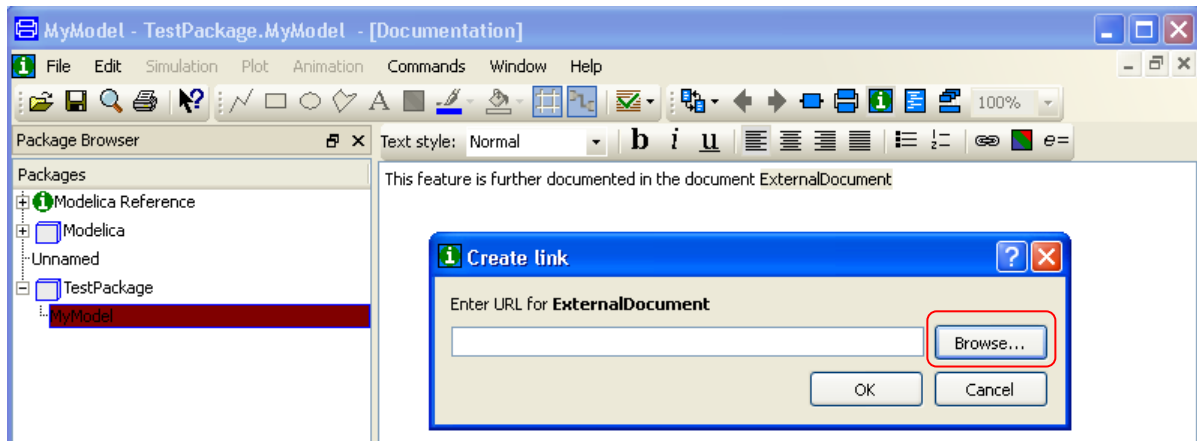
The images created will be stored in a folder `Images\equations` located in the same folder as the corresponding Modelica package. The folder will be automatically created if not present.

Editing of an existing Modelica equation or expression

By selecting an equation or expression that has been created using the **Insert equation or expression** button in the documentation editor, the cursor will place itself after or before the equation/expression. Popping the context menu by right-clicking and selecting **Edit equation** the editor described in previous section is displayed, and editing of the equation/expression can be made. Clicking **OK** the edited equation/expression will replace the previously selected one.

New option of creating links to documents in the documentation layer

The dialog of the command button **Create link** now has a **Browse** button which makes it possible to create local links to e.g. `.pdf` and `.html` files. This feature enables creation of documentation that refers to external documentation.

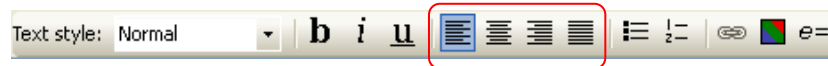


The link browsed will be shown in the dialog. The link created will use the `modelica://` scheme described previously if possible; absolute path will be used if the document referred is located on e.g. another drive. When referring to external documentation, it is recommended to store such documentation in a folder created in the folder where the top package resides. Doing so facilitates moving the package without losing the links; the corresponding folder should be moved together with the package.

In order to be able to generate local links, the model/package has to be saved (using e.g. **File > Save**) before a local link can be inserted. If initial saving has not been done, a warning will be displayed.

New possibilities of aligning text in the documentation layer

The toolbar of the documentation editor has been extended to facilitate selection of alignment of selected text.



The buttons for text alignment are in order **Left align**, **Center align**, **Right align** and **Justify** (the default is **Left align**).

4.3 Simulating a model

4.3.1 Improved simulation performance

By utilizing sparse Jacobian handling, the simulation is in many cases faster. Models that in particular will benefit by this improvement are e.g. large models including heavy media calculations. For example, many models of the AirConditioning Library run 2-6 times faster in Dymola 7.4.

Note that *no* additional setting like e.g. analytical Jacobian is needed in order to utilize this feature.

The sparse Jacobian handling is currently only available for DASSL and LSODAR integration algorithms.

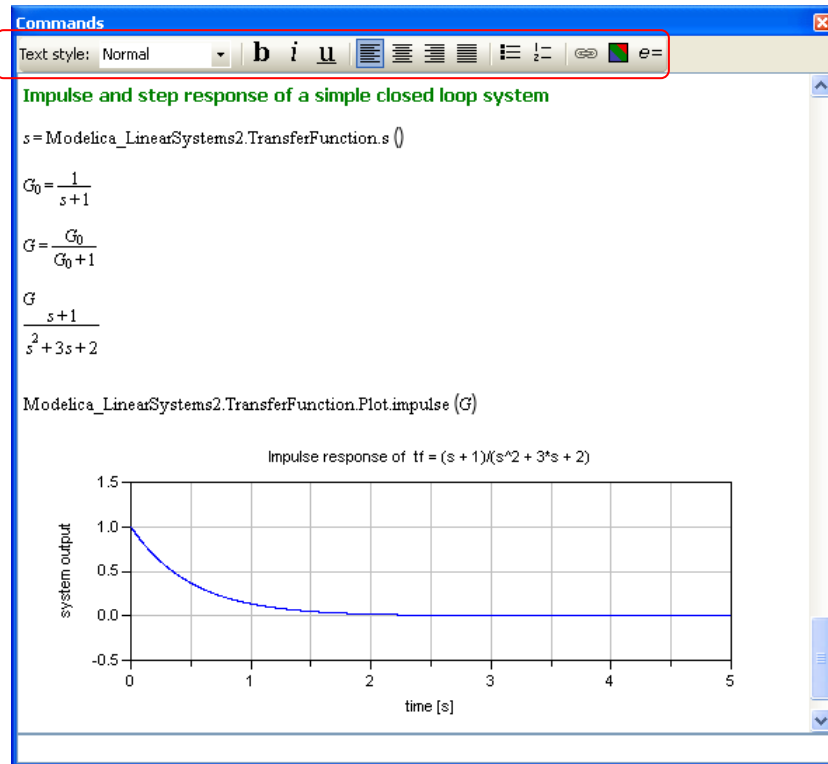
4.3.2 Default displayUnit in parameter dialog

Now the default display units, as defined in the script `displayunit.mos`, are displayed in the parameter dialog when no display unit is defined in the variable.

4.3.3 Command window

Documentation editor activated by default

The documentation editor available in Dymola 7.3 by setting a flag is now by default activated in the command window.



The old scripting window can still be used, by setting the flag:

```
Advanced.ScriptingWindow.UseNewFeatures=false
```

Improved handling of equations and expressions in the command log

The documentation editor available in the command window has been extended – please see section “Improved handling of expressions and equations in the documentation layer” on page 24 for a description of the features also available in the command window. For the handling of created image objects, please see below.

Improved storage and handling of image objects in a saved command log

If a command log contains any image object, a folder *DymolaLogName_files* is created when saving the log in .html format by the command **File > Save Log....** The image objects will be stored in this folder depending on the type:

- Images inserted using the button **Insert image** will be *copied* to the folder *DymolaLogName_files*.
- Image objects created using the button **Insert equation or expression** or created when editing such equation/expressions will be copied from a temporary folder to a sub-folder generated in the *DymolaLogName_files* folder.

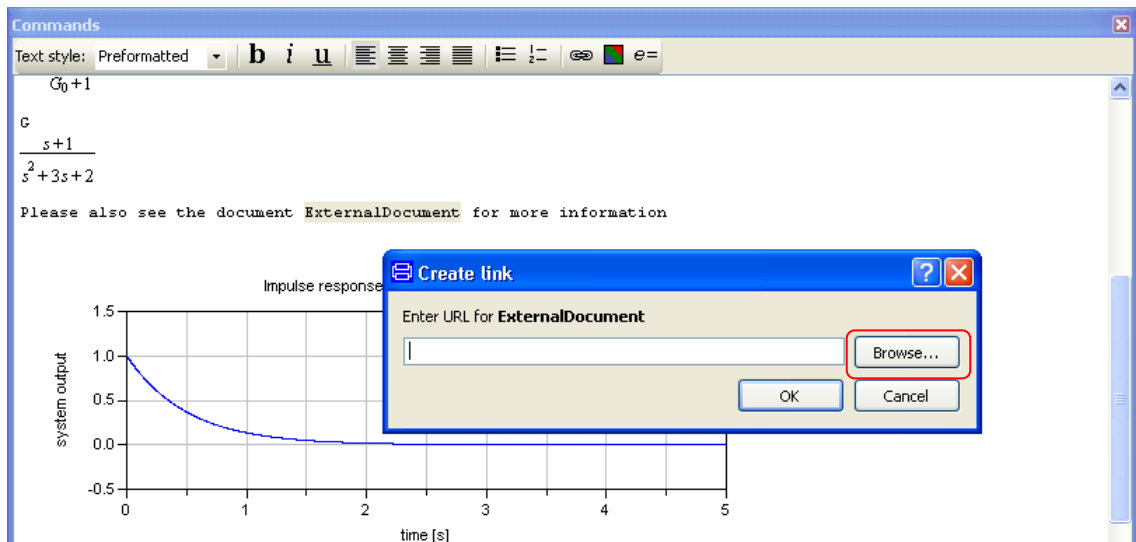
- Image objects created by Dymola as “math” rendering output to commands will be created in the sub-folder generated as well when saving the log.

The generated links in the html log will be relative to the folder *DymolaLogName_files*. The result will be that it will be easy to move the complete log by moving the html file and the corresponding folder, without losing any links.

Please compare with the handling of local links below, in particular when it comes to storage of referred documents.

New option of creating links to documents in the documentation layer

The dialog of the command button **Create link** now has a **Browse** button which makes it possible to create local links to e.g. .pdf and .html files. This feature enables creation of documentation that refers to external documentation.



The link browsed will be shown in the dialog. The links created will be relative to the location of the working directory of Dymola. Please note the consequences of this:

- The command log has to be saved in the working directory of Dymola, at least initially. This will be the default directory when the log is saved by the command **File > Save Log...**
- If the command log should be moved, documents referred to should be stored in such a way that it is easy and intuitive to move them together with the log. We recommend storing the document referred to in a folder named e.g. *LinkedDocuments* located in the working directory of Dymola *before* referring to them by the **Create link** button. By moving this folder together with the log, the links are preserved.

Please compare this with the corresponding handling of images in the command log using e.g. the **Insert image** button, see section “Improved storage and handling of image objects in a saved command log” on page 27.

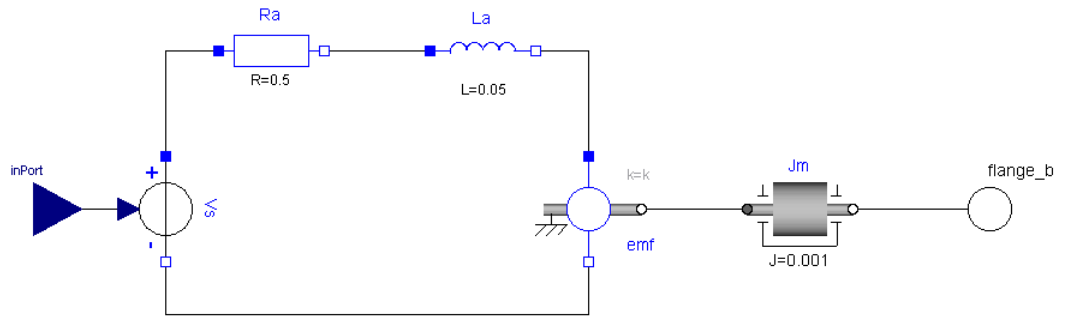
New possibilities of aligning text in the command log

The documentation editor in the command window has been extended – please see section “New possibilities of aligning text in the documentation layer” on page 26 for a description of the features also available in the command window.

4.3.4 Debugging models

Improved error diagnosis of electrical circuits

The improvement of the error diagnosis of electrical circuits in the package Modelica.Electrical is being implemented in Dymola to display error messages for e.g. electrical circuits missing a ground object or connectors of an electrical component not being connected. A simple erroneous electrical motor circuit (missing the ground object)



will, when translated, as a part of the error message, display the following:

The model includes the following hints:

An electrical current cannot be uniquely calculated.

The reason could be that

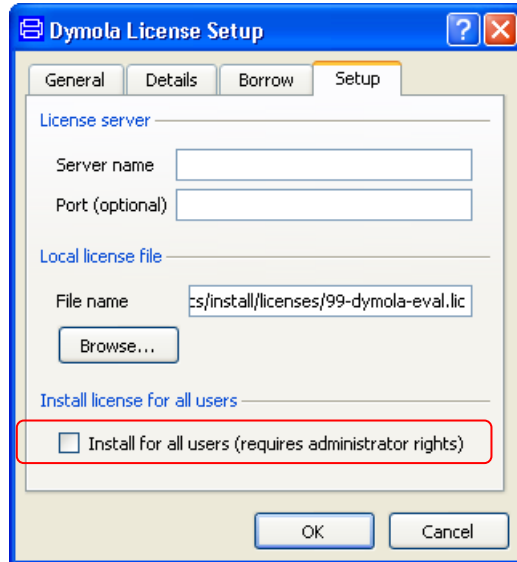
- a ground object is missing (Modelica.Electrical.Analog.Basic.Ground) to define the zero potential of the electrical circuit, or
- a connector of an electrical component is not connected.

See also section “Modelica Standard Library” on page 34.

4.4 Installation

4.4.1 Simplified installation of license file for all users

When setting up a sharable license or installing a node-locked license, the **Setup** tab for the command **Help > License...** has been enhanced.



You have the option of installing the license file only for the currently logged in user, or for all users on this computer. The latter requires administrator rights.

4.4.2 Remote installation of Dymola

Dymola (whether downloaded as a zip file or on CD) consists of a number of files (.msi and .cab). Remote installation of dymola.msi is possible using the appropriate tools, such as msixec. For example, the following command makes a quiet installation of Dymola and all libraries with Modelica version 3:

```
msiexec /i dymola.msi INSTALLLEVEL=201 /quiet
```

The value of the `INSTALLLEVEL` property controls which components are installed according to the table:

INSTALLLEVEL	Description
unspecified	Installs Dymola and standard libraries
201	As above and also installs commercial libraries compatible with Modelica language version 3.
301	As above and also installs commercial libraries compatible with Modelica language version 2.2.2
1001	As above and also installs Japanese translations of dialogs and menus

4.5 Other simulation environments

4.5.1 Dymola – Simulink interface

Compatibility

The Dymola – Simulink interface now supports Matlab releases from R14SP2 (ver. 7.0.4) up to R2009b (ver. 7.9). Official support for Matlab R14 (ver. 7) and R14SP1 (ver. 7.0.1), has been discontinued with Dymola 7.4. Only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. The LCC compiler is not supported.

4.5.2 Real-time simulation

Compatibility – dSPACE

In the selection of supported dSPACE releases we have focused 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, we 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.

Compatibility – xPC Target

Compatibility with Matlab xPC Target has been verified for all Matlab releases that are supported by the Dymola – Simulink interface. Only Microsoft VisualC compilers have been tested.

4.5.3 FMI for Model Exchange

Introduction

FMI stands for “Functional Mock-up Interface” and is a key development effort within the MODELISAR project, see

http://www.itea2.org/public/project_leaflets/MODELISAR_profile_oct-08.pdf

An important part of FMI is a specification for model exchange, allowing any modeling tool to generate C code or binaries representing a dynamic system model which may then be seamlessly integrated in another modeling and simulation environment.

The FMI for model exchange specification version 1.0 was released on January 28, 2010 and can be downloaded here

<http://www.functional-mockup-interface.org/>

It is also available in Dymola using the command **Help > Documentation**. The specification is separated into an execution part (C header files) and a model description part (XML schema). A separate model description is used in order to keep the executable footprint small. An implementation of the FMI for model exchange specification is called an FMU (Functional Mock-up Unit).

Support in Dymola

The Dymola FMI support consists of two built-in functions for FMU export and import, respectively.

translateModelFMU

Exporting FMU models from Dymola is achieved by the function

```
translateModelFMU(modelname, storeResult)
```

The input string `modelname` defines the model to open in the same way as the traditional `translateModel` command in Dymola. The Boolean input `storeResult` is used to specify if the FMU should generate a result file (`dsres.mat`). The function outputs a string containing the FMI model identifier on success, otherwise an empty string.

After successful translation, the generated FMU (with file extension `.fmu`) will be located in the current directory.

importFMU

Importing FMU models to Dymola is achieved by the function

```
importFMU(modelname, includeAllVariables)
```

The input string `modelname` is the FMU file (with the `.fmu` extension). The Boolean input `includeAllVariables` should be set to true if other variables than inputs, outputs and parameters should be included. The function outputs true if successful, false otherwise.

The Dymola FMU import consists of (1) unzipping the `.fmu` archive, (2) transforming the XML model description into Modelica, and (3) opening the resulting Modelica model in Dymola.

By setting the variable `includeAllVariables` to false, only inputs, outputs and parameters from the model description are included when generating the Modelica model. Such black-box import can be used as separate compilation of models to substantially reduce translation times.

System requirements

FMI is currently only supported for Windows.

Current limitations

FMU export

- The result file generation is currently only fully supported for the solvers `Lsodar`, `Dassl`, `Euler`, `Rkfix2`, `Rkfix3` and `Rkfix4` when importing the FMU in Dymola. For the other solvers, the number of result points stored will typically be too low. However, the values are accurate for the time-points at which they are computed.
- A static wrapper library is provided in addition to the DLL on the same directory level. This is not in accordance with the FMI standard which requires any static libraries to be

independent and provided in compiler specific subdirectories. The import function relies on this wrapper library being present.

- When imported by a target simulator, the generated FMU does currently not support inclusion of several instances.

4.5.4 Importing Simulink models (using FMI)

The Dymola distribution contains a package, `rtwsfcnfmfmi.zip`, which supports export of FMUs from Matlab/Simulink. This package contains an implementation of the FMI for model exchange specification on top of model code generated by Real-Time Workshop. The Matlab Target Language Compiler is used to construct the XML model description.

The zip archive is located in the `$DYMOLA/mfiles` directory, but since it is independent of Dymola it may be extracted to any location. See the included `README` file for more instructions on installation and usage.

This package for Simulink model FMU export together with the Dymola support for FMU import now facilitates simulation of Simulink models in Dymola.

4.5.5 Interfacing Dymola and SIMULIA software

It is now possible to interface Dymola and the following SIMULIA tools:

- Abaqus
- iSight
- Fiper

This could be used for e.g. co-simulation. An example of Dymola and Abaqus co-simulation of a high fidelity anti-lock brake system is presented in the paper <http://www.modelica.org/events/modelica2009/Proceedings/memorystick/pages/papers/0110/0110.pdf>.

For more information, please see www.Dymola.com/interfacing-other-software.

4.6 Migration

4.6.1 Conversion of location of class-level annotations

Dymola contains an automatic conversion that moves class-level annotations to last in the class as specified in the Modelica Language Specification.

4.6.2 Conversion of URI's

Dymola now supports the Modelica URI `'modelica:/'` scheme for images and links. Please see details of this and the available conversion script in section “Modelica URI scheme supported for images and links” on page 22.

4.7 Libraries

4.7.1 Modelica Standard Library

The sub-library Electrical has been improved; better error diagnosis has been implemented using `unassignedMessage` annotations. Such annotations have been added to the electrical connectors in the library to enable display of error messages if e.g. a ground object is missing in an electrical circuit or if a connector of an electrical component is not connected. For an example in Dymola, please see section “Improved error diagnosis of electrical circuits” on page 29.

4.7.2 Hydraulics Library

The new version 3.0 of Hydraulics Library has a number of major changes against previous versions, and is for that reason not fully backward compatible. Some changes were necessary to improve the model fidelity and the upcoming possibility to run the library also for thermo-hydraulics applications.

Some important changes are:

- A number of models have been completely redesigned, e.g. the oil models and the cylinder models.
- A number of models have been added, e.g. a elastic hose model, a new pump model and a new accumulator. Some now obsolete models have been removed as well.
- A non-backward compatible change is the connector modification, where the earlier volume flow rate variable q has been replaced with the mass flow rate variable m_flow . If new models have been built by connecting models from the library no problems should arise, otherwise it may be needed to manually edit the code. This was necessary to make sure the mass is always conserved. The volumes at the ports of all connectors are now also optional.

The name of the library has been changed to Hydraulics 3.0, earlier HyLib. In order to migrate from earlier versions (Hylib 2.5, 2.6 or 2.7) the conversion script `ConvertHyLib_from_2.7_to_3.0.mos` has to be run manually, see the manual “Dymola User Manual Volume 2”, chapter 9.1.1 “How to migrate”.

A beta version of the Hydraulic Element add-on to the Hydraulics Library is available to interested customers. It allows to model hydraulic component design, using only physical interfaces for connections between physical components. Such detailed components are fully compatible with the current Hydraulics Library and can be used mixed with the existing models. Interested customers may contact Modelon via email: info@modelon.se.

For backward compatibility reasons, the older version of the Hydraulics Library, HyLib 2.7, is available in the distribution and can be opened using **File > Open**.

4.7.3 Modelica_LinearSystems2

The library has been further developed; some improvements in version 2.1:

- Functions to solve Sylvester equation, to calculate nullspace and condition numbers of matrices have been added.
- The block Controller.Interpolator to increase the sampling frequency with linear interpolation and optional mean-value filtering is provided.
- Bode diagrams of state space systems are now based on numerical more reliable zeros and poles representation instead of on transfer functions.
- The block Controller.FilterFIR is now corrected and enabled.

4.8 Documentation

The handling of Dymola User Manual, in particular the version handling, has changed somewhat. The manuals were previously of the same version as the software (e.g. “Dymola x.y”). From now on the manuals will be of version “Month + Year”. This allows e.g. new releases of manuals between software releases, which make it easier to include more information and making it available for customers earlier.

In the distribution of Dymola 7.4 Dymola User Manuals of version “January 2010” will be present; these manuals include all relevant features/improvements of Dymola 7.4 presented in Release Notes, except the FMI handling, which is covered in a separate document.

5 Dymola 7.3

5.1 Introduction

There are many improvements and additions in Dymola 7.3 compared to 7.2. Main features are improvement in Modelica authoring and documentation. The Modelica programmer will benefit from features like code completion, automatic indenting and automatic syntax highlighting when programming, and from a new documentation editor and better rendering of formulas etc when documenting. A new information browser is also added. New Linux versions are supported.

5.2 Libraries

5.2.1 Modelica Standard Library

MSL 3.1 uses a number of new language elements compared to MSL 3.0 (e.g. `stream`, `inStream` and `actualStream` in `Modelica.Fluid`). A new sub-library `Modelica.Fluid` is also included (previously `Modelica_Fluid`). A number of new components have also been added to existing sub-libraries. For further details, please see the release notes of the library.

MSL 3.1 is the latest Modelica Standard Library and builds on the MSL 3.0. Version 3.1 is backward compatible to version 3.0 and 3.0.1; models developed with version 3.0 or 3.0.1 will work without any changes in version 3.1.

MSL 2.2.2 can be used for older models when conversion to Modelica 3.0 or later is not wanted. It is backward compatible with version 2.2.1 and 2.2 with a few exceptions in `Modelica.Media`.

The Dymola 7.3 distribution contains two versions of Modelica Standard Library (MSL): version 2.2.2 released in August, 2007 and version 3.1 build 4 released in August, 2009. MSL 3.1 is default in Dymola 7.3. See section 5.4.1 “New version of Modelica Standard Library used” on page 38 of this booklet for how to change default MSL version. When an old model using MSL 2.2.1 is opened, a conversion is made to the default MSL version.

5.2.2 Hydraulics Library

A new version of Hydraulics Library is now available, version 2.7. This version has some changes compared to previous versions.

- Addition of a new long-line model using a transmission line model.

- Steady-state initialization option added to basic volumes. The option has to be activated on the volume level.

5.2.3 Air Conditioning Library

AirConditioning Library version 1.7.1 is included. The updated library contains:

- An improved version of the ExcelInterface with several new features, e.g. better handling of structural parameters.
- SpatialPlot visualizer now uses degC values for temperature axis.

5.2.4 Vehicle Dynamics Active Safety Library

The library has now been extended with a new sub-option for modeling of systems related to Active Safety. The option contains the following:

- Base-classes, templates, components, and experiments for simulations of vehicles with active safety functions.
- Model fidelity selection: Models to mimic behavior of active safety systems
- Test experiments for vehicles: evasive lane change, J-turn.
- Vehicles.ControlUnits package with control units and related components, function blocks for ESP, ABS, TRC, ABD.
- Scenes package with objects and sensors that can detect objects, road curvature, and distance to closest objects.
- Vehicles.Chassis.SingleTrack package with model and experiment for parameter estimation and model reduction.
- Chassis motion sensors
- Brake systems with modulator unit and control unit

5.2.5 Modelica_LinearSystems2

The Modelica_LinearSystems library has been further developed with several new features.

- The operator overloading concept has been fully utilized.
- The ‘analyze’-function to determine the characteristic of a system (eigenvalues, zeros, controllability, stability, etc.) and to analyze the relation of the system states to the dynamics of the uncoupled modal states is provided.
- The ‘Design’ package contains functions for controller design, i.e. pole assignment, LQ controller, Kalman Filter, and LQG controller.
- The Controller package contains a sub package ‘Template’ which provides standard controller structures (e.g. a state-feedback-control-structure and a two degree of freedom controller template with an inverse system model in the feed forward loop).

5.2.6 Modelica_StateGraph2

A new library Modelica_StateGraph2 is included to model safe hierarchical state diagrams in combination with any Modelica model. This is an update of the old Modelica.StateGraph

library. All parts have been redesigned and significantly improved based on the experience with the experimental ModeGraph library. The library has been designed to simplify usage and improve safety aspects.

5.3 New information browser

A new information browser is used to display model documentation, e.g. when using any **Info** button or **Info** command in any context menu.

Information browser.

The screenshot shows a window titled "Modelica.Mechanics.Rotational.Components" with a sub-window for "IdealPlanetary". The page contains the following information:

Information

The IdealPlanetary gear box is an ideal gear without inertia, elasticity, damping or backlash consisting of an inner **sun** wheel, an outer **ring** wheel and a **planet** wheel located between sun and ring wheel. The bearing of the planet wheel shaft is fixed in the planet **carrier**. The component can be connected to other elements at the sun, ring and/or carrier flanges. It is not possible to connect to the planet wheel. If inertia shall not be neglected, the sun, ring and carrier inertias can be easily added by attaching inertias (= model inertia) to the corresponding connectors. The inertias of the planet wheels are always neglected.

The icon of the planetary gear signals that the sun and carrier flanges are on the left side and the ring flange is on the right side of the gear box. However, this component is generic and is valid independantly how the flanges are actually placed (e.g. sun wheel may be placed on the right side instead on the left side in reality).

The ideal planetary gearbox is uniquely defined by the ratio of the number of ring teeth z_r with respect to the number of sun teeth z_s . For example, if there are 100 ring teeth and 50 sun teeth then ratio = $z_r/z_s = 2$. The number of planet teeth z_p has to fulfill the following relationship:

$$z_p := (z_r - z_s) / 2$$

Therefore, in the above example $z_p = 25$ is required.

According to the overall convention, the positive direction of all vectors, especially the absolute angular velocities and cut-torques in the flanges, are along the axis vector displayed in the icon.

Parameters

Name	Description
ratio	number of ring_teeth/sun_teeth (e.g. ratio=100/50)

Connectors

Name	Description
sun	Flange of sun shaft
carrier	Flange of carrier shaft
ring	Flange of ring shaft

5.4 Developing a model

5.4.1 New version of Modelica Standard Library used

By default Dymola will start with Modelica Standard Library 3.1. If the user wish to use Modelica Standard Library 2.2.2 this can be selected using the command **Edit > Options...** selecting the **Versions** tab. Concerning new features in Modelica Standard Library 3.1 please see the section "Libraries" in the beginning of this booklet.

5.4.2 Window handling



A new button **Undock** is available for the package and component browser. By clicking on this button, the window is undocked from Dymola main window and is treated as a separate window. The same function was previously - and still is - available by double-clicking on the window header or dragging the header. By double-clicking on the header of the undocked window it will be docked again.

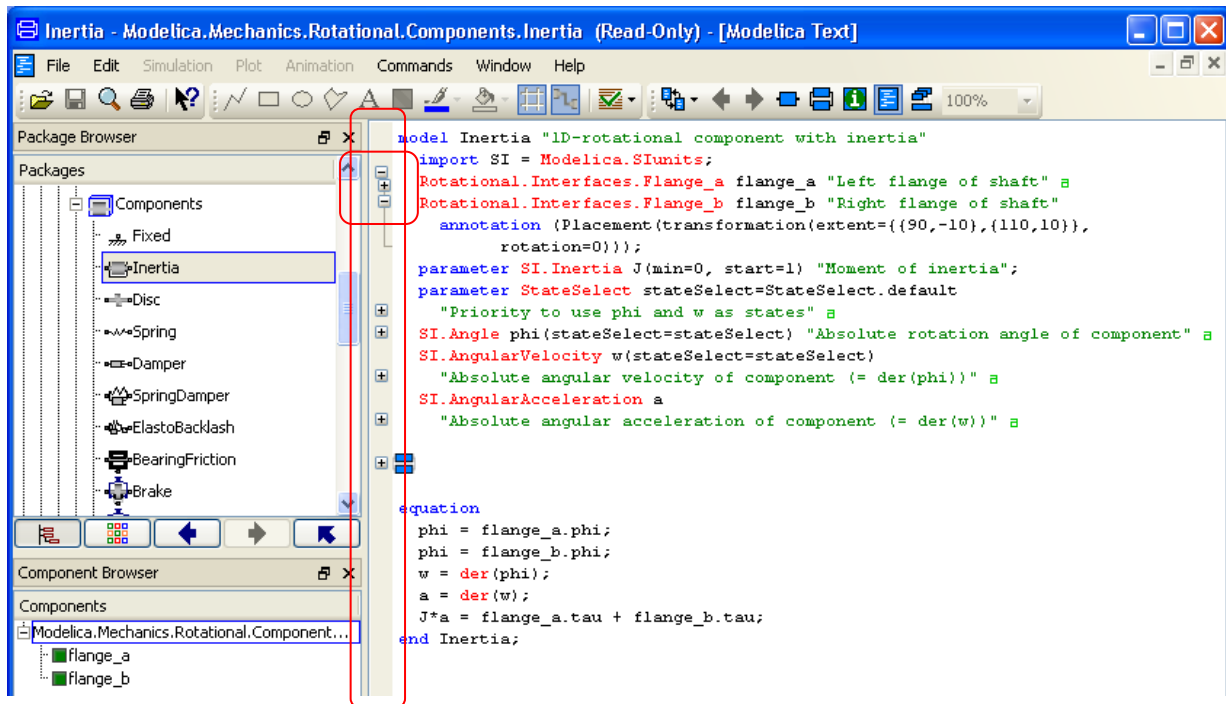
Zooming and panning

By holding the Control key while working in the diagram layer, it is possible to zoom and pan using the mouse. Please see the manual for details.

5.4.3 Modelica text editor

Improved structure of components, connections and annotations

In the Modelica text editor, a sidebar with icons enables individual expanding/collapsing of components, connections and annotations.



In the example the component annotation is expanded, revealing the flanges. The annotation of Flange_b is in turn expanded, revealing the placement annotation for that flange.

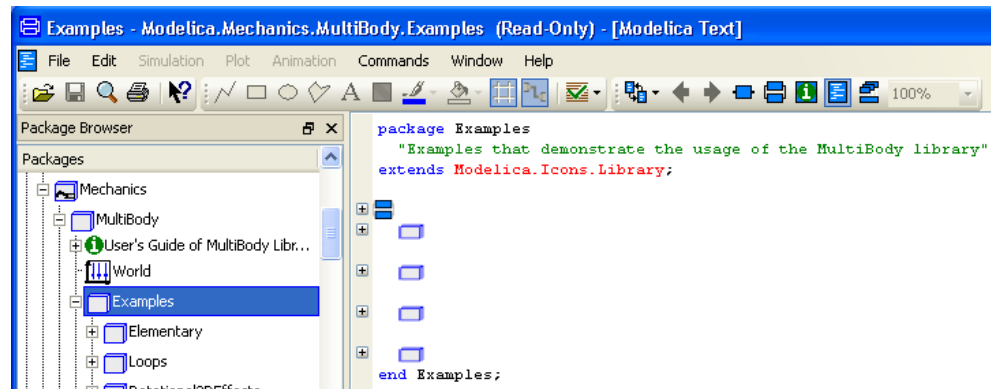
It is still possible to expand/collapse all annotations using the **Expand** entry in the context menu.

Individual expansions are remembered; if an annotation that contains an expanded annotation inside is collapsed, the expanded connection inside is showed again when the “primary” annotation is expanded again.

Improved structure for local packages

Local packages are packages defined in another package. An example (the four last symbols are local packages; they correspond to the four packages under Electrical in the package browser):

Local packages.



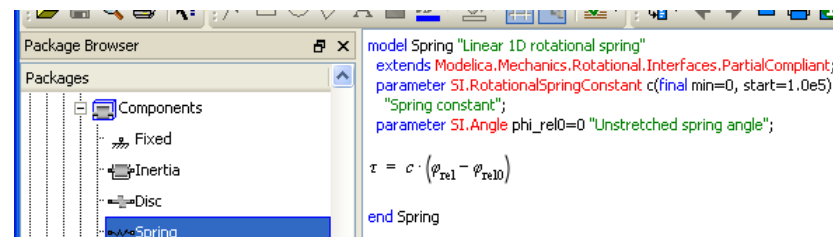
Local packages are by default only shown as symbols the Modelica Text layer. They can be expanded in two ways.

One way is to right-click in the Modelica Text layer and select the command **Expand > Load Local Packages**. All local packages will be loaded if needed, including any local packages deeper in the tree structure. Then each package can be individually expanded by clicking on the corresponding symbol or “+”.

By selecting **OK**, all local packages will be loaded, and the selected package will be expanded.

Mathematical notation for equations

Equations in the Modelica text can be rendered as mathematical formulae. This feature is enabled with the **Use mathematical notation** command in the context menu.



Automatic syntax highlighting and indentation

New text will be automatically syntax highlighted as you type, except types (e.g. Real). To get also types color coded, please right-click and select **Highlight Syntax** or press **Ctrl+L**.

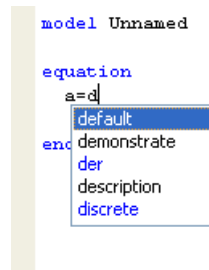
HTML text in annotation will have automatic color coding of tags as well. Please see section “Documentation” below.

Indentation is automatically handled while typing.

Code completion

Code completion can be activated by pressing **Ctrl+Space** in the Modelica Text Layer. This will bring up a context menu containing all the words beginning with the letters written so far or all words available if nothing has been written yet. As you type, the list of possible choices will decrease until only one word remains. If something not matching the words in the completion list is typed, the menu will automatically be closed.

Code completion example.



The screenshot shows a code editor with the following text: `model Unnamed`, `equation`, `a=d`, and `enc`. A context menu is open over the text, listing the following options: `default`, `demonstrate`, `der`, `description`, and `discrete`. The word `default` is currently selected and highlighted in blue.

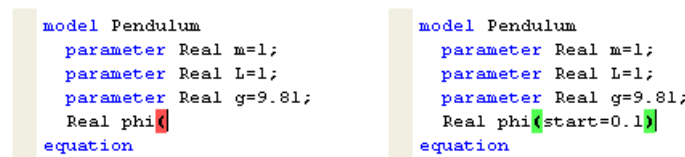
The code completion menu contains all the words from the current Modelica Text Layer, all Modelica keywords, highlighted in blue, as well as all the words from models previously shown in the Modelica Text Layer.

To select the word to input, either left-click on the word in the menu or navigate to the word with the arrow keys and press **Enter** to input the currently selected word. Closing the menu without inputting any word can be done by pressing **Escape** or clicking outside of the context menu.

Bracket handling

When an excess bracket is inserted, it is marked by red (figure to the left below). When the corresponding enclosing bracket is added, both brackets will be marked with green (figure to the right below).

Bracket handling.



The left screenshot shows a code editor with the following text: `model Pendulum`, `parameter Real m=1;`, `parameter Real L=1;`, `parameter Real g=9.81;`, `Real phi`, and `equation`. The closing bracket of the `Real phi` line is marked with a red square, indicating it is an excess bracket.

The right screenshot shows the same code editor with the following text: `model Pendulum`, `parameter Real m=1;`, `parameter Real L=1;`, `parameter Real g=9.81;`, `Real phi(start=0.1)`, and `equation`. Both the opening and closing brackets of the `Real phi` line are marked with green squares, indicating they are properly matched.

When clicking on any bracket, that bracket and the matching bracket will be marked by green (like in picture to the right above).

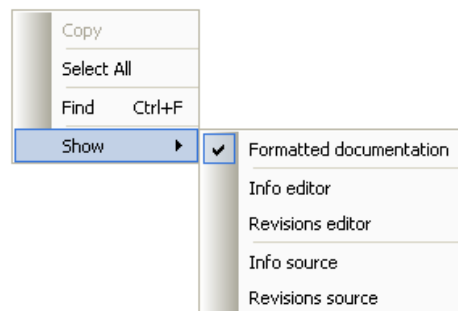
Similar bracket handling is also applied for brackets enclosing tags when working with HTML code in annotations.

5.4.4 Documentation

New documentation editor

The text in the documentation layer of the edit window can now be authored using a wysiwyg (“what you see is what you get”) documentation editor, if the class is not read-only.

By right-clicking in the Documentation layer for an editable class and selecting **Show** the following selections can be made.



Formatted documentation is the resulting documentation and the default selection when displaying the documentation layer.

Info editor will display the documentation editor for the information part of the documentation layer text.

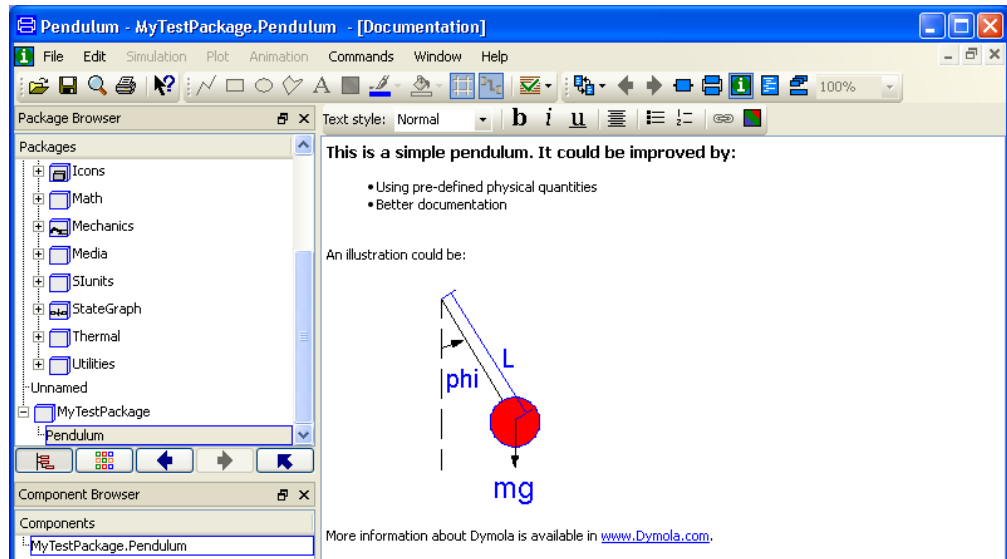
Revisions editor will display the documentation editor for the revisions part of the documentation layer text.

Info source will display the html code of the information part of the documentation layer text. The code is editable, with automatic color coding of tags and automatic indication of tag brackets (a single bracket will be red, enclosing brackets will be green).

Revisions source will display the html code of the revisions part of the documentation layer text. The code is editable in the same way as for the info source.

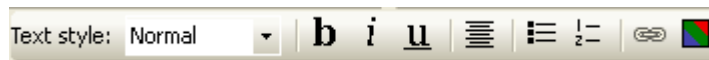
If the class is not editable the items **Info editor** and **Revisions editor** will not be displayed, and the html source code cannot be edited.

Example of documentation editor for info part.



The toolbar available:

Toolbar.



The two last buttons are for insertion of links and images, respectively. For more information about the buttons in the toolbar, please see Dymola User Manual Volume 1.

5.5 Simulating a model

5.5.1 Window handling



A new button **Undock** is available for the variable browser and command window. By clicking on this button, the window is undocked from Dymola main window and is treated as a separate window. The same function was previously - and still is - available by double-clicking on the window header or dragging the header. By double-clicking on the header of the undocked window it will be docked again.

5.5.2 Plot window

Legend with transparency

By default the legend now is drawn with transparent background not to hide curves.

The corresponding setting is found using in the plot setup menu reachable e.g. by right-clicking in the plot window, selecting the entry **Setup...** and then the **Legend** tab.

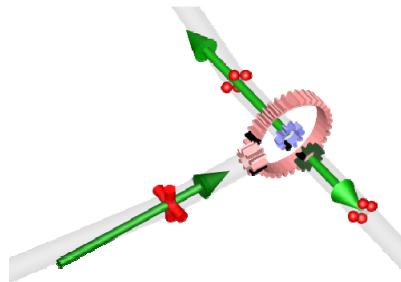
Zooming and panning

It is possible to zoom and pan in the plot window. See zooming and panning in the graphical editor.

5.5.3 Animation window

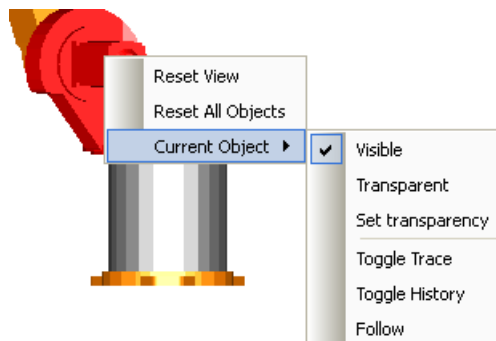
Visualization of forces

It is now possible in Dymola to create and display “pushing vectors” (the position of the end point of the tail of the visualized force/torque arrow is varied instead of the position of the head). The example below shows the visualization of the torque flow through an active rear differential.



Context menu

The context menu has been slightly changed:



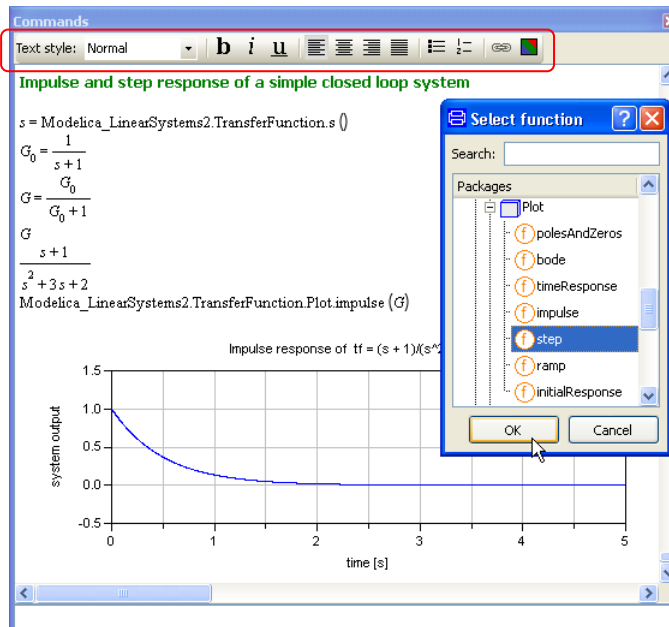
Unchecking the default selection **Visible** for a selected object will make it invisible and impossible to select.

Previous entry **Solid** has been removed; it corresponded to resetting the alpha value to 1.0 (the alpha value is reached using **Set transparency**).

5.5.4 Command window

The documentation editor available in the command window

A documentation editor is now available in the command window, facilitating e.g. the documentation of simulations. The documentation editor is the same as the one previously described – for details please see Dymola User Manual Volume 1. A flag has to be set to enable the editor, please see below.



Better rendering of expressions etc.

It is now possible to display Modelica expressions with mathematical notation. For a description of this feature please see below.

Plots automatically inserted in command log

Whenever a plot function is executed, the content of the plot window is inserted in the command log. Two flags have to be set to true to enable this feature; the general flag activating new command window features (see below) and the flag `Advanced.ScriptingWindow.IncludePlot`. Both flags are by default set to false.

Please note that displaying variables in the plot window by selecting variables from the variable browser does not in itself execute the plot function – then the plot function should be executed each time a variable is selected/deselected which would not be a wanted feature. The plot function will be executed (displayed in the command log) when another action is performed in Dymola (e.g. another simulation). However, the user can force Dymola to execute the plot function when working with the variable browser by clicking in the command line of the command window.

Animation window content automatically inserted in command log

Whenever the function `animationRedraw()` is executed, the content of the animation window will be inserted in the command log. Two flags have to be set to true to enable this feature; the general flag activating new command window features (see below) and the flag `Advanced.ScriptingWindow.IncludeAnimation`. Both flags are by default set to false.

The user can execute this function by entering `animationRedraw()` in the command line of the command window.

Activating the new command window features

The features of the documentation editor in the command window, mathematical rendering of expressions and plot window/animation window content in command log all requires setting the flag:

```
Advanced.ScriptingWindow.UseNewFeatures=true
```

The flag is by default set to false. Changing the flag from false to true (e.g. by typing in the command above in the command line of the command window) will have the following consequences:

When the next command is executed, the content of the command window will be erased and the new command given (and following commands and outputs) will be displayed in the new way, with better rendering.

This will also have implications concerning saving the content of the command window:

Once a new command is given in the “new mode”, the content displayed in previous mode cannot be saved as a .txt or .HTML file (using e.g. the commands **File > Save Log** or **File > Save Script > Command log**). Only the content of the “new mode” will be saved.

However, it is still possible to save the total content as a .mos file (using any of the commands mentioned), since that file is created in another way.

5.5.5 Scripting

Improved rendering of formulas, algorithms etc.

The command window is improved to show formulas, algorithms etc. in a more intuitive way. Please note that a flag has to be set to activate this feature, see above.

Rendering as text or not for inputs and outputs

The input from the command line in the command window is analyzed by Dymola. If the input is evaluated as an expression, the input is presented in a formatted way, “math formatting”. If the input cannot be evaluated as an expression (e.g. if semicolon “;” concludes the input) then the input will be presented as text.

The output is analyzed and presented the same way.

Input/output formatted as “math” can be selected as objects in the command log, however parts of such an object cannot be selected/copied. Input/output presented as text can be marked as usual text. The text in a “math” object will have a somewhat different style compared to a corresponding text.

Greek characters

If e.g. a variable name is interpreted by Dymola as a Greek character, Dymola will render that character. E.g. alpha is rendered α , and Gamma is rendered Γ . This also applies when the Greek character is trivially embedded, as for e.g. alpha_2, rendered α_2 . Trailing numbers following Greek characters are rendered as subscripts; e.g. alpha12 is rendered α_{12} .

Greek characters can of course also be used as indexes, e.g. x_beta will be rendered x_β .

The rendering of Greek characters is by default active, to disable it, set the flag

```
Advanced.RenderGreekLetters=false;
```

Index rendering

Index within brackets [] are rendered as subscripts, e.g. x[2] is rendered as x_2 and alpha12[3] is rendered as α_{12_3} .

The content after the last underscore _ in e.g. a variable name is rendered as a subscript. E.g. v_12 is rendered v_{12} . Exception: When indexing using brackets [] is used, that will overrule the above, e.g. v_12[3] is rendered v_{12_3} .

Some examples of rendering:

Modelica expression	Rendering
division <code>(a*(1+b))/(1+c)</code>	$\frac{a(1+b)}{1+c}$
square root <code>sqrt(1/(1+a))</code>	$\sqrt{\frac{1}{1+a}}$
power <code>(1+a)^(1+sigma)</code>	$(1+a)^{1+\sigma}$
absolute values <code>abs(1/(a-3))</code>	$\left \frac{1}{a-3} \right $

sum <code>sum({1,2,3})</code>	$\Sigma\{1, 2, 3\}$
simple matrix (declaration) <code>A=[1,2;3,4]</code>	$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
derivative of x <code>der(x)</code>	$\frac{dx}{dt}$
array construct <code>{i+j for i in 1:3, j in 4:7}</code>	$\{i+j \text{ for } i \in 1:3, j \in 4:7\}$
if-then-else-expressions <code>y = if x<0 then -x else if x==0 then 0 else if x==1 then 1 else x</code>	$y = \begin{cases} -x & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x = 1 \\ x & \text{else} \end{cases}$
logical expressions <code>p and not (p or q)</code>	$p \wedge \neg(p \vee q)$

The examples above do not cover all features, they just point to the fact that a more user-friendly rendering is implemented. This will of course also be the case when e.g. making a function call, the call

```
Modelica_LinearSystems.StateSpace.constructor([1,2;3,4],[1;2],
[1,2],[0]);
```

will give the result:

$$= \text{Modelica_LinearSystems.StateSpace} \left(A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, C = [1 \ 2], D = [0] \right)$$

5.6 Linux

5.6.1 New Linux versions supported

General

Dymola 7.3 runs on Red Hat Enterprise 5.1 with gcc version 4.1.1, and compatible systems. Please note that multi-criteria design optimization and coupling to Matlab/Simulink are not supported on Linux.

More Linux-specific notes are available in the “Dymola User Manual Volume 1”, chapter “Appendix – Installation”, section “Installation on Linux” or by using the command

```
man dymola
```

Improvement: GUI responsive during simulation

In previous Linux versions the GUI froze during the last part of the translation and the simulation. Now the GUI remains responsive at every stage. Also, a requested stop of an ongoing simulation now has immediate effect.

Full set of integrators supported

All integrator algorithms available in MS Windows are now also available on Linux.

5.7 Installation

5.7.1 New installation process on Linux

Dymola for Linux is distributed as an RPM package. The package is installed using the command

```
# rpm -i name-of-distribution.rpm
```

For installation on Debian or Kubuntu systems conversion to the deb format is required using the alien command:

```
# alien -k name-of-distribution.rpm
```

5.7.2 Dymola License Server

License server name

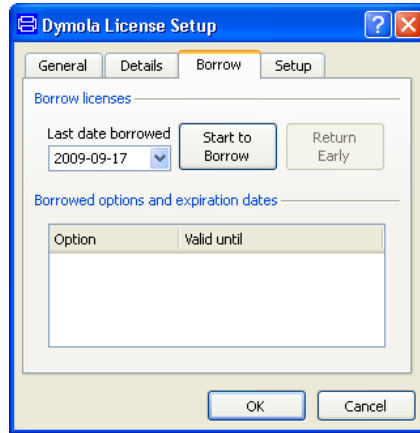
Before installation of the license server, the license file must be updated with the actual name (or IP-number) of the server. The license file contains a line identifying the server:

```
SERVER server.name.here 000102DE37CD
```

The part *server.name.here* must be changed to the name of the actual server before installing the license file. It should be noted that the last part (the hostid) cannot be edited by the user.

License borrowing

A new user interface for license borrowing from a license server has been implemented. Instead of running a separate utility program, borrowing and early returns is performed from Dymola, see **Help > License... > Borrow**.



Licenses are borrowed by selecting an end date and clicking on **Start to Borrow**. Dymola must then be restarted to borrow licenses. The list in the lower half of the dialog displays currently borrowed licenses and when they will be automatically returned to the server.

Currently borrowed licenses can be returned early when the computer is connected to the license server again.

5.8 Other simulation environments

5.8.1 Dymola – Simulink interface

Compatibility

The Dymola – Simulink interface supports Matlab releases from R14SP1 (ver. 7.0.1) up to R2009b (ver. 7.9). Only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. The LCC compiler is not supported.

5.8.2 Real-time simulation

Compatibility – dSPACE

In the selection of supported dSPACE releases we have focused 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, we 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.

Compatibility – xPC Target

Compatibility with Matlab xPC Target has been verified for all Matlab releases that are supported by the Dymola – Simulink interface. Only Microsoft VisualC compilers have been tested.

5.9 Advanced Modelica Support

Dymola supports the following features added in Modelica Language Specification 3.1

- Operator overloading.
- Preserving order for packages stored in separate files by automatically creating a `package.order` file.
- Creating arrays in expandable connectors.
- Automatic sizing of connectors.

And additionally:

- Improved handling of expandable connectors.
- Several improvements related to handling of arrays of records.
- Symbolic processing improved so that enabling of conditional components will not influence the rest.
- Automatic differentiation extended for more cases.

6 Dymola 7.2

6.1 Introduction

There are many improvements and additions in Dymola 7.2 compared to 7.1, especially regarding authoring of Modelica text. Furthermore, Modelica_Fluid library is included, as well as two additions to the Vehicle Dynamics Library; Trucks and Drivelines.

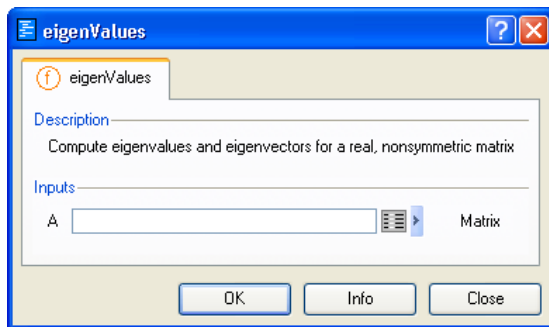
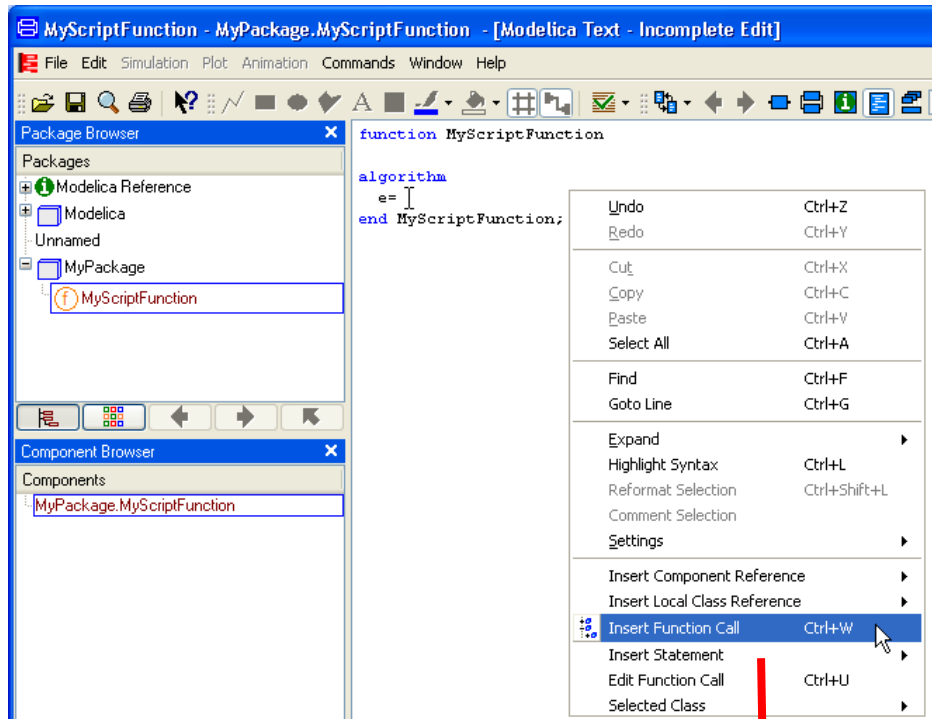
6.2 Developing a model

6.2.1 Graphical user interface

Insert Function Call simplifies making function calls

The context menu in the Modelica Text layer editor, Parameter dialog, Variable declaration, and Command Input line have a new entry **Insert Function Call...** that allows you to browse for functions and then enter the arguments, allowing easy insertion of a function call.

The below example shows how to use **Insert Function Call...** to insert the function call `Modelica.Math.Matrices.eigenValues` in the function `MyScriptFunction`.





Improved package browsing

A number of commands involve selecting a package (e.g. selecting what package to extend from, selecting where a package should reside).

Now a browser button is present beside the combo-box of packages in such menus to pop a package browser. This makes it more convenient to find the relevant package (in particular working with large packages).

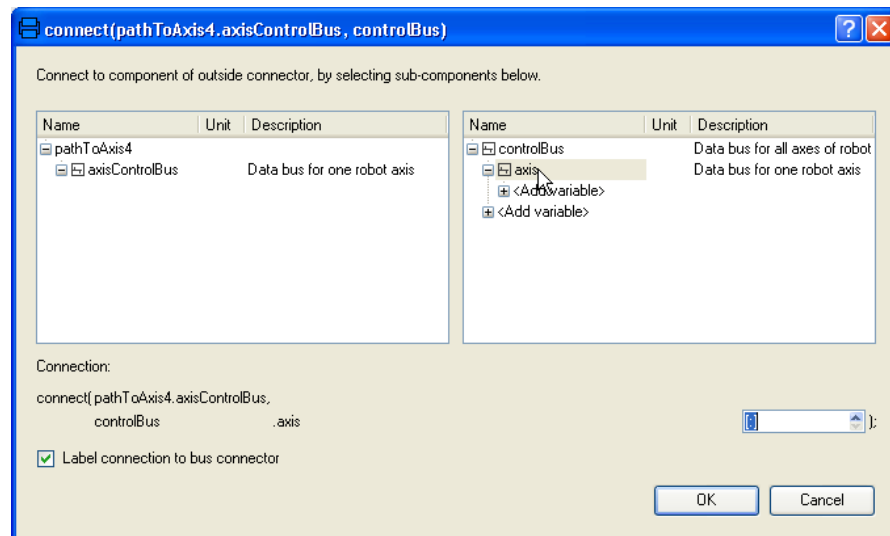
If a non-package is selected, the containing package is selected.

Other new features in the graphical user interface

- *Final (read-only) parameters and their values can be shown in the parameter dialog using the entry **Show final parameters in parameter dialog** in the **Appearance** tab reached by the command **Edit > Options...** The default is not to show these parameters.*

Expandable connectors

The handling of expandable connectors now also supports connection to a specific array slot in the expandable connector dialog.



It is possible to use parameter expressions inside the subscripts as well (even though there is a spinbox); and when introducing a new variable e.g. axis[1] can be used as well.

6.2.2 Unary Connectors

Creation of unary connections has been greatly simplified. A new annotation `__Dymola_connectorSizing` has been added to allow for automatic calculation of array indices.

Create an integer parameter with the new annotation:

```
parameter Integer nIn=0
annotation(Dialog(__Dymola_connectorSizing=true));
```

Then use the parameter as array index for a connector array:

```
StepIn nPorts[nIn];
```

When creating connections involving the connector array the array index for the connection will automatically be calculated. The first connection will have index 1, the second index 2, and so on. This feature is very useful for state machines and fluid libraries.

6.2.3 Annotation to define an inverse of a function

Functions may have an additional annotation to define an inverse of the function:

```
function f1
  input Real x;
  input Real y;
  input Boolean b[4,3];
  output Real z;
  annotation(__Dymola_inverse(y=f2(z,x,b)));
algorithm
  ..
end f1;
```

The meaning is that function "f2" is an inverse to the function "f1" where the previous output "z" is now an input and the previous input "y" is now an output.

The inverse requires that for all valid values of the input arguments of f2(z,x,b) and y being calculated as $y := f2(z,x,b)$ implies the equality $z = f1(x,y,b)$.

Function "f1" can have any number and types of arguments. The current restriction is that both "f1" and "f2" must have exactly one scalar Real output argument and that "f2" must have exactly the same arguments as "f1", but the order of the arguments may be permuted.

The annotation is exploited when solving nonlinear equations symbolically as described in the manual "Dymola User Manual", Section 8.9 "Symbolic solution of nonlinear equations in Dymola.

6.3 Simulating a model

6.3.1 Graphical user interface

Recent Windows button

A new command button, **Recent Windows**, is now by default available in the Plot toolbar. Clicking the button will toggle between the two last shown sub-windows.

Clicking the arrow displays a menu with all sub-windows available (plot, animation, diagram layer, visualizer). The menu alternatives for e.g. plot windows are based on the plot heading (if specified) or the names of the plotted variables.

6.3.2 Parameter values

When translating a model, it is checked that each parameter has a value. A parameter declared with the attribute `fixed = false` will be calculated at initialization from the initial equations or initial algorithms. In a general model library, it may be difficult to provide good values. Modelica allows the value of the attribute `start` to be used as a default value. Otherwise parameters must be given values through bindings.

Dymola issues an error for a parameter with `fixed = true` having neither value nor start value. This may be relaxed by setting the flag

```
Advanced.IssueErrorForUnassignedParameter = false
```

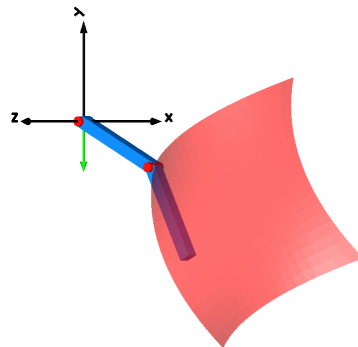
which turns the error into a warning and forces a zero (`false`, `""`) value to be used for the parameter.

Dymola issues a warning for a parameter with `fixed = true` having no value but a start value. The idea is to hint a user that the parameter value is generic and that the user ought to set it.

The warning may be suppressed by setting the flag

```
WarnAboutParametersWithNoDefault = false
```

6.3.3 Surface-objects with transparency



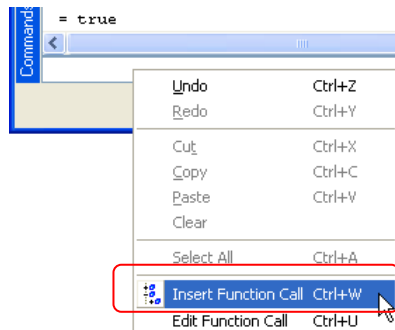
A new visualization object, `SurfaceFramed`, has been introduced. The object supports transparency and frame-input. Data (except array sizes) can be changed dynamically during a simulation. It is presently not possible to export such animation to VRML, or apply it to other objects.

6.3.4 Scripting

Insert Function Call simplifies handling of function calls

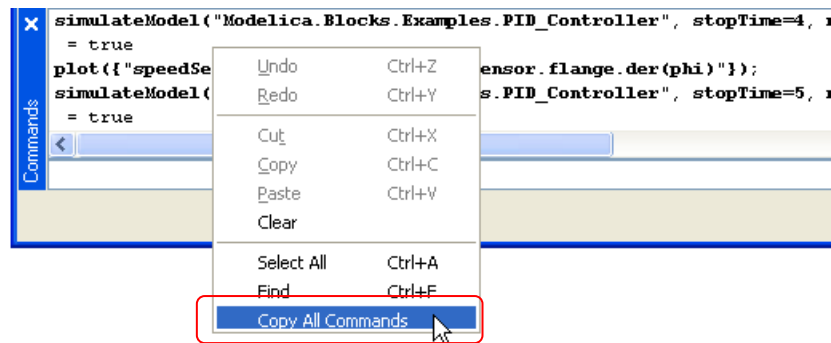
As been mentioned above, the new entry **Insert Function Call** is available in a number of context menus. For scripting, the new entry in the context menu of the Command input line and the Modelica Text layer editor are important, enabling easy inserting of functions when working with scripting. For these menus the shortcut **Ctrl+W** is also available.

Example of Context menu with Insert Function Call: Command input line.



Copying of commands from the command log

The context menu in the command log in the Commands window has been extended with the entry **Copy All Commands**. This makes it easy to use the commands given in scripting, e.g. by copying them to the Modelica text layer of a scripting function.



6.4 Installation

6.4.1 Installing a node-locked license

Obtaining relevant host id

The procedure obtaining *node-locked licenses* is changed; the relevant host id of the computer where Dymola should be used must be supplied to sales before purchase. The license that the customer will receive will contain this information.

To be able to easily find out the host id of a computer without having Dymola installed, a small file `hostid.exe` can be obtained from your Dymola distributor.

For information about installation in general, please see Dymola User Manual Volume 1, chapter 6 “Appendix – Installation”.

6.5 Other simulation environments

6.5.1 Dymola – Simulink interface

Compatibility

The Dymola – Simulink interface supports Matlab releases from R14SP1 (ver. 7.0.1) up to R2008b (ver. 7.7). Matlab R13 (ver. 6.5), R13SP1 (ver. 6.5.1), and R13SP2 (ver. 6.5.2) are no longer officially supported in Dymola 7.2.

The new DymolaBlock GUI that was introduced with Dymola 7.0 is now supported for Matlab R2006a to R2008b. Only Visual Studio C++ compilers are supported to generate the DymolaBlock S-function. The LCC compiler is not supported.

The DymolaBlock GUI

The GUI has been improved to better support the use of workspace references (instead of numerical values) for parameters and start values. When entering a workspace reference in the GUI it is first checked if the variable exists (a warning is issued otherwise) and then if the dimensions match (otherwise it is an error). It is possible to enter workspace references in the GUI before defining them in the workspace. In this case, however, there will be no check of matching dimensions.

The GUI also allows for matrix variables to be entered numerically using their actual size (and not only on row-major form as before). The manipulation to vector form is done internally. Entering a matrix variable in this way again requires that the dimensions match the corresponding Modelica variable.

Scripting

Two new features have been introduced to simplify the setting of DymolaBlock parameters and start values from the Matlab command prompt and from scripts.

- During compilation, `dsin.txt`, is copied to `<modelname>.txt`. This makes it possible to load parameters and start values using the `loaddsin` utility for several models (`dsin.txt` only refers to the block that was compiled last).
- A new Matlab utility, `setParameterByName`, has been introduced to simplify setting elements of the parameter and start value vectors without having to manually search for the variable.

Example use for the robot demo model:

```
>> [p,x0,pnames,x0names]=loaddsin('fullRobot.txt');
>> p=setParameterByName(pnames,p,'mLoad',25);
>> p=setParameterByName(pnames,p,'mechanics.world.mue',4.5320e14);
>> x0=setParameterByName(x0names,x0,'axis1.gear.spring.phi_rel',10);
>> setParametersFDSin('untitled/DymolaBlock',pnames,p,x0names,x0);
```

6.5.2 Real-time simulation

Support for multi-tasking of DymolaBlocks

The Dymola simulation routines have been updated to enable linking of separately compiled DymolaBlock S-functions into a single real-time application, making it possible to use several DymolaBlocks in Simulink models intended for real-time targets. This facilitates use of the Simulink multi-tasking feature for more efficient utilization of computational resources in real-time simulations.

Compatibility – dSPACE

In the selection of supported dSPACE releases we have focused on releases that introduce support for a new Matlab release and releases that introduce a new version of a cross-compiler tool. In addition, we 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.

Compatibility – xPC Target

Compatibility with Matlab xPC Target has been verified for all Matlab releases that are supported by the Dymola – Simulink interface, Only Microsoft VisualC compilers have been tested.

6.5.3 DDE communication

Dymola DDE commands

Commands can be sent just to be executed, or with special DDE-request to allow the caller to collect the result from Dymola (since a normal execute does not allow more advanced return codes.)

The special DDE-requests are: “ModelicaString:expression” and “MatlabString:expression”, in both cases the result of the expression (usually a function call) is returned as a string, containing a literal expression in one of the syntaxes (of Modelica or Matlab)..

Example (Matlab, calling the function `Modelica.Math.Matrices.solve` for solving a real system of linear equations $A*x=b$ where A is a matrix and x and b are vectors. The format [1,1] means that the clipboard format CF_TEXT is used for the for the request and that the result is returned as a string. `eval` is used to make Matlab to execute that resulting string as an expression or statement.):

```
>> ch=ddeinit('dymola','model');
>> res=eval(ddereq(ch,'Matlabstring:Modelica.Math.Matrices.
solve([1,2;3,4],[1,5])',[1,1]))
```

Note 1: Multiple outputs are now supported.

Note 2: You should adjust the timeout in the calling program (e.g. Matlab) to allow for the command to complete. (In the Matlab example above, the timeout is by default 3000 ms; if a timeout of 4000 ms is wanted, the last line would be:

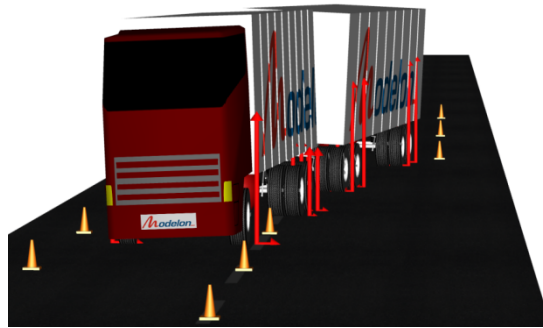
```
>> res=eval(ddereq(ch,'Matlabstring:Modelica.Math.Matrices.
solve([1,2;3,4],[1,5])',[1,1],4000))
```

6.6 Libraries

6.6.1 Vehicle Dynamics Library

Vehicle Dynamics Trucks

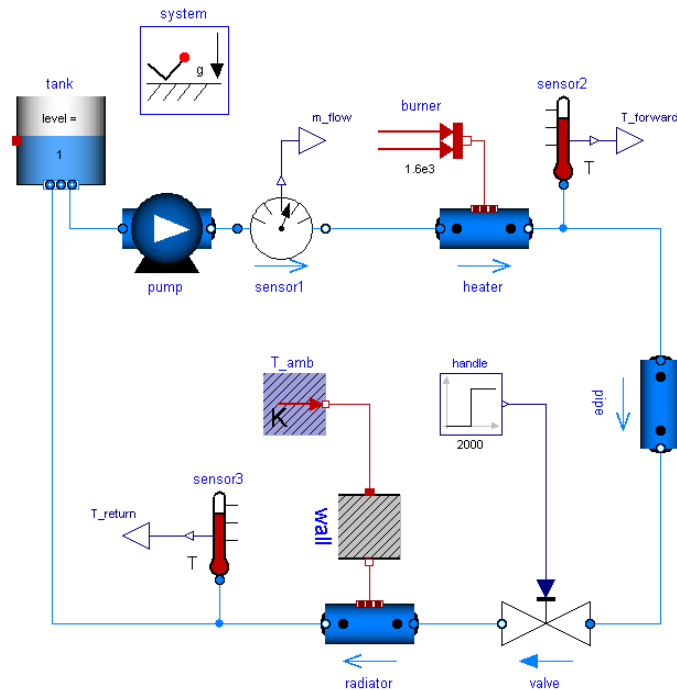
Heavy vehicles benefit particularly from the open and flexible structure of the Vehicle Dynamics Library due to the large variety of configurations of axles, frames, loads, etc. The Trucks option of the Vehicle Dynamics Library covers combinations such as trucks with full trailers and tractors with semi-trailers. The component set is complemented with models of air suspension, liquid loads, twin tires, and more.



Vehicle Dynamics Drivelines

With the Drivelines option, it is possible to study the interaction between the driveline and the chassis on a more detailed level; including wind-up, joint-bending, reaction forces and torques. Efficient models of for example shafts makes it possible at a fraction of the cost compared to multi-body models. The Drivelines extension also contains templates for convenient realization of different driveline layouts.

6.6.2 Modelica_Fluid Library



Modelica_Fluid is a free library from Modelica Association. The library contains components describing 1-dimensional thermo-fluid flow in network of pipes. A unique feature is that the component equations and the media models as well as pressure loss and

heat transfer correlations are decoupled from each other. All components are implemented such that they can be used for media from the Modelica.Media library (a part of Modelica Standard Library).

6.6.3 Automotive Demos Library

The new Automotive Demos Library demonstrates how components from a number of commercial Modelica libraries can be combined to model a wide range of automotive systems in the form of complete automotive example models.

The examples include a car with an active four wheel drive system, a car with detailed damper models, a hybrid car with a power split device and a hybrid car ready for export to Simulink.

The number of examples available depends on which of what libraries you have license for. License for the PowerTrain Library is always needed.

6.6.4 Hydraulics Library and Pneumatics Library

Positions of cylinder, directional control valves and all other valves where the position of the valve spool gives feedback about the system state are now animated in the Diagram layer in Simulate mode. In addition, visualizers for pressure and mass flow can be used to represent mass flows via arrow sizes and pressure levels via colors.

6.7 Advanced Modelica Support

6.7.1 User-defined derivatives – improvements

It is possible to declare the derivative to a function in order to be able to differentiate it if the automatic differentiation of functions in Dymola is not sufficient.

Previously generating Jacobians for non-linear systems involving functions with records and/or matrices as both input and output did not always work, nor using multiple outputs. This has now been corrected.

7 Dymola 7.1

7.1 Introduction

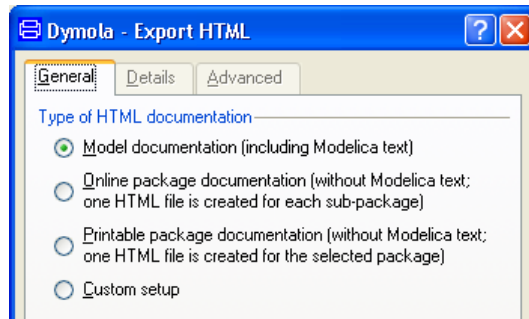
The most significant changes in Dymola 7.1, compared to 7.0, are support for Modelica stream connectors, realtime simulation and code export. This part of the booklet describes improvements and added features in Dymola version 7.1 compared to the earlier version 7.0.

7.2 Developing a model

7.2.1 Graphical user interface

HTML document generation menu improved

The menu for generation of HTML documentation has been improved:



The three first selections in the Type of HTML documentation group correspond to the most common needs for documentation. Ticking the **Custom setup** enables two more tabs that can be used to modify any other setting.

No option has been removed; they are just displayed in a more user-friendly way. For more information, please see relevant section in “Dymola User Manual Volume 1”.

Documentation of classes extended from

Documentation in the documentation layer and in generated HTML now contains a list of the class(es) extended from. The names can be clicked on to view the extended class(es). The extends are presented in the order they are declared, separated by comma.

Conditional connector rendering

The rendering of conditional connectors has been changed; when using the context command **Show Component** conditional connectors are rendered with a dotted rectangle, if disabled, just like conditional components.

7.2.2 Expandable connectors

The handling of expandable connectors has been redesigned to reconstruct the causality also for variables declared in the connector (normally declared with unit but without causality to ensure that the same connector declaration can be shared for both sources and sinks). This enables the check to always find a source for each causal signal if one exists - either internally in the model or externally (provided there is an expandable connector in the top-model).

7.2.3 Modelica language support

Stream connector support

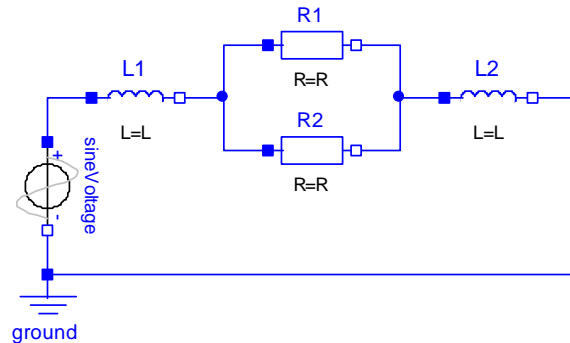
Dymola 7.1 supports in full the stream connector concept accepted for inclusion in Modelica specification 3.1:

- The prefix `stream`.
- The operators `inStream` and `actualStream`.
- Generation of connection equations.
- Automatic adding of `smooth/noEvent` for `m_flow*actualStream(h)`.
- The proposed regularization.
- Symbolic check that models are balanced.

Index reduction

The support of index reduction has been improved to deal with some implicit constraints between dynamic variables. As described in “Dymola User Manual Volume 2”, section “8.8.3 Index reduction”, the index reduction algorithm focuses on the structure of the equations. The practical experience of this approach is good. However, there are examples where the results is not satisfactory. Such examples appear typically in models of electrical circuits.

Consider the example



The model has two potential state variables, namely the currents through the inductors: $L1.i$ and $L2.i$. However, they cannot be selected as states simultaneously because these currents must be equal, $L1.i = L2.i$. In this example it is easy to see for a human being. Unfortunately, the original Pantelides's algorithm will not detect this constraint, because it is too implicit in the model equations. After alias elimination of the connector variables, the zero sum current equations for the connections between the inductors and the resistors are

$$\begin{aligned} L1.i &= R1.i + R2.i \\ L2.i &= -R1.i - R2.i \end{aligned}$$

Assuming $L1.i$ and $L2.i$ to be states, the structural analysis of Pantelides's algorithm indicates that $R1.i$ and $R2.i$ can be solved from these two equations. However, this is not true and the simulation will fail because the system actually is singular from that respect. It is easy to see by just adding the two equations giving $L1.i = L2.i$.

The improved support of index reduction looks for such kinds of implicit constraints between potential state variables and manipulates the equations to make them explicit.

7.3 Simulating a model

7.3.1 Improved simulation initialization

Starting a simulation of a model with values from a previous simulation (e.g. to be able to start from a pre-defined stable state) has been improved; a larger number of final values are saved when using the command **File > Save Script...** and then ticking **Variables**, **Final** and **All**.

7.3.2 Scripting language

The new built-in function `DymolaVersion()` returns the version and date of Dymola as a string.

7.4 Installation

7.4.1 C compiler notes

The C++ compiler Visual Studio.Net 2003 Toolkit (7.1) is no longer supported. The only supported free compiler is Visual Studio 2008 Express Edition.

7.4.2 Dymola License Server

The option **License server** is now available when selecting components to install. Installing this component *only* will create a directory `C:\Program Files\Dymola 7.1\bin` where the files needed for setup of a Dymola License Server is available. No installation of the Dymola program is thus needed on the server.

7.5 Other simulation environments

7.5.1 Dymola – Simulink interface

Compatibility

The Dymola–Simulink interface does not support Matlab versions prior to Matlab 6.5 (R13).

DymolaBlock GUI

A new feature has been added in the Simulink DymolaBlock GUI:

- A new checkbox to add the compiler option `bigobj` when compiling large models.

The new checkbox is available both for the new DymolaBlock GUI (supported for Matlab R2006b and later) introduced with Dymola 7.0 and for the old version of the GUI that is launched for unsupported Matlab versions.

7.5.2 Real-time simulation

The support for real-time hardware-in-the-loop simulation has been improved. The following now apply to all supported real-time platforms

- The model initialization is performed in *MdlStart* to avoid overruns in the first time-step.
- The manual has been updated with up-to-date step-by-step configuration instructions

The type of models that can be exported for real-time simulation is restricted. See section “Code and model export” starting on page 67 below.

RT-LAB

RT-LAB is no longer supported.

7.6 Code and model export

Dymola 7.1 has support for exporting models and model source code. Three export alternatives with different functionality are provided. Furthermore, run-time licenses are provided in order to allow models developed with a standard (no export) Dymola license to be simulated on other computers.

The export options are described in detail in the separate manual document “Dymola Code and Model Export”. Dassault Systèmes also provides a template project, *StandAloneDymosim*, that describes how to interface models exported with Binary Model Export and Source Code Generation to standard integration routines to build stand-alone applications. This project together with the `dsmodel.c` model API are also described in the document “Dymola Code and Model Export”.

7.6.1 Real-time simulation

The Real-time Simulation option enables the model to be used in environments not supporting the Microsoft C compilers. The option is specifically designed for real-time platforms, such as the dSPACE and xPC platforms that are supported by Dymola for Hardware-In-the-Loop (HIL) simulation. The following restrictions apply

- Real-time Simulation only allows export of models that use inline integration, i.e., that have embedded fixed-step integrators.
- The run-time routines exported using Real-time Simulation does not include the most advanced routines that are normally included from binary libraries. Most notably, models with dynamic state selection cannot be used in real-time simulations.

7.6.2 Binary model export

The Binary Model Export option allows the model to be exported to other Windows computers without requiring a Dymola license at the target system. The simulation functionality of the exported model is the same as on a computer having a Dymola license.

7.6.3 Source code generation

Source Code Generation exports code that can be used on any platform without the need of a Dymola license at the target system. A special translation command, `translateModelExport`, should be used with Source Code Generation and a number of built-in flags are available that can be used to modify the contents of generated model code.

Source Code Generation allows export of readable and well-documented code facilitating inspection, debugging, profiling, etc. This makes this export option suitable for advanced model-based applications, such as rapid prototyping.

The Source Code Generation option includes the functionality provided by Real-time Simulation (without the inline integration restriction) and Binary Model Export when models are translated in Dymola or Simulink.

The Binary Model Export and Source Code Generation options both allow export of symbol table information, e.g., model structure, variable names, types, and units as an XML file.

7.6.4 Dymola run-time

Dymola run-time licenses are introduced to enable models developed by users that lack export options to be run at other computers. Dymola run-time works both for the Dymola Simulator (`dymosim.exe`) and for models developed with the Simulink interface. The location of the license file containing the run-time options should be specified using the environment variable `DYMOLA_RUNTIME_LICENSE`.

7.7 Libraries

7.7.1 VehicleDynamics

A new VehicleDynamics library that supports Modelica 3.0 is now available. This library is a commercial library that demands a license.

7.8 Documentation

Dymola User Manual

Dymola User Manual Volume 1 and Volume 2 have been updated to cover Dymola up to the present version 7.1. **Please note** that the indexes of the on-line manuals are now “clickable” (links).

Other documentation

A new document is available: “Code and Model Export” (not included in the distribution, available for those having the relevant options)

8 Dymola 7.0

8.1 Introduction

This part of the booklet describes improvements and added features in Dymola version 7.0 compared to the earlier version 6.1.

8.2 Developing a model

8.2.1 New version of Modelica Standard Library used

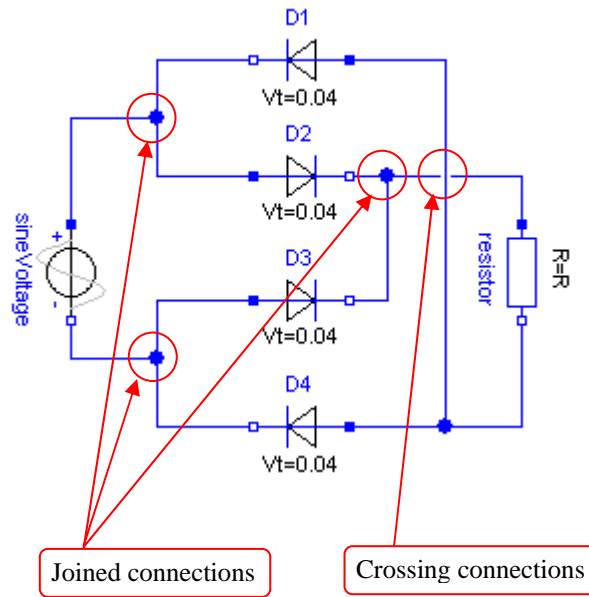
By default Dymola will start with Modelica Standard Library 3.0. If the user wish to use Modelica Standard Library 2.2.2 this can be selected using the command **Edit > Options...** selecting the **Versions** tab. Concerning new features in Modelica Standard Library 3.0 please see the section “Libraries” in this booklet.

Modelica Standard Library versions 1.6 and 2.2.1 are no longer part of the Dymola distribution.

8.2.2 Graphical user interface

Rendering of joined and crossed connections improved

The rendering of joined and crossing connections has been improved. Joined connections are rendered with a circular “blob” while crossing connections are rendered using a space.



A circular *hollow* “blob” indicates graphical overlapping of connections of sub-components from the same hierarchical connector (e.g. a bus).

Context menu for graphical primitives added

Cut	Ctrl+X
Copy	Ctrl+C
Delete	Del
Duplicate	Ctrl+D
Order	
Manhattanize	Ctrl+M
Smooth	
Flip Horizontal	
Flip Vertical	
Insert Point	
Remove Point	
Rotate 90	Ctrl+R
Rotate -90	
Set Corner Radius...	
Edit Annotation	

The context menu is reached by right-clicking on a graphical object. New features in this menu are:

- **Smooth** available for lines, polygons and connections. This feature can also be reached from the **Line Style** menu.
- **Insert Point** / **Remove Point** available for lines, polygons and connections. It is possible to add a point (corner) selecting the entry **Insert Point**. The point (corner) is inserted in

the point closest to the cursor in the selected object. An existing point (corner) can be deleted by placing the cursor in the vicinity and use the entry **Remove Point**.

- **Set Corner Radius...** enables the creation of a rectangle with rounded corners.
- **Edit Annotations...** enables structured editing of annotations for graphical object(s) in a hierarchical tree.

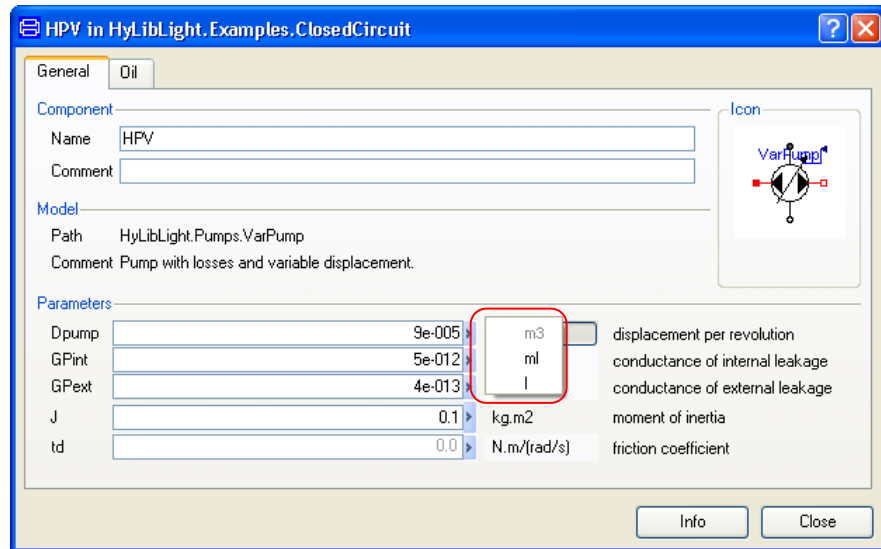
Other new features in the graphical user interface

- *Origin representation* – the origin is rendered by a red cross when any operation working on the origin has been applied, e.g. a rotation or a flip.
- *Bitmap storage can be selected* – the bitmap selection dialog has been extended with a checkbox that allows the user to specify whether the bitmap should be stored in the model or as an external file reference.
- *Elliptic arcs can be created* – by double-clicking on an ellipse a menu will pop up that enables the user to set the start and end angle of an elliptic arc.
- *Better control of printing coordinates* by using the **Map coordinate units to mm on print** entry in the **Appearance** tab reached by the command **Edit > Options...** .
- *Selection whether movements of connectors should be applied in both the icon and diagram layer* or not by using the **Apply movement of connectors to both layers** entry in the **Appearance** tab reached by the command **Edit > Options...** .
- *Attributes can be copied between the icon layer and diagram layer and vice versa* using the entries **Copy attributes to Icon layer** / **Copy attributes to Diagram layer** respectively in the **Graphics** tab reached by the command **Edit > Attributes....** Attributes affected are the ones in the menu.
- *Aspect ratio can be preserved when resizing components* using the entry **Preserve aspect ratio when resizing components of this class** in the **Graphics** tab reached by the command **Edit > Attributes....**

8.2.3 Unit checking and unit deduction

The Dymola feature of unit checking was introduced in Dymola 6.1.

- The Dymola feature for checking of units is now enabled by default, i.e., `Advanced.CheckUnits=true` by default.
- The unit checking has been further improved, in particular the error messages give clear hints for common errors.
- Deduction of units is enabled by default, i.e. `Advanced.DeduceUnits=true`. The feature deduces units of variables based on how they appear in equations. Logging can be enabled by setting `Advanced.LogDeducedUnits=true`. The deduction of units may detect additional unit inconsistencies.
- Deduced units for variables without units are shown in the variable browser.
- The attribute `displayUnit` is supported for input in parameter dialogs and automatically used for plotting as well.



- When the display unit is changed, the value in the parameter edit field is automatically converted to the chosen display unit if it hasn't been edited.

Note: For the special case of temperatures (in general – any type with an offset in the conversion) a new annotation on the type is needed to differentiate between absolute temperature / temperature difference: `annotation(__Dymola_absoluteValue=true)`. (The default is absolute.)

8.2.4 Modelica language support

Modelica 3 semantics is supported. When Modelica Standard Library 3.0 is loaded, Dymola applies the Modelica 3 semantics and uses Modelica 3 graphical annotations. The result is e.g. possibility to develop and maintain larger models, better possibilities for good diagnostics (because e.g. the requirements on local variables/equations balancing) and a better GUI. Dymola can deduce the number of variables and equations symbolically. This may allow Dymola to deduce that the equations are size consistent and that the model is balanced for any legal set of parameter values. This feature is enabled in the Modelica 3 mode when checking and the flag `Advanced.SymbolicSizeCheck=true`. To enable detailed diagnostics set `Advanced.LogSymbolicSizeCheck=true`.

Some features/improvements in Dymola:

- The behavior of the square root function (`sqrt`) has been changed to yield an error if taking the square root of a negative number. (Previously a zero was returned. The old functionality can be obtained either for all cases by setting `Advanced.GuardedSqrt=false` or for a specific case replacing `sqrt(x)` by `sqrt(max(0,x))`.)
- The behavior of the absolute value function (`abs`) has been changed due to Modelica 3.0 requirements. The function has been changed to not yield any event when changing sign.

(The old functionality of generating an event when changing sign can be obtained by setting `Advanced.AbsSignEvent=true`.)

- Declaration order of instantiated models has been modified so that inherited elements are placed where the `extends`-clause is (and not always first); this change is especially important for functions and records.
- Modelica allows vendor-specific annotations. All Dymola specific annotations can be preceded by `__Dymola_` as specified in the Modelica language specification.
- In general, identifiers can start with underscore (although not recommended if not vendor-specific annotations).
- Enumerations are partially supported..
- Partial derivatives of functions now support the correct syntactic construct `function foo_d=der(foo, d);`
- `break` and `return` are supported in functions. Note: classes or components named `break`, `return` or `der` will generate warnings. They should be removed.
- Within statements are now checked for correctness and written to the files (including empty within-statements at the top). Previous versions of Dymola can read within-statements ensuring that this does not break compatibility.

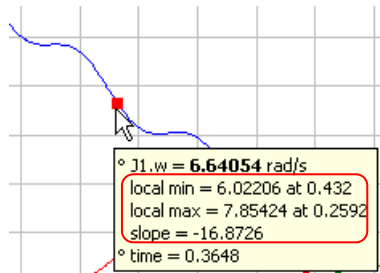
8.3 Simulating a model

8.3.1 Plot window

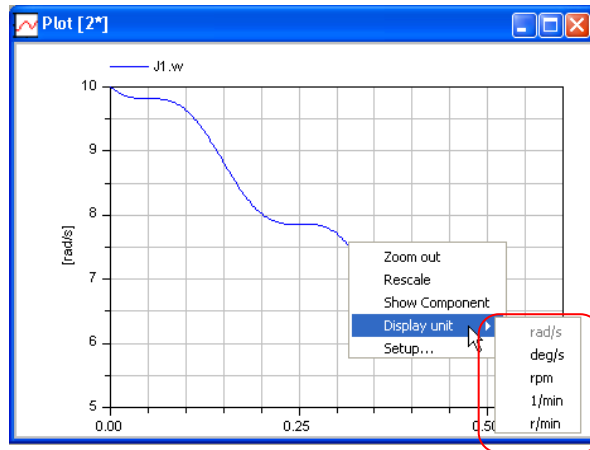
Dynamic tooltip introduced

Resting the cursor over a plotted signal displays a dynamic tooltip with information about the closest data point. That data point is also highlighted. The closest local min and max value is also shown, as well as the slope. If several signals are close to the cursor, data for all of them are shown; if possible a common value for the independent variable is used.

Tooltip for a single signal.



Context menu added



The context menu is reached by right-clicking in a plot window.

- **Zoom out** replaces earlier zoom out behavior.
- **Show Component** will display the component that contains the variable corresponding to the selected curve. The component will be shown selected in a window displaying the diagram layer, with relevant zooming. If a window containing the diagram layer is not open, such a window will be opened automatically.
- **Rescale** will rescale the window to the needed resolution to show the plotted curves.
- **Display unit** enables changing the display unit (see figure above).
- **Setup...** corresponds to the command **Plot > Setup...**

Horizontal zoom added

Pressing SHIFT + left mouse button and keeping pressed while moving the mouse to the left or right will span a “ruler”. The corresponding part of the x axis will fill the window when the left mouse button is released.

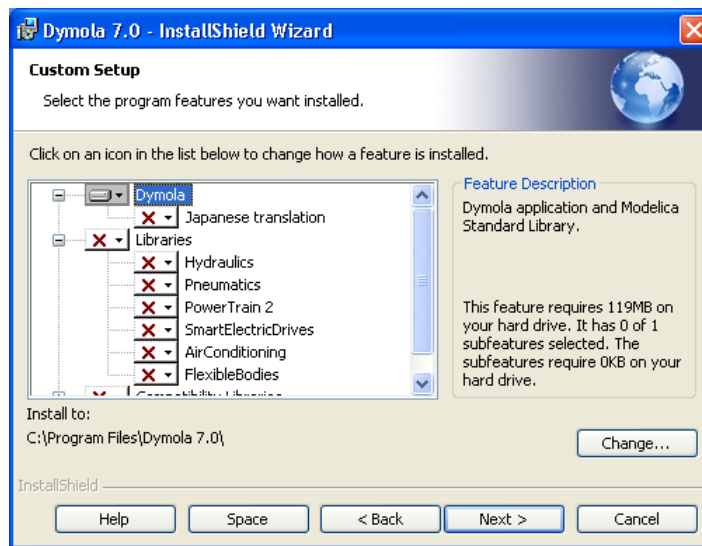
8.3.2 Diagram layer in Simulate mode – improved usage

- When selecting **Show variables** in the context menu of a selected component in the diagram layer window the component is highlighted in the variable browser and the last plot window is automatically activated.
- A selection in the variable browser automatically highlights the corresponding component in the diagram layer window if active.
- A diagram layer window is automatically opened if demanded by a command (e.g. **Show Component** in the plot window context menu or a **Display** command in the variable browser context menu

8.4 Installation

8.4.1 New installation process on Windows

A new installation process is used when installing in Windows. Dymola and all libraries are distributed on the same CD, every component being compatible and of the relevant version. The selection of what components to install is very convenient, as is a modification of the installation (adding of libraries etc.).



Directory handling

The default directory when installing is `Program Files\Dymola + the version number of Dymola`. This means that there is no need to remove previous version(s) of Dymola.

Do not delete or rename the Dymola directory once installed. Windows keeps track of all installed directories and the **Repair** operation in **Add/Remove Programs** assumes that the directory is unchanged.

C compiler notes

To translate models in Dymola you must also install a supported C compiler. The C compiler is not distributed with Dymola. The C compiler needs to be installed only once, even if you install multiple versions of Dymola.

Dymola supports Microsoft Visual Studio 2008, both the Professional edition and the Express edition. Dymola also supports older Microsoft compilers (Visual Studio 6, Visual Studio .NET 2003 and Visual Studio .NET 2005).

To download the free Express edition compiler please visit Microsoft's website. Note that you need administrator rights to install the compiler.

<http://www.microsoft.com/express/vc/>

The C compiler can be installed before or after you install the Dymola. You can run Dymola and browse models, but to translate any model you must install the C compiler.

Please note that earlier free versions of the Microsoft compiler are not supported; the reason is that they do not include a full set of Windows libraries. We recommend Visual C++ 2008 or later.

Selecting compiler might be required.

To change the compiler Dymola uses to translate the model, use the command **Simulation > Setup...** and the **Compiler** tab. This is required if you have used GCC compiler with earlier versions of Dymola.

8.5 Model Management

8.5.1 Encryption in Dymola

The support for licensed encrypted libraries has been extended to specify multiple options. As an example consider:

```
annotation(__Dymola_Protection(Library=
    {"LicenseOption1 LicenseOption2", "LicenseOption3"}));
```

In this case the user has to have LicenseOption3 **or** (LicenseOption1 **and** LicenseOption2).

8.5.2 Model and library checking

The regression testing is improved, now the testing also compares model translation statistics. Using this option, the statistics of the translated model can be included in the regression testing to detect changes in, for example; the number and sizes of linear and nonlinear system of equations and the number of state variables.

8.6 Other Simulation Environments

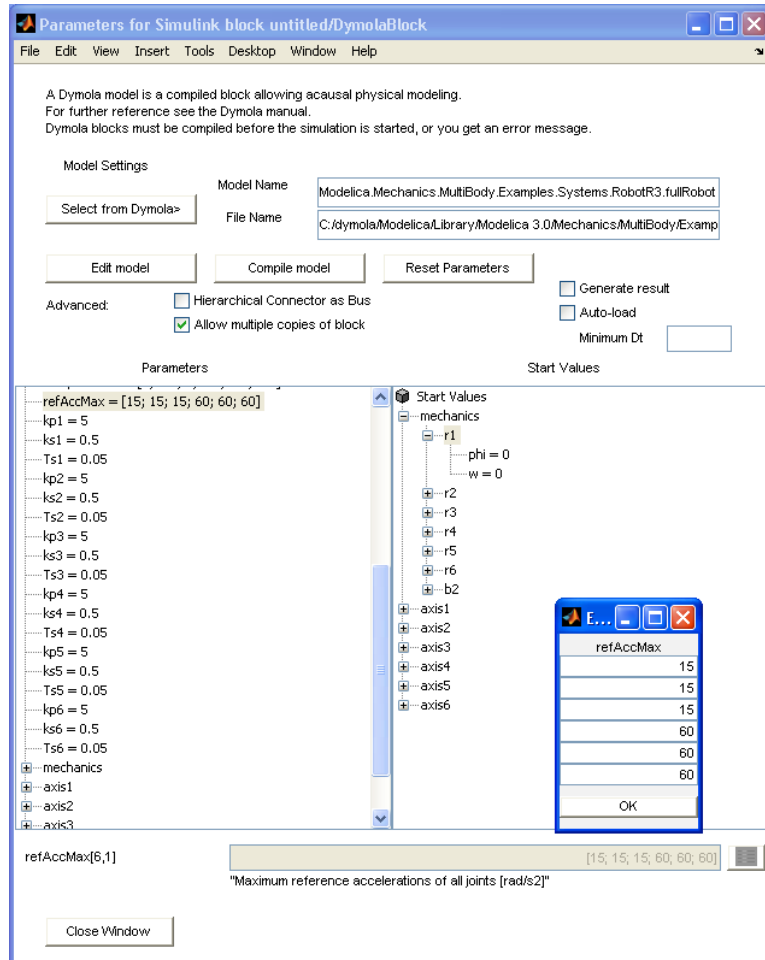
8.6.1 Dymola – Matlab interface

New GUI for the Simulink DymolaBlock

An improved GUI has been developed for the Simulink DymolaBlock. The important new features are

- Tree views for parameters and start values for easier navigation.
- Description strings and sizes of matrix variables are shown.

- An editor may be launched to enter values for matrix variables.



It should be noted that the new GUI is Java-based and only compatible with recent Matlab versions (Matlab R2006b and later). The old version of the GUI is launched automatically for unsupported Matlab versions.

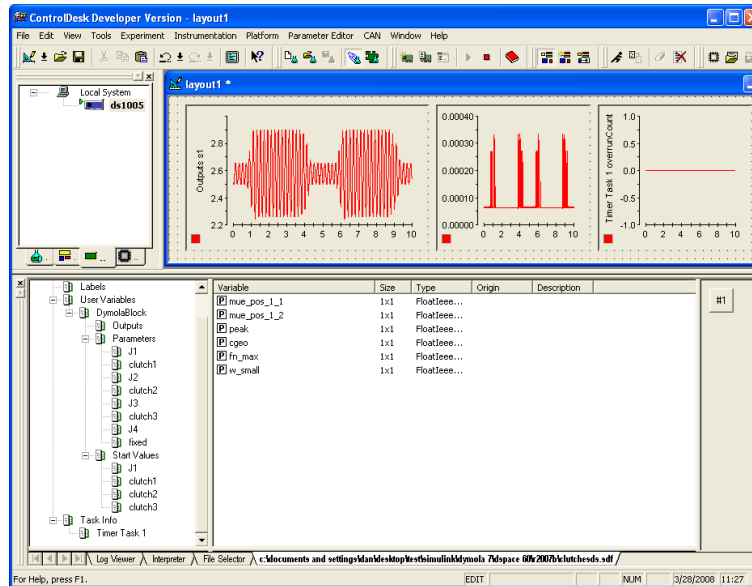
8.6.2 Real-time simulation

dSPACE systems

The following improvements apply specifically to dSPACE systems.

- The Dymola manual has been updated with step-by-step configuration instructions for recent dSPACE and Matlab versions.
- The model initialization is performed in *MdlStart* to avoid overruns in the first time-step.

- A new Matlab function *dym_rti_build* is provided to build real-time Dymola models for dSPACE. Enter *help dym_rti_build* in Matlab for a detailed description of the function.
- **Note:** *dym_rti_build* does not work with dSPACE 6.x and Matlab 2007b due to a bug in a dSPACE RTI function. Please contact Dymola.Support@3ds.com to obtain a bug fix if you are running in this configuration.
- In addition to code generation, compilation and loading, *dym_rti_build* also creates a user variable description file required for improved display of DymolaBlock variables (outputs, parameters, and start values). The Dymola variables are accessed as *User Variables* in the dSPACE ControlDesk variable browser:



8.7 Advanced Modelica Support

8.7.1 Symbolic solution of nonlinear equations in Dymola

Dymola has new features for symbolic solution of non-linear equations.

- Solving a nonlinear equation with single appearance of the unknown by applying function inverses.
- Solving a nonlinear equation with special patterns for the unknown.
- Partitioning of a system of equations into a linear and non-linear (one variable) part.
- Using min and max values of variables to possibly collapse if-then-else clauses.

8.8 Migration

The conversion script handling has been extended to allow conversion to Modelica 3.

8.9 Libraries

8.9.1 Modelica Standard Library

The Dymola 7.0 distribution contains two versions of Modelica Standard Library (MSL): 2.2.2 released on August 31, 2007 and 3.0 release on March 24, 2008. MSL 3.0 is default in Dymola 7.0. See section 8.2.1 of these release notes for how to change default MSL version. When an old model using MSL 2.2.1 is opened, a conversion is made to the default MSL version.

The Modelica Standard Library 2.2.2 is backward compatible with 2.2.1 and 2.2 with a few exceptions in Modelica.Media. Many new components have been introduced. For details, see the [release notes](#) of the library (12 pages).

Modelica Standard Library 3.0 has been adapted to the Modelica 3.0 semantics, i.e. with restriction for balanced models and new graphical annotations. The Rotational and Translational libraries have also been changed and initialization in the Mechanics libraries has been simplified. For details, see the [release notes](#) of the library (13 pages).

8.9.2 Modelica_LinearSystems

Bode plot of stateSpace systems and linearized Modelica models is now possible in the library Modelica_LinearSystems.

8.9.3 VehicleInterfaces

A new VehicleInterface library for building automotive models by plug and play is available.

8.9.4 FlexibleBodies

A new analytic FlexibleBodies library with flexible beam models is available. This library is a commercial library that demands a license.

8.10 Documentation

The Dymola User Manual has been updated and now consists of two volumes – [Volume 1](#) describes the fundamentals, while [Volume 2](#) describes more advanced features. The main changes, updates and additions are:

The User Manual consists of the previous manual “Dymola User Manual” and the manual “Dymola User Manual – Dymola 6 Additions” + some new material. The structure of the new manual has been changed compared to the old ones to reflect the above description.

Material from “Release Notes” and the previous chapter “Recent features in Dymola” in the old manuals have been integrated in the appropriate chapters. This also means that the present manuals cover Dymola up to the present version 7.0.

Please note that the indexes of the manuals are now “clickable” (links).