UM Control

This manual will help you to understand how to integrate models created in Matlab/Simulink with models created in Universal Mechanism software and conversely how to include UM models into Matlab/Simulink models.

2024

Contents

Getting Started Using Universal Mechanism: Matlab/Simulink interface

This manual leads you through examples of importing Matlab/Simulink models in Universal Mechanism (UM) and vice-versa examples of importing UM models in Matlab/Simulink models.

It is supposed that you already have studied the gs_UM.pdf[1] manual, which is devoted to basics of UM modeling and know how to create a new model, add new bodies and joints, generate and compile equations of motion (**UM Input**) and simulate mechanical systems (**UM Simulation**).

In this manual we will consider two examples. The first one is the simulation of an inverted pendulum with the control system that keeps the pendulum in inverted vertical position. The second one is the example that illustrates inclusion of the DC motor to a shaft. The second example shows starting the shaft up to nominal angular velocity and applying load to it. Time histories of current, voltage, electromagnetic moment and angular velocity of the shaft are shown.

Mechanical parts of both models are usual UM models. Models of the control system for the inverted pendulum and the DC motor are prepared in Matlab/Simulink and then UM and Matlab/Simulink models are coupled. The combination of the mechanical and control parts gives us a possibility to analyze dynamics of the complex system. The coupling of UM and Matlab/Simulink models can be done with the help of (1) the **Matlab Import** tool that exports Matlab/Simulink models as a Dynamic-Linked Library and then loads it in UM and with the help of (2) the **CoSimulation** tool that supposes export of UM models and using them in Matlab/Simulink as an S-function.

It is assumed that you go through the manual step by step sequentially. Information given in some sections can be shortened or even omitted below.

[1] www.universalmechanism.com/download/90/eng/gs_um.pdf

Compatibility

UM Control / Matlab Import

The Matlab/Simulink interface is supported by the **Matlab Import** tool from the **UM Control** additional module. Before coming to the rest part of the manual please check if the **UM Control/Matlab Import** is available on your computer. Run **UM Simulation** and from the **Help** menu select **About**. The list of the available modules is shown in the **Configuration** section.

The current version of the **UM Control/Matlab Import** supports the interface with Matlab 7.12-13 (R2011), Matlab 8.X (R2012-R2016) and Matlab 9.X / R2016-R2021. The x64 version of the UM Control/Matlab Import supports the interface with Matlab 8.X x64 (R2013-R2015) and Matlab 9.X / R2016-R2021.

To compile your own Matlab/Simulink models you need the following Matlab toolboxes to be installed: **Matlab Coder** and **Simulink Coder**. You can check if these toolboxes are installed using **ver** command in the Matlab command window.

In addition, to compile your own Matlab/Simulink models you need one of the supported Microsoft compilers to be installed. You can see supported Microsoft compilers for actual and previous Matlab releases. To compile your own Matlab/Simulink models for Universal Mechanism you need of one supported version of the Microsoft Visual C++ compiler. Other compilers including MinGW and Intel Parallel Studio are not supported. Please note that the only listed versions of Microsoft Visual C++ are supported. Higher or lower versions are not supported. You can find a list of supported compilers via one of the links above. Then find the **Supported Compilers** section. Finally check supported Microsoft Visual C++ versions for the MATLAB Coder, see pictures below.

Previous Releases: System Requirements and Supported Compilers

Release	Windows	Linux	Mac	Solaris/UNIX	Supported Compilers	Platform Availability
R2021a (MATLAB 9.10)	Details	Details	Details	N/A	Details	Details
R2020b (MATLAB 9.9)	Details	Details	Details	N/A	Details	Details
R2020a (MATLAB 9.8)	Details	Details	Details	N/A	Details	Details
R2019b (MATLAB 9.7)	Details	Details	Details	N/A	Details	Details

MATLAB Product Family – Release 2021a									
Compiler	MATLAB For MEX-file compilation, loadlibrary, C++ interface, and external usage of MATLAB Engine and MAT-file APIs	MATLAB Coder For all features	GPU Coder For all features	SimBiology For accelerated computation	Fixed-Point Designer For accelerated computation	HDL Coder For accelerated testbench simulation	HDL Verifier For DPI and TLM component generation	Audio Toolbox For validating and generating audio plugins	ROS Toolbox For custom messages and code generation
MinGW 6.3 C/C++ (Distributor: *mingw-w64*) *Download Now* Available at no charge	✓	✓		✓	✓	✓	✓		
Microsoft Visual C++ 2019 product family	✓	✓	✓	✓	✓			✓	
Microsoft Visual C++ 2017 product family [4]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Visual C++ 2015 Professional [4]	✓	✓	✓	✓	✓	✓	✓	✓	
Intel Parallel Studio XE 2020 for C/C++ [3]	✓	✓		✓	✓				
Intel Parallel Studio XE 2019 for C/C++ [3]	✓	✓		✓	✓				
Intel Parallel Studio XE 2018 for C/C++ [3]	✓	✓		✓	✓				

To be successfully exported Matlab/Simulink models should contain continuous elements only. Discrete elements are not supported.

UM Control / CoSimulation

Export UM models to Matlab/Simulink is supported by optional **CoSimulation** tool from the **UM Control** module. Run-time support of imported UM model under Matlab/Simulink environment is provided by the mathematical core of Universal Mechanism as a COM-server that should be also installed on your computer **(UM COM Server)**. All necessary UM components are installed automatically during installation of UM itself.

The current **UM Control/CoSimulation** version supports the interface with **Matlab 7.X** and **Matlab 8.X** (releases **R14** and later).

Note! 32-bit version of **Matlab Import** are compatible with 32-bit Matlab versions only. 64-bit Matlab versions are supported only in 64-bit version of UM.

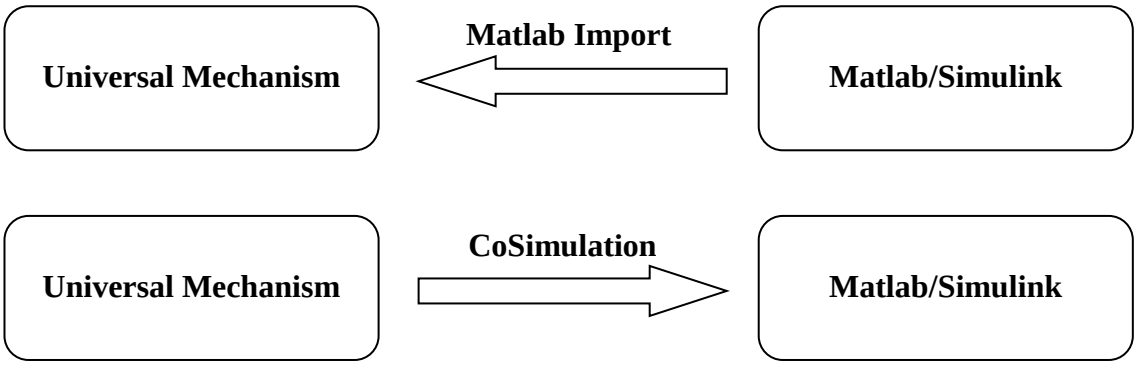
Copyright and trademarks

1. Preface

Two approaches to link mechanical models in Universal Mechanism software with Matlab/Simulink models are presented in this manual. Let us consider them in details.

1. Matlab Import tool

This approach supposes that Matlab/Simulink model firstly exported from Matlab/Simulink and compiled as Dynamic-Linked library (DLL) and then loaded into Universal Mechanism using the **Wizard of external libraries**. Simulation of dynamics of the complex model that includes both UM and Matlab/Simulink models is carried out in UM side.



2. CoSimulation tool

The second approach supposes that Matlab/Simulink model includes special **S-Function** block that presents model of mechanical part of the system imported from UM. Exporting UM model and generating the corresponding m file is done with the help of **Wizard of export to Matlab/Simulink.**

Export/import directions shown above in fact reflects the host application in that environment simulation of dynamics of the complex model is performed. Data exchange for both mentioned approaches anyway is bidirectional.

2. Description of models

In this section the models of the inverted pendulum and the DC motor are considered. We discuss in details features of mechanical models, including its parameters and details of preparing UM models to be used along with Matlab/Simulink models.

2.1. Inverted pendulum

2.1.1. Model description

The model of the inverted pendulum is shown in Figure 2.1. The model consists of a cart and a pendulum. Parameters of the model are given in Table 1. Control scheme has one input – ψ angle (deviation of the pendulum from the vertical position) and one output – F force, which should be applied to the cart to keep the pendulum in the inverted position.

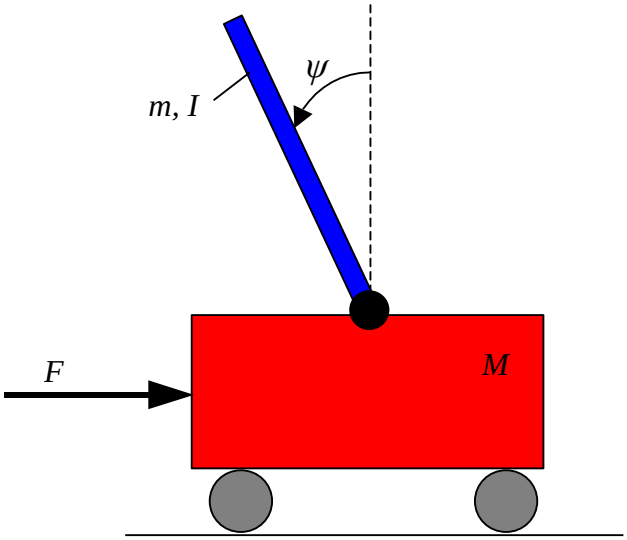


Figure 2.1. Inverted pendulum

Table 1

Parameters of the model

Parameter (in Figure)	Parameter (model)	Comment	Value
F	force	Force applied to the cart	
M	mass_cart	Mass of the cart	0,5 kg
m	mass_pend	Mass of the pendulum	0,2 kg
I	ix	Moment of inertia of the pendulum relative to its axis of rotation	0,006 kg·m2
	l	Distance between center of mass of the pendulum and its axis of rotation	0,3 m
	b	Friction coefficient of the cart	0,1 N/m/sec
Ψ		Pendulum angle from vertical	rad

2.1.2. Preparing UM-model

Although Matlab/Simulink DLL is imported in the **UM Simulation** program at the stage of its simulation we should provide the future interconnection between UM and Matlab/Simulink models already at the stage of the model description.

If the control scheme calculates output force(s) and/or torque(s), which should be applied to the mechanical system, we have to introduce such force elements into the UM-model when creating the model in the **UM Input** program.

In the model of the inverted pendulum *a general force* was introduced in the model. The force acts on the cart and its Y-component is given as the force parameter, see Figure 2.2. We will assign the control scheme output to the force parameter in the **Wizard of external libraries**. In this manner we apply the control force to the cart.

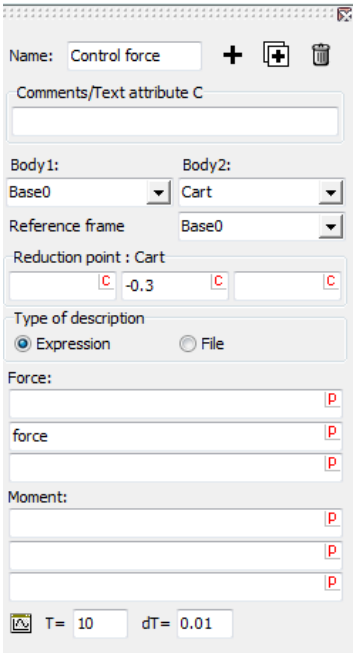


Figure 2.2. General force for connecting with control scheme

2.2. DC motor

2.2.1. Model description

The mechanical part of the model consists of a shaft with moment of inertia J_o, Figure 2.3. The shaft has one revolution degree of freedom relative to the base (inertial frame of reference).

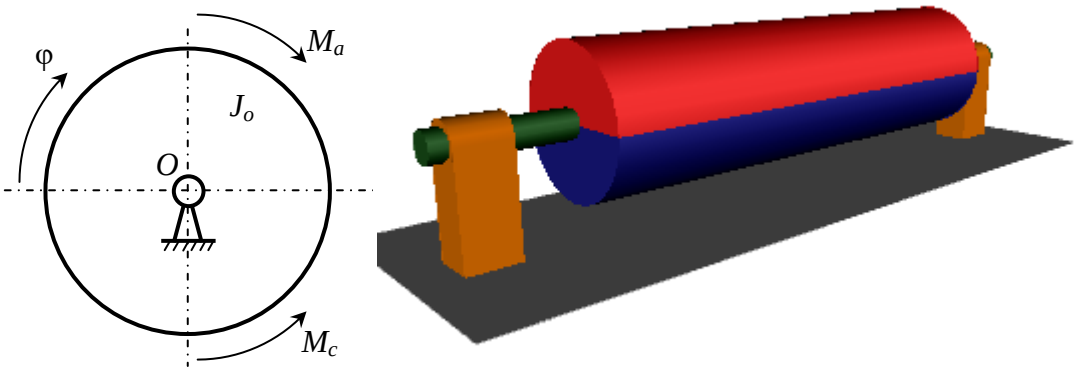


Figure 2.3. General view of the model

Model of the DC motor is described in Matlab/Simulink. It is supposed that the rotor of the DC motor and the shaft are connected rigidly. The model of the DC motor has one input and three outputs.

Input:

- angular velocity of the shaft (rotor).

Outputs:

- active electromagnetic moment M_a,
- current in an anchor circuit,
- voltage in an anchor circuit.

Active electromagnetic moment Ma is introduced as a *general force* element. Resistance moment M_r is a stepped time function in a revolution joint, see Figure 2.4. The stepped time function is used as an obvious case in this lesson.

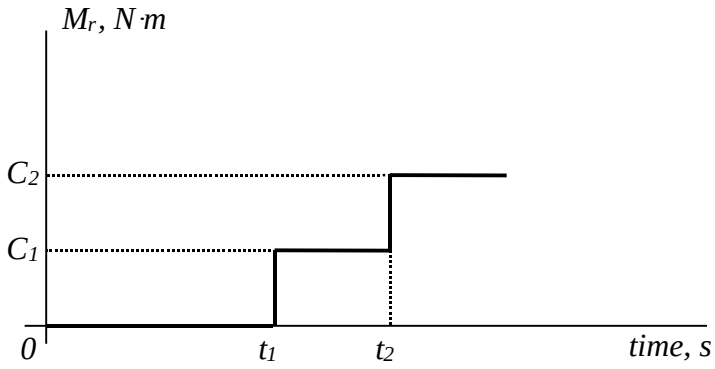


Figure 2.4. Resistance moment as a stepped time function

Table 2

Basic model parameters

Figure	Model	Comment	Value
M_a	driving_torque	Active electromagnetic moment	
M_c		Resistance moment	F(t)
J_o	Iy	Moment of inertia of the shaft relative to the axis of revolution	0,1 kg·m^2
t_1	t1	Time t_1, see Figure 2.4	5 s
t_2	t2	Time t_2, see Figure 2.4	8 s
C_1	C1	Constant resistance moment C_1, see Figure 2.4	5 N·m
C_2	C2	Constant resistance moment C_2, see Figure 2.4	10 N·m

2.2.2. Mechanical part

As it was mentioned above if the imported Matlab/Simulink model calculates output forces or moments, it is necessary to introduce the correspondent force elements on the stage of the model description in **UM Input**. The possibilities of the future connection between UM and Matlab/Simulink models should be provided in advance.

In the considered example we introduced a general force **Driving torque** into a UM-model, see Figure 2.5a. This *general force* has the only non-zero component. It is moment relative to the Y-axis, which is set by means of **Ma** parameter. We will connect this parameter with electro-magnetic moment that is produced in the Matlab/Simulink model of the DC motor.

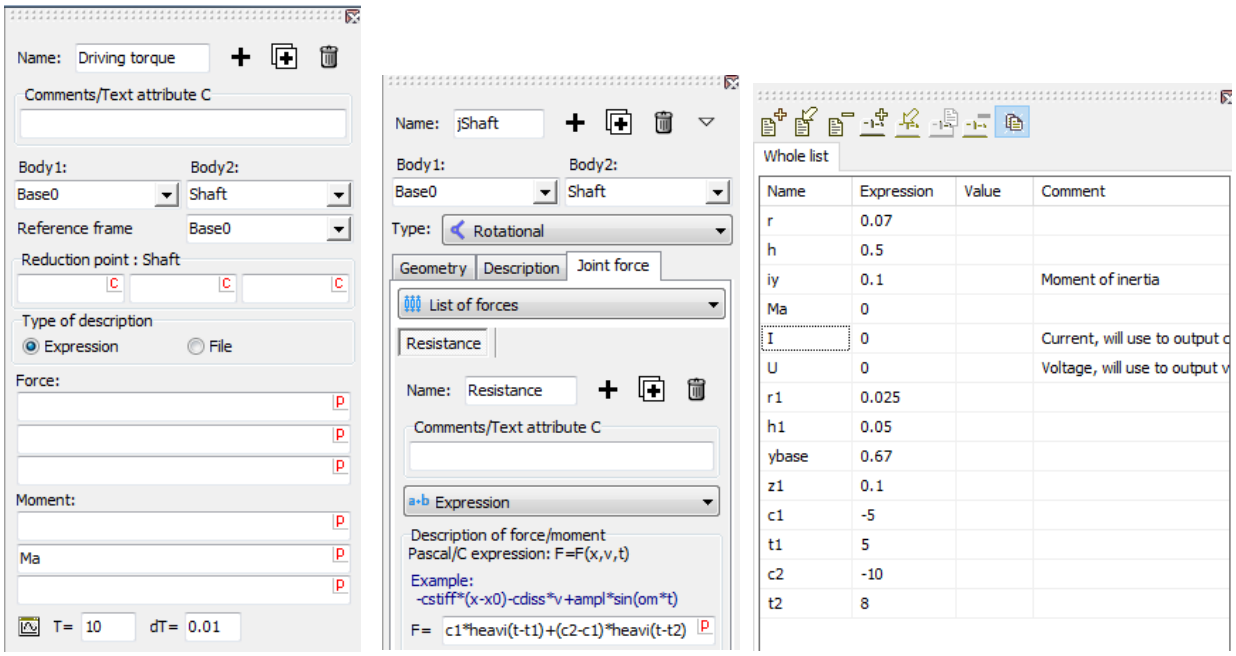
Resistance moment is described as follows:

Mr = c1**heavi*(t-t1)+(c2-c1)**heavi*(t-t2), where

t1, t2, c1 and c2 are parameters of the model, see Figure 2.4b. Note that

$$heavi(t) = \begin{cases} 1, t > 0 \\ 0, t \le 0 \end{cases}$$

In order to plot current and voltage in an anchor circuit two parameters are introduced in the model: I (current) and U (voltage), Figure 2.5c. During the simulation we will assign outputs of Matlab/Simulink model (second and third output) to these parameters. It gives us a possibility to plot time histories of these values.



a) b) c)

Figure 2.5. Active and resistance moments, list of parameters of the model

2.2.3. Electrical part

In this lesson we will use DC motor with separate excitation and constant magnetic flux. Reference data for the motor are given in Table 3.

Table 3

DC motor parameters

Parameter		Value
Power, kW		3,6
Voltage, V		220
Nominal rotational speed, r.p.m.		3000
Maximal rotational speed, r.p.m.		4000
Efficiency, %		79
Resistance at 15ºC, Ohm		
	anchor circuit	0,42
	additional poles	0,356
	excitation circuit	129
Inductance of anchor circuit, mHn		4,8

DC motor is turned on with the help of starting resistors. The first resistor of 2.5 Ohm is turned off in 1.2 s. The second resistor of 2.035 Ohm – in 2.5 s.

2.3. Restrictions

There are restrictions for connection Matlab/Simulink outputs to UM identifiers. You can only assign a Matlab/Simulink outputs to identifiers that influence on force elements: attachment/reduction points, stiffness and damping coefficients etc. All kinds of forces are supported.

According to UM basic principles assigning Matlab/Simulink outputs to UM identifiers that parameterize inertia parameters, graphical objects, and joint geometry are not allowed. Such assigning will lead to incorrect results of numerical simulation. You can use **Object simulation inspector** to change values of such parameter before each numerical experiment but not during the experiment.

3. Using Matlab Import tool

3.1. Workflow

Simulation of mechanical systems with imported Matlab/Simulink models supposes the following steps to be done.

- Creating the model of a control scheme in Matlab/Simulink.
- Exporting the created model from Matlab/Simulink as a Dynamic-Loaded Library (DLL).
- Creating the model of a mechanical system in Universal Mechanism (**UM Input** program).
- Loading created UM-model in the **UM Simulation**. Importing Matlab/Simulink DLL into an UM-model and setting connection between a mechanical part and a control scheme with the help of **Wizard of external libraries**.
- Simulating dynamics of the obtained model.

UM considers a Matlab/Simulink model as a black box, which somehow transforms input signals to output ones. Inputs of a Matlab/Simulink model are assigned with UM *variables* that could be obtained with the help of **Wizard of variables**. Outputs of a Matlab/Simulink model are assigned to UM parameters.

For implementation of control efforts active forces and torques are introduced in the UM-model. That forces and torques are given with the help of parameters, which Matlab/Simulink outputs are assigned to.

3.2. Inverted pendulum

A model of the inverted pendulum without the linked systems of control you can find in the {UM Data}\SAMPLES\TUTORIAL\inv_pend directory. Before the lesson starts check up its presence in the given directory. Creating UM-model of the inverted pendulum is not discussed in this manual. Only questions of import Matlab/Simulink model are considered. That's why please check if the **inv_pend** model is on your computer otherwise you can download it here: www.universalmechanism.com/download/90/inv_pend.zip. In this lesson we will not analyze in detail all the stages of the mechanical model creation, we will only describe connection of the mechanical part with the control system of Matlab/Simulink.

A model of the inverted pendulum with the imported Matlab/Simulink control scheme is in the {UM Data}\SAMPLES\simulink\inv_pend_ctrl directory.

3.2.1. Export from Matlab/Simulink

Export control scheme from Matlab/Simulink means compiling of the prepared model into dynamic-loaded library, which is then connected to Universal Mechanism.

Note.	It is supposed that you have already been accustomed to the basics of Matlab/Simulink simulation. In any case you may omit this paragraph and come back here later. The **inv_pend\pendpid_ctrl.dll** is the outcome of export from Matlab/Simulink. You may omit the stage of DLL creation and use the one already prepared.
Note.	Block **Derivative** is not supported in exported models. Use **Transfer Fnc** block instead of **Derivative** one.

Export a Matlab/Simulink model consists of the following steps.

1. Preparing the control scheme for export procedure.
2. Copying files, which are necessary for compiling, to the directory of the model.
3. Setting compiling options and compiling DLL.

Preparing a control scheme

Firstly, it is necessary to exclude from the Matlab/Simulink model all the components that cannot be compiled: all input/output components, all components without source code etc.

Besides it is necessary to include “IN” and “OUT” components into the model. These components are used for future connection of the Matlab/Simulink model with UM-model. In the model of the inverted pendulum we have the only input (pendulum angle from vertical) and output (force that acts on the cart).

The Matlab/Simulink model of the control for the inverted pendulum is shown in the Figure 3.1.

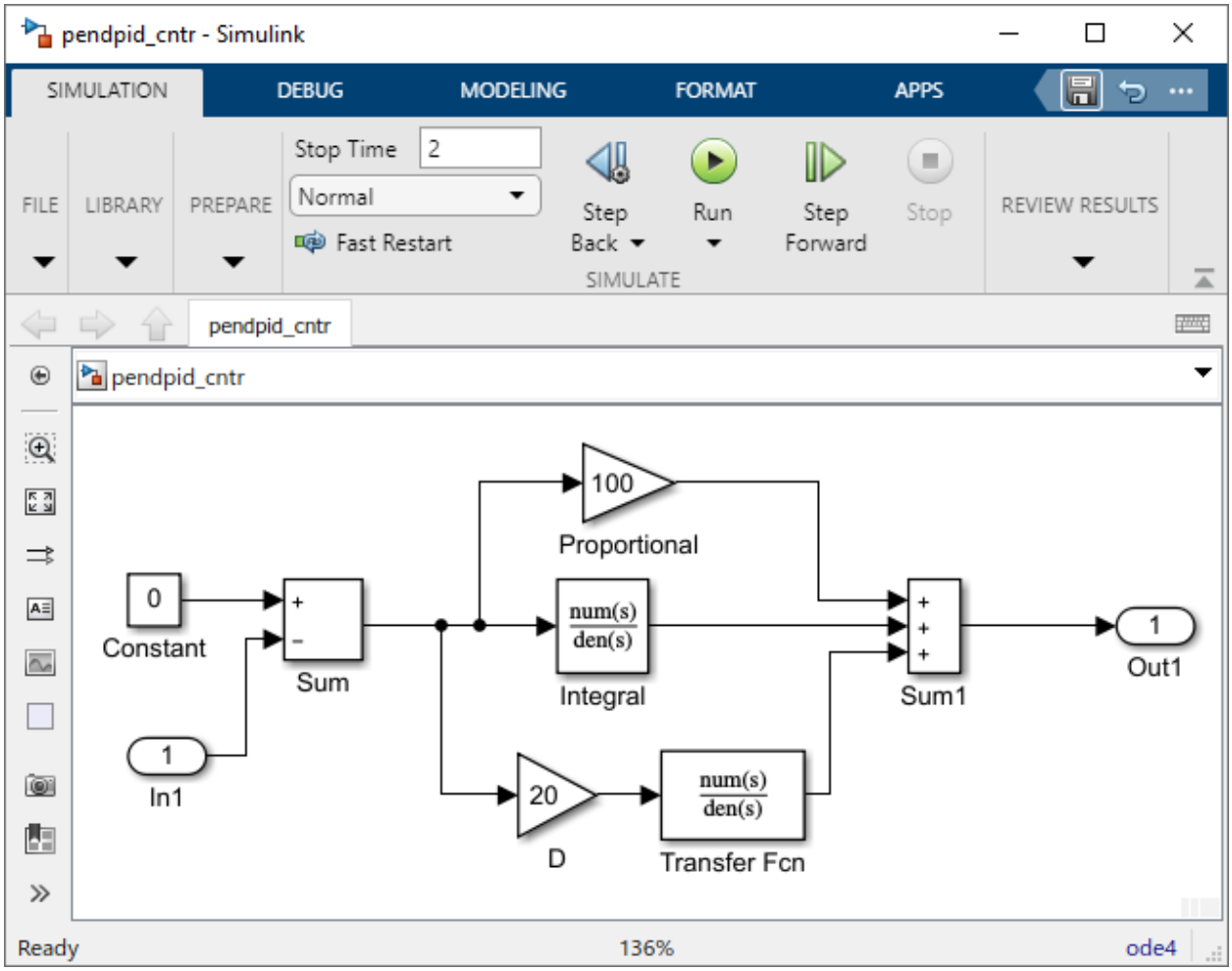


Figure 3.1. Matlab/Simulink model of the control scheme for inverted pendulum

Copying necessary files and setting current directory

1. For the successful compilation it is necessary to copy several auxiliary files from the **{UM Data}\simulink** to the model's directory (to the same directory where **pend_pid.mdl** is situated). Your following actions depend on the version of Matlab that you use.

Matlab 8.X / R2013-R2015 32-bit

Copy **rsim.tlc, rsim_vc.tmf**, and **um.tls** files from the **{UM Data}\Simulink\R2013_2015\x32** directory to the model's directory.

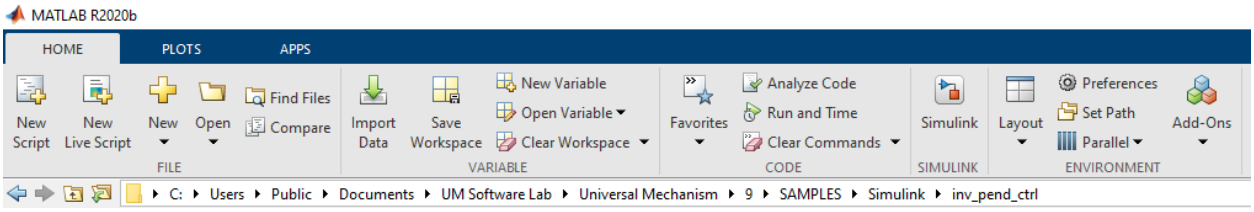
Matlab 8.X / R2013-R2015 64-bit

Copy **rsim.tlc, rsim_vc.tmf**, and **um.tls** files from the **{UM Data}\Simulink\R2013_2015\x64** directory to the model's directory.

Matlab 9.X / R2016-R2020 64-bit

Copy all files (**rsim.tlc, rsim_vc.tmf** and **um.tls**) from the **{UM Data}\Simulink\R2016_R2020x64** directory to the model's folder.

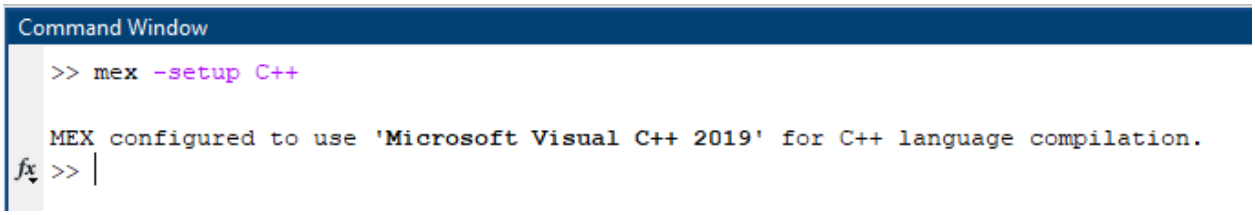
2. Point to Matlab main window. Make sure that the field **Current Directory** (**Current Folder**) under Matlab/Simulink environment set to the model's directory.



Code generation and building the DLL

We need to set up compiling options first and then build the output DLL.

1. Point to the Matlab **Command Window**. Run the **'mex –setup'** (or **mex -setup C++** in the recent Matlab releases) command in the Matlab/Simulink command window and select one of the supported compilers from **Microsoft Visual C/C++** family that is supported by your Matlab version.



Note. It is strongly recommended to use **Microsoft Visual C/C++** to export your model from Matlab/Simulink. Please note that every particular Matlab version supports the particular list of compilers. Please use the following table to check supported compilers for every Matlab release: https://www.mathworks.com/support/sysreq/previous_releases.html.

Your following actions depend on the version of Matlab that you use.

Code generation and building the DLL in Matlab 8.X / R2013-R2015

1. Point to the model's window. From the **Code** menu select **C/C++ Code** and point to **Code Generation Options**. New dialog window appears.
2. Set **System target** file to **rsim.tlc**.
3. Set **Language** to **C.**
4. Make sure that **Make command** is set to **make_rtw**.
5. Set **Template makefile** to **rsim_vc.tmf**.
6. Set **Solver options / Type** to **Fixed Step**.
7. Select an appropriate numerical method in the **Solver options / Solver**, for example **ode4 (Runge-Kutta)**.
8. To compile the DLL come back to the **Code Generation** and click **Build**.

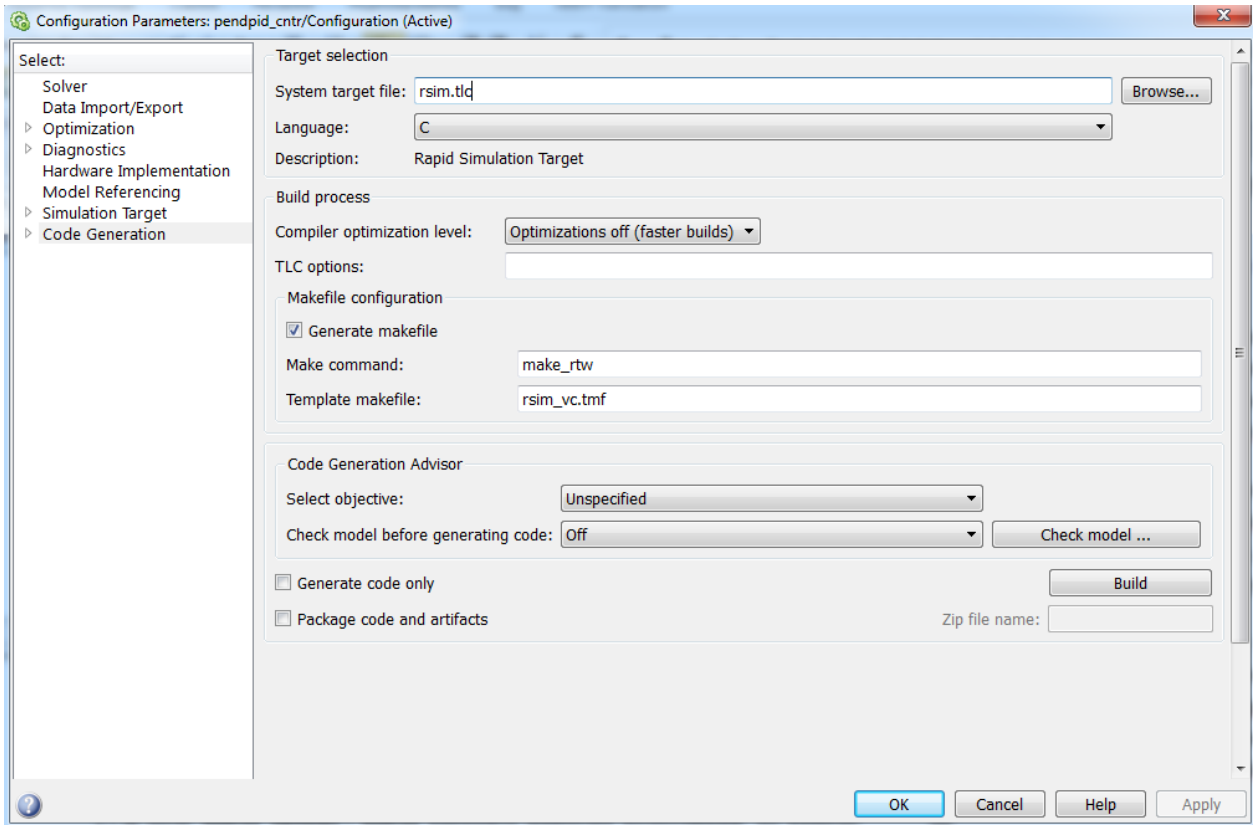


Figure 3.2. Compiling options in Matlab 8.X R2011-R2015

During compilation process the command window of Matlab/Simulink shows messages about the process. Successful compiling ends with the message "**Creating LIBRARY** ..\pendpid_cntr.lib **and object** ..\pendpid_cntr.exp".

Note. To use dynamic-loaded library (DLL) generated with the help of Matlab R2011-R2015 in UM it is necessary that the correspondent Matlab release was installed on the same computer.

As a result of a successful compilation the **pendpid_cntr.dll** file is created.

Code generation and building the DLL in Matlab 9.X / R2016-R2020

1. Point to the [**pendpid_ctrl**] model's window, see Figure 3.2.
2. In the **MODELING** gallery select **Model Settings / Model Settings** (see Figure 3.2) or click the right mouse button on the white background of the window and from the context popup menu select the **Model Configuration Parameters** menu item. **Configuration Parameters** window appears, see Figure 3.4.

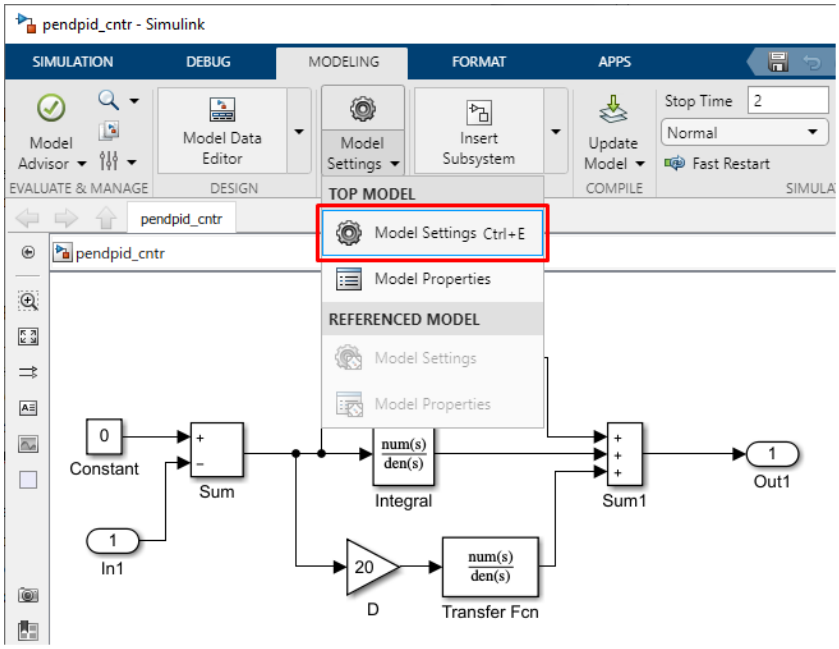


Figure 3.3. Model settings in Matlab 9.X (R2016-R2020)

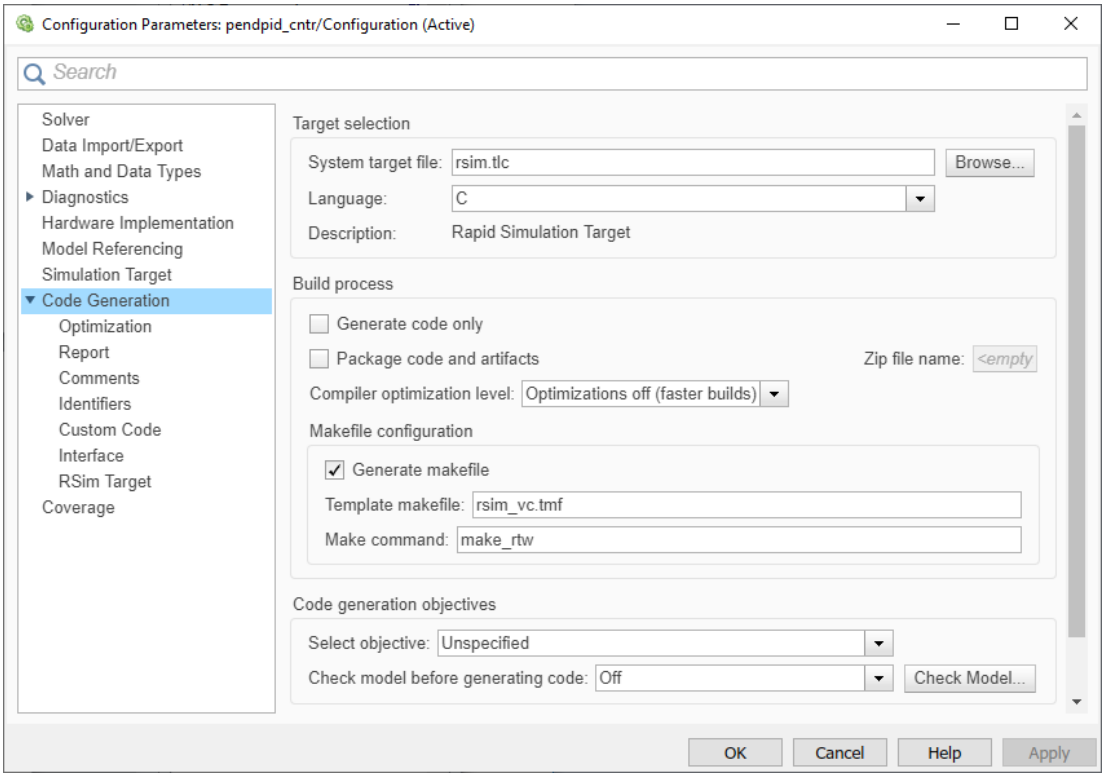


Figure 3.4. Code Generation settings in Matlab 9.X (R2016-R2020)

3. In the list in the left part of the window select the **Code Generation** item.
4. In the **System target file** type **rsim.tlc**.
5. Set **Language** to **C**.
6. In the **Template makefile** field set **rsim_vc.tmf**.
7. Make sure that **Make command** is set to **make_rtw**.
8. In the list in the left part of the window select the **Solver** item.
9. Set **Solver options | Type** to **Fixed Step**.
10. In the **Solver options | Solver** select the suitable numerical method, for example, **ode4 (Runge-Kutta)**.
11. Then select **Code Generation / Optimization** and set **Default parameter behavior** to **Tunable**, see Figure 3.5. It will give a possibility to change Simulink model parameters on the UM side. If you will not do this the list of Simulink model parameters in UM will be empty.
12. Close the **Configuration Parameters** window.

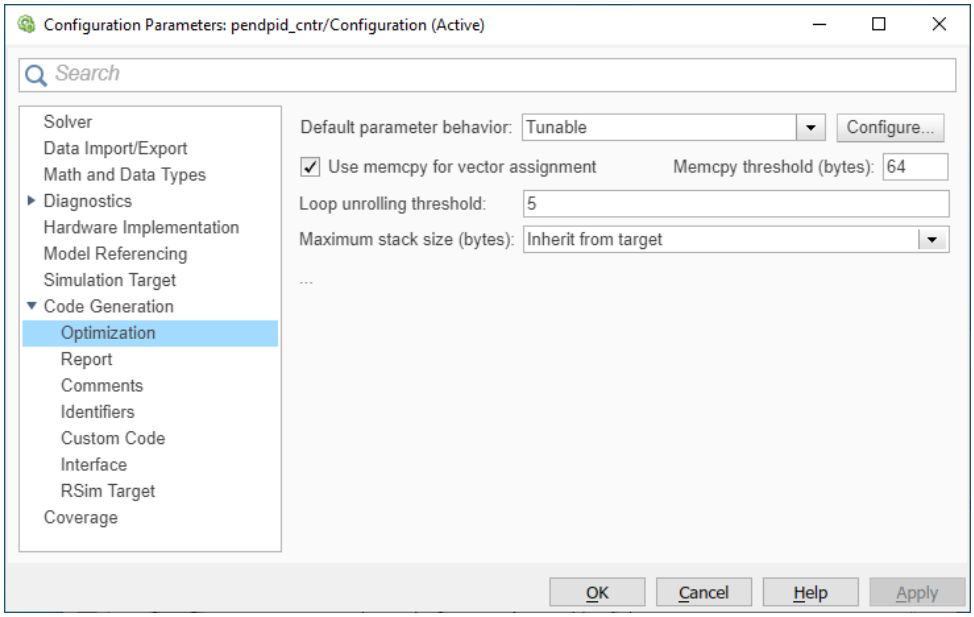


Figure 3.5. Tunable Parameters in Matlab 9.X (R2016-R2020)

13. Finally in the **APPS** gallery select the **Simulink Coder** and the click the **Generate code and build model** button, see Figure 3.10.

During compilation process the **Diagnostic Viewer** window of Matlab/Simulink shows messages about the process. Successful compiling ends with the message "*Build process completed successfully*»", see Figure 3.7.

Note. To use dynamic-loaded library (DLL) generated with the help of Matlab R2016-R2020 in UM it is necessary that the correspondent Matlab release was installed on the same computer.

As a result of a successful compilation the **pendpid_cntr.dll** file is created. Source code will be placed in the newly created *pendpid_cntr_rsim_rtw* directory.

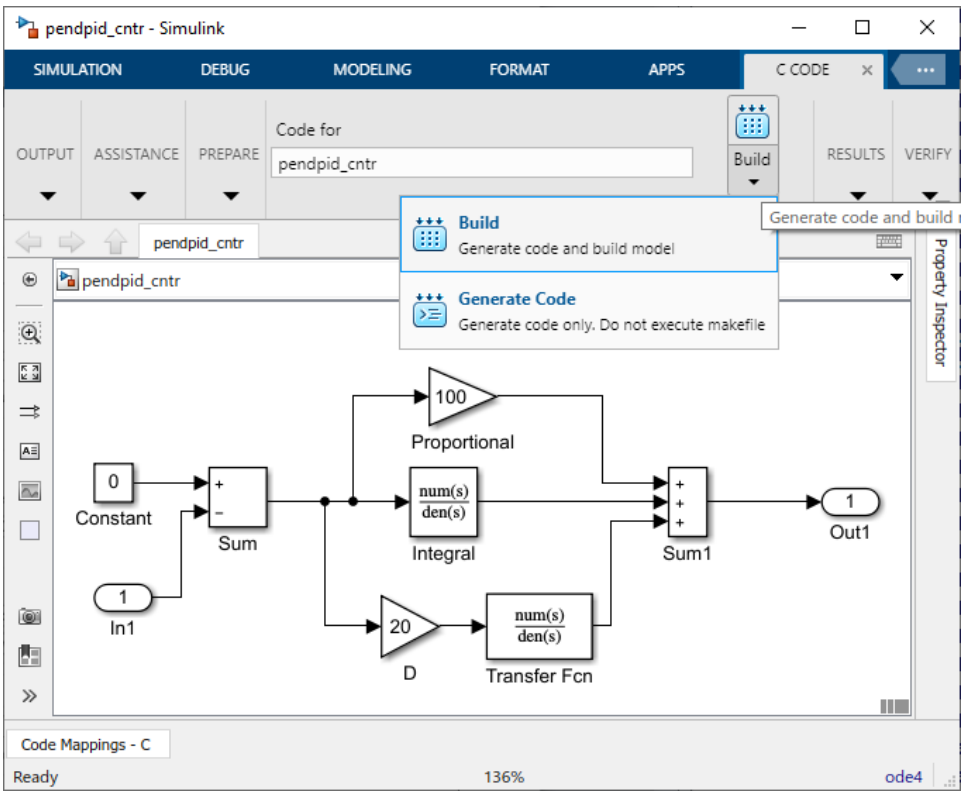


Figure 3.6. Generate code and build the model in Matlab 9.X (R2016-R2020)

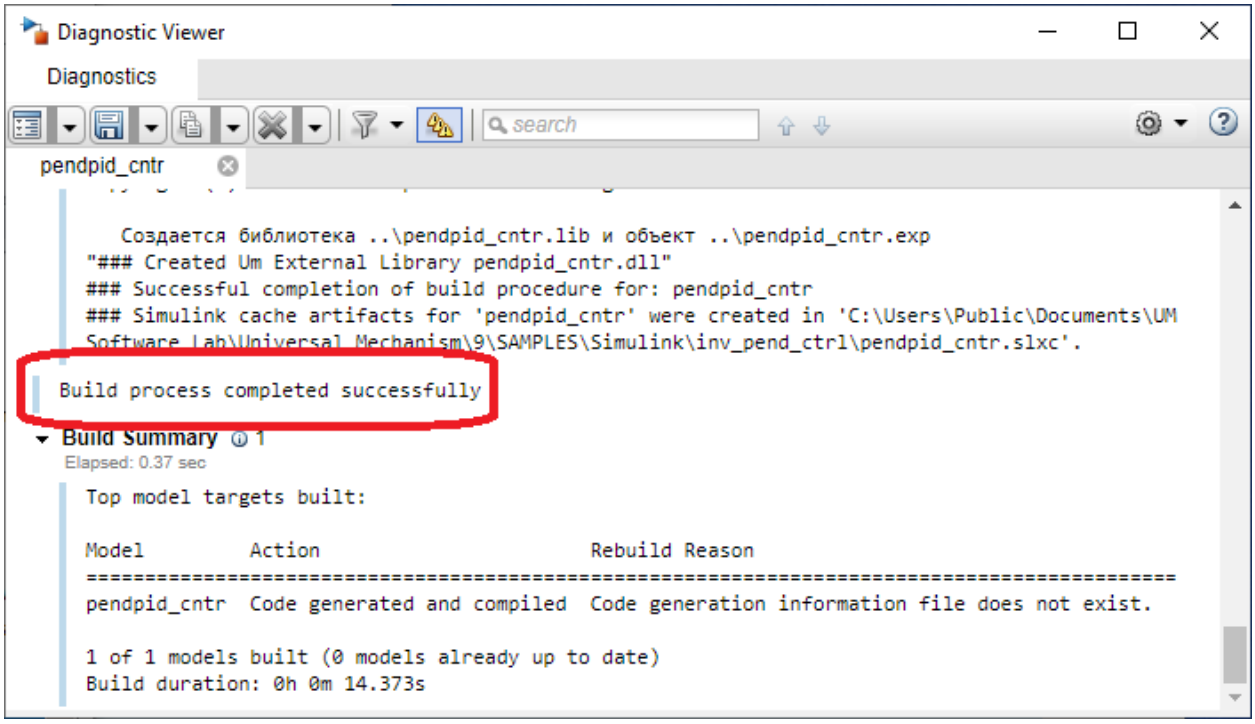


Figure 3.7. Diagnostic Viewer in Matlab 9.X (R2016-R2020)

3.2.2. Import Matlab/Simulink library in UM

Loading Matlab/Simulink library

1. Run the **UM Simulation** program.
2. Load the {UM Data}\SAMPLES\TUTORIAL\inv_pend model.
3. Create new animation window.
4. From Tools menu select **External library interface…**.
   Window of **Wizard of external libraries** appears.

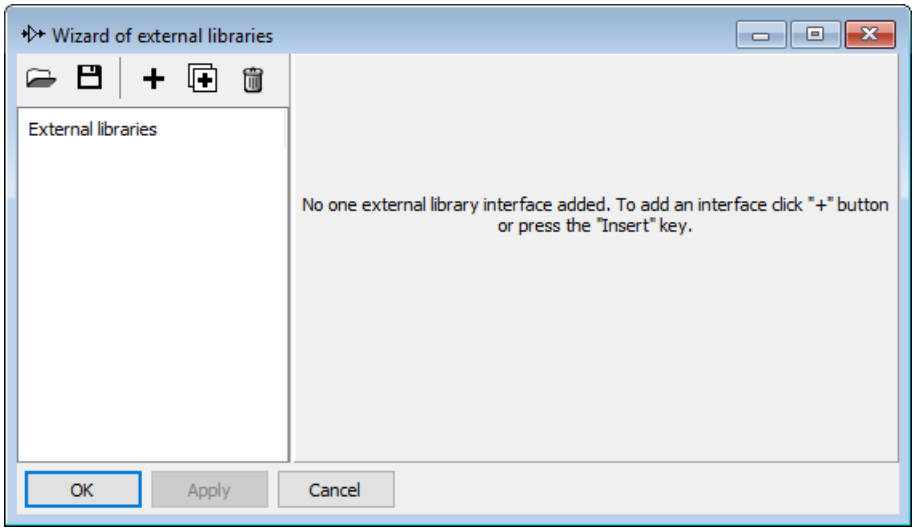


Figure 3.8. Wizard of external libraries

5. Click the + button to load new Matlab/Simulink library.
6. In the **Path to external library** box select the ..\inv_pend\pendpid_cntr file, see Figure 3.9. If you have created DLL then choose it or choose DLL from the **inv_pend** directory.
7. Turn on the check box against **Interface 0** element in **External library** list in the left, see Figure 3.9.

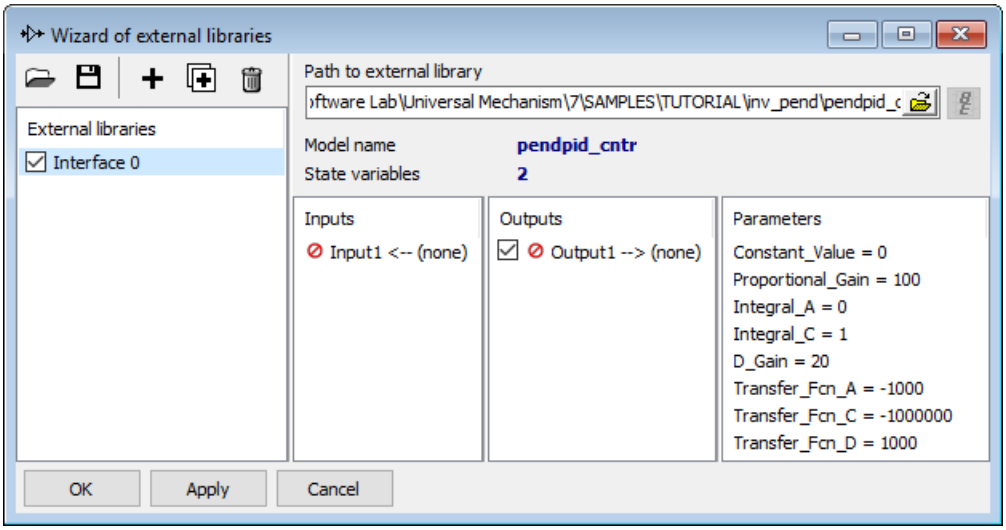
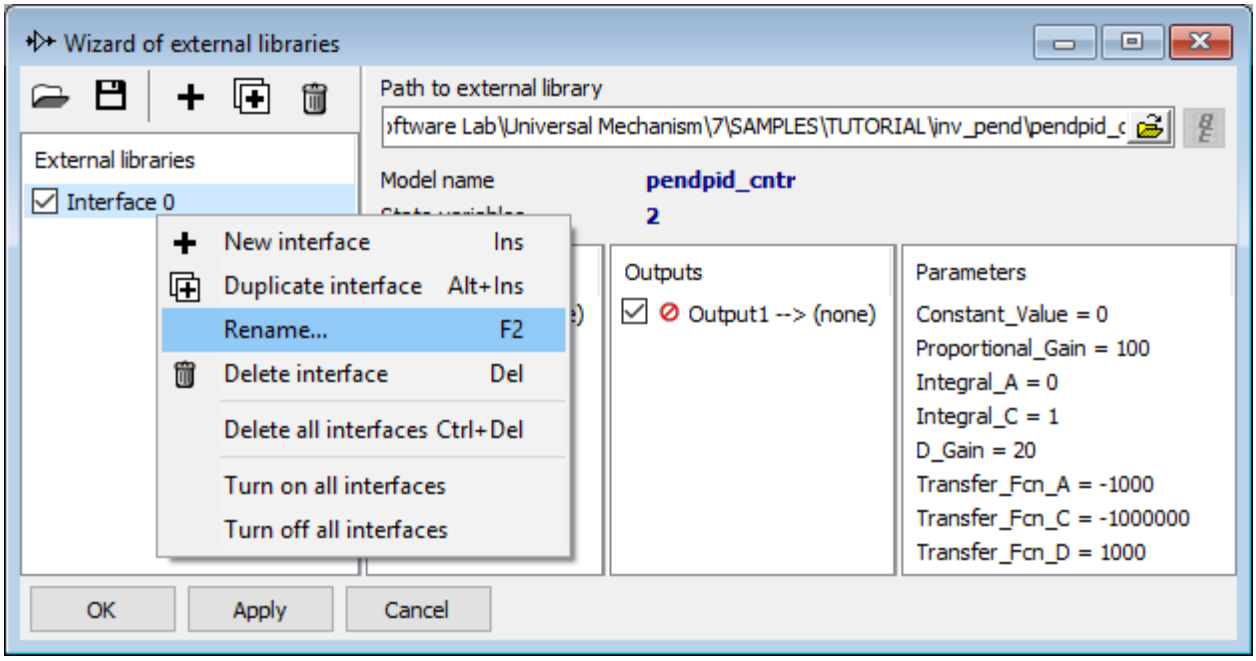


Figure 3.9. Add new interface

Wizard loads the selected model and determines the number of input and output components and parameters of the Matlab/Simulink model. Our example includes one input and one output components: pendulum angle from vertical and control force correspondently.

Rename interface

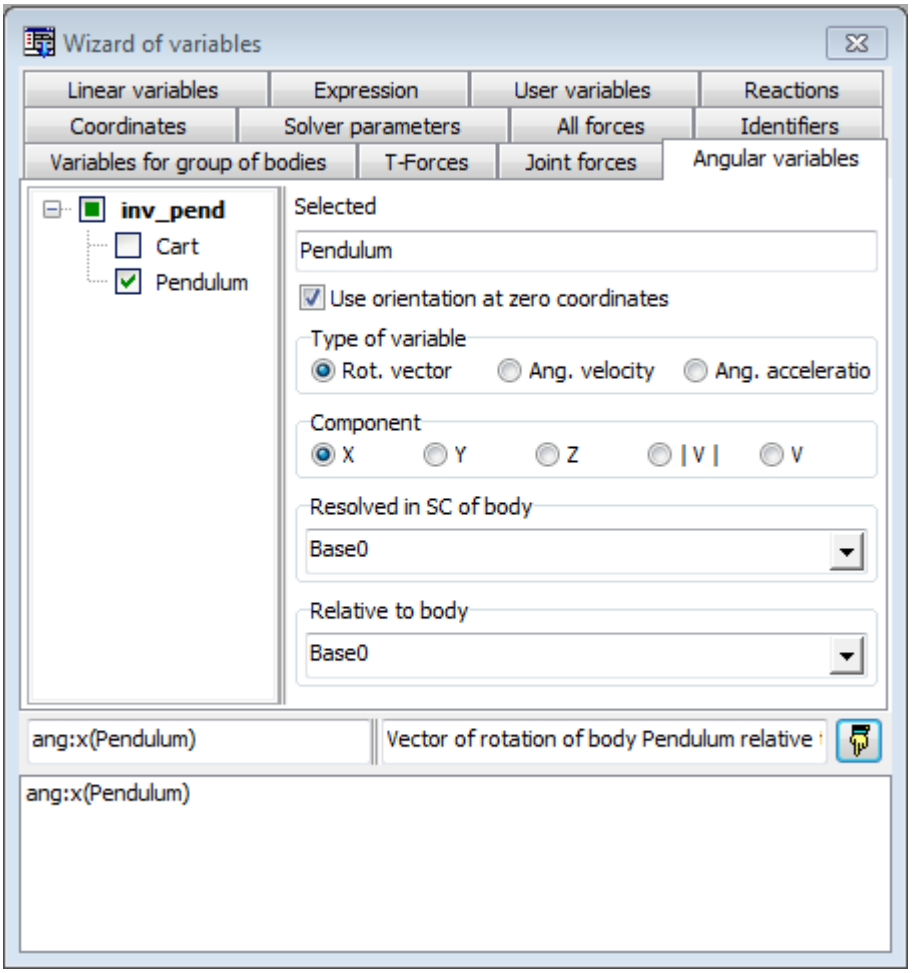
1. In the **External library** list select the only item **Interface 0**, open context menu and point to the **Rename…** menu item.
2. Replace **Interface 0** with **Control force** and press the **Enter** key.



Assign input components of the Matlab/Simulink model

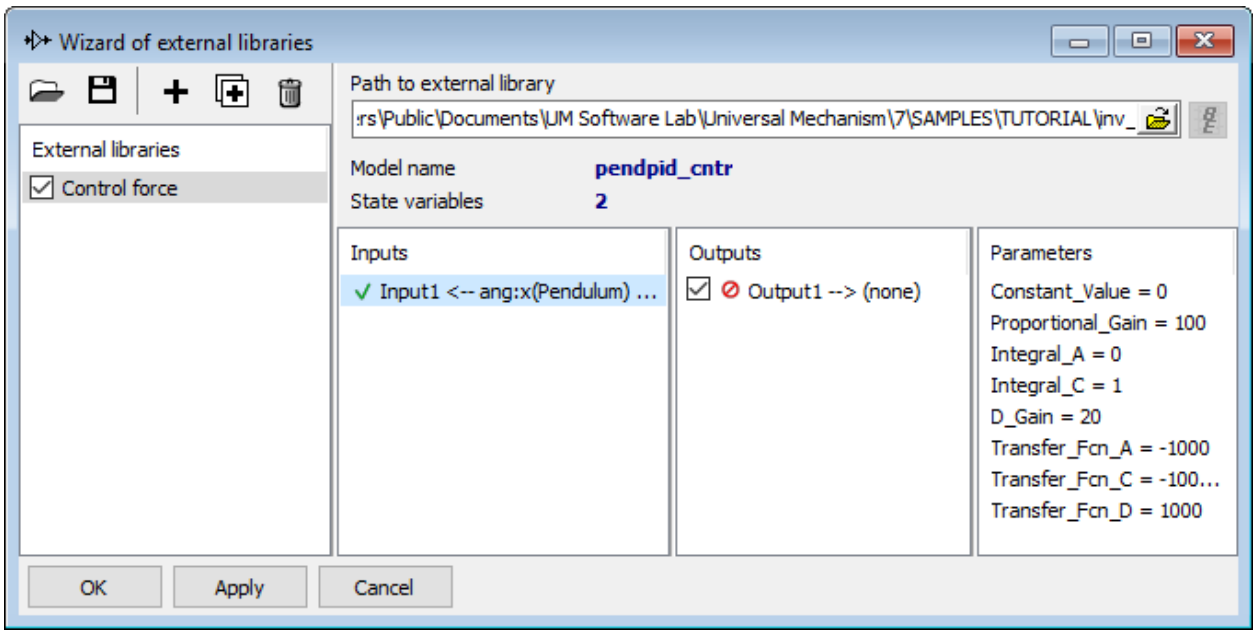
Let's create the '*pendulum angle from vertical' variable* with the help of the **Wizard of variables** and then assign that variable to the Matlab/Simulink model input.

1. Open **Wizard of variables**.
2. Select the **Angular variables** tab.
3. Turn on **Use orientation at zero coordinates**.
4. Select **Pendulum** in the list of bodies in the left, set **Type of variable** to **Rot. vector**, set **Component** to **X**.
5. Create the new variable with the help of the  button. The new variable appears in the container of variables.



6. Drag the **ang:x(Pendulum)** from the **Wizard of variables** to **Matlab/Simulink interface wizard** to the **Input1**.

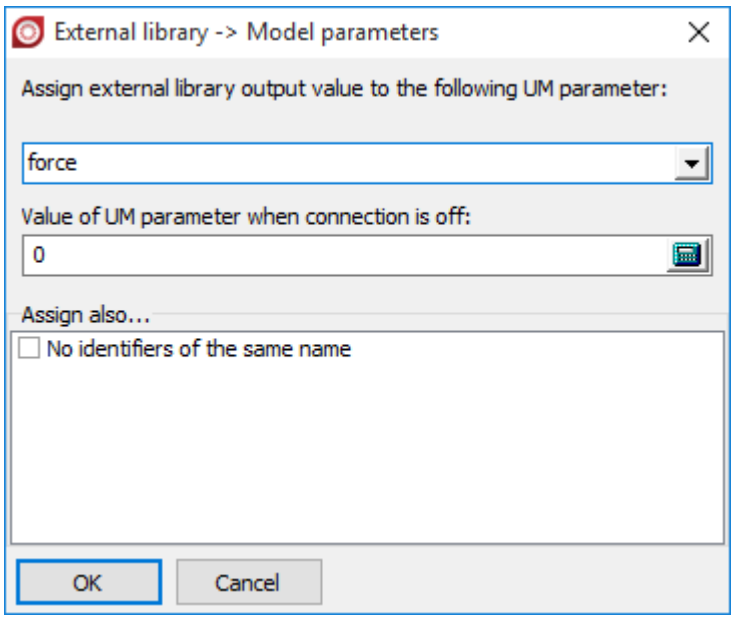
The **Input1** is now selected with the green mark, what means that this input is already assigned with a variable.


Wizard of external libraries
Path to external library
:rs\Public\Documents\UM Software Lab\Universal Mechanism\7\SAMPLES\TUTORIAL\inv_
External libraries
Control force
Model name
pendpid_cntr
State variables
2
Inputs
Input1 <-- ang:x(Pendulum) ...
Outputs
Output1 --> (none)
Parameters
Constant_Value = 0
Proportional_Gain = 100
Integral_A = 0
Integral_C = 1
D_Gain = 20
Transfer_Fcn_A = -1000
Transfer_Fcn_C = -100...
Transfer_Fcn_D = 1000
OK
Apply
Cancel

Assign output components of the Matlab/Simulink model

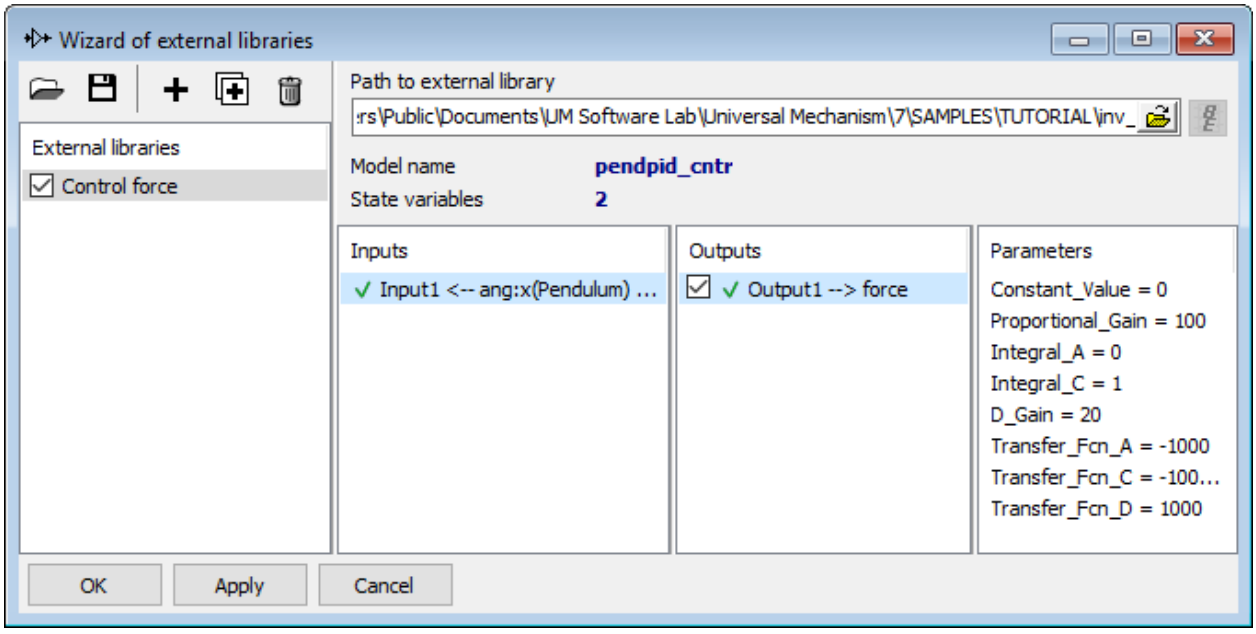
Output components are assigned to UM-model parameters. Such parameters are usually used for force description.

1. Double click on the **Output1** item in the **Output** list. New dialog window for assigning Matlab/Simulink output component and UM parameters appears.
2. Select **force** in the field **UM parameter**.
3. Click **OK**.



Now the Matlab/Simulink interface is completely defined.

4. Click **Apply** for updating data without closing the window or **OK** for updating data and closing the window.

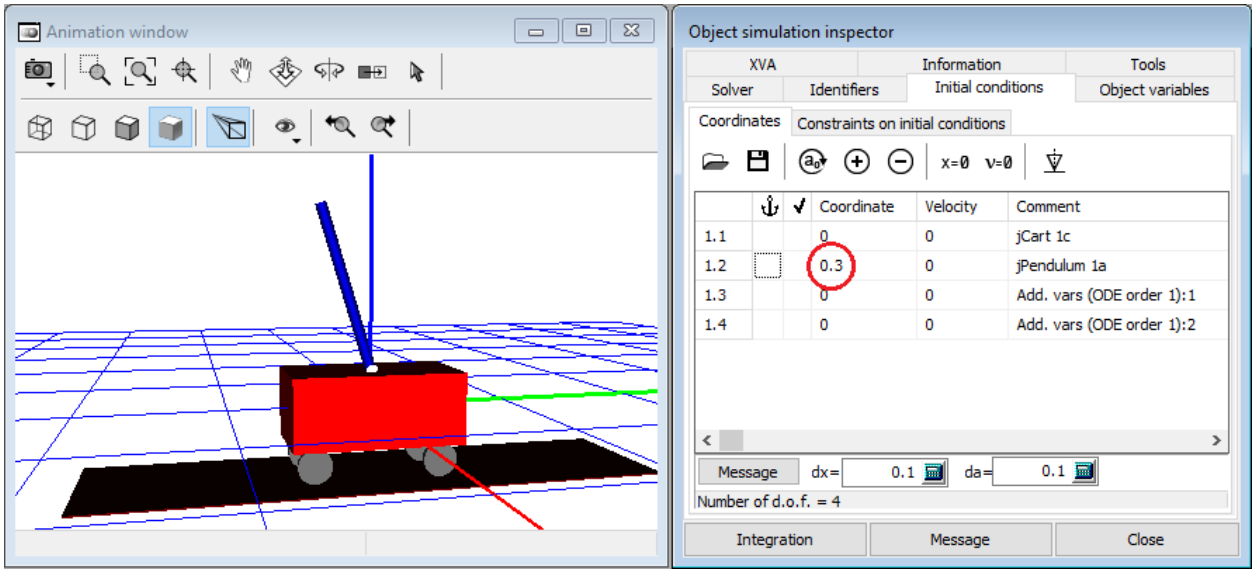


3.2.3. Simulation of motion

The model of the inverted pendulum now includes mechanical part and imported control scheme. Now we deflect the pendulum and start simulation to check if the control scheme really works.

1. From the **Analysis** menu select **Simulation…. Object simulation inspector** appears.
2. Point to the **Initial conditions** tab.
3. Let's deflect the pendulum with 0.3 radian. Set **Coordinate/1.2** to **0.3.**

Note. The first (1.1) coordinate is the longitudinal coordinate of the cart. The third (1.3) and fourth (1.4) coordinates are state variables of the control scheme.



Set the parameters of the simulation process and start simulation.

4. Select the Solver tab and set **Solver** to **Park**, set **Simulation time** to **0.1**, **Step size for animation** to **0.0002** and **Error tolerance** to **1E-7**.
5. Click the **Integration** button.

In the animation window you can see that the inverted pendulum is really stabilized by the control scheme.

Visualizing the control force and plots of variables

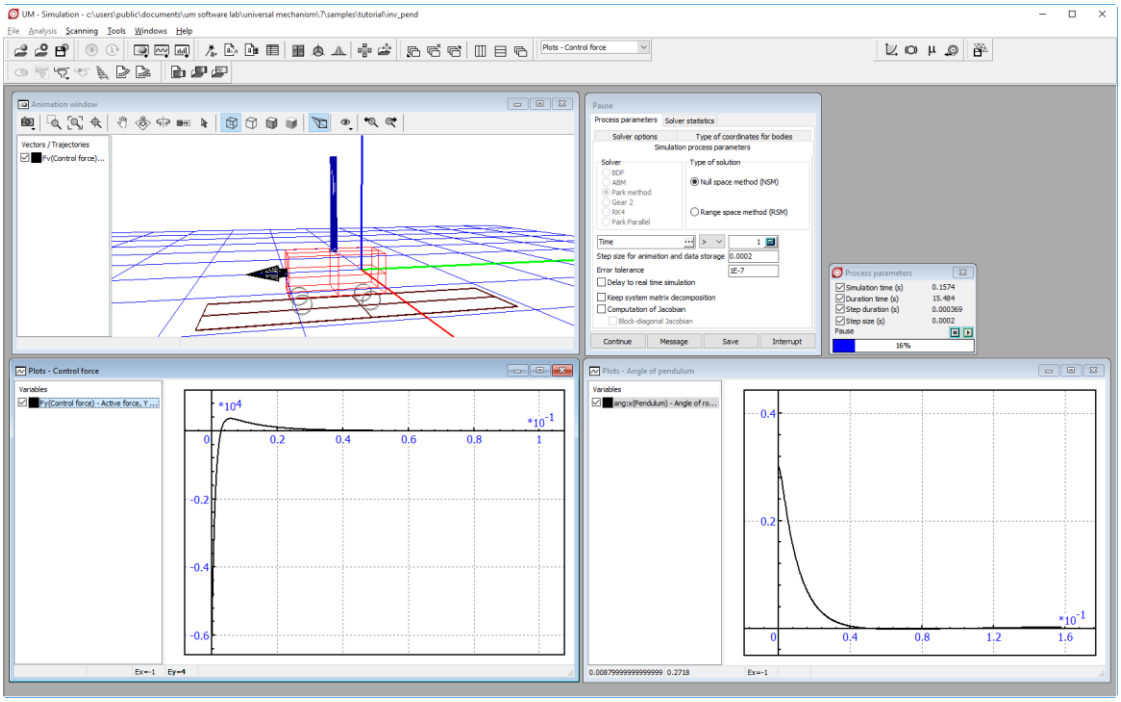
Now we show the control force in the animation window and plot oscillograms of the control force and pendulum angle from vertical.

Create the new variable – the vector of the control force to animate in the animation window.

1. Open the **Wizard of variables**.
2. Select the **T-Forces** tab, set **Type of variable** to **Force**, **Component** to **V** (vector), and **Acts on body to Cart**.
3. Click the  button and create new variable to the container of variables and then drag it to the animation window.
4. Click the  button to switch animation window to wire mode. It gives us a possibility to observe the vector even inside the cart.

Now create two variables for graphical windows: value of the control force and pendulum angle.

5. Open new graphical window for the plot of the control force.
6. Point to **Wizard of variables/T-forces** again, set **Component** to **Y**. Create the variable and drag it to the graphical window.
7. Open one more graphical window. This window is intended for a plot of the pendulum angle.
8. Select **Wizard of variables/Angular var**. (angular variables).
9. Select **Pendulum** in the list of bodies, set **Type of variable** to **Rot. vector** and **Component** to **X**.
10. Create the variable and drag it to the second graphical window.
11. Close the **Wizard of variables**.
12. Start integration.



3.3. DC motor

A simple electromechanical system is considered in this section. Mechanical part of the system consists of a shaft under the influence of active moment Ma and resistance moment M_r, Fig. 3.5. A DC motor that produces the active electromagnetic moment Ma presents electrical part. The resistance moment M_r is supposed to be a time function. A model of a mechanical part of the system without DC motor is situated in the {UM Data}\SAMPLES\TUTORIAL\dc_motor directory. Please, check, if this model is in that directory, otherwise, please, download the model using the link www.universalmechanism.com/download/90/dc_motor.zip

Mechanical part is created with the help of Universal Mechanism software, DC model is created using Matlab/Simulink software.

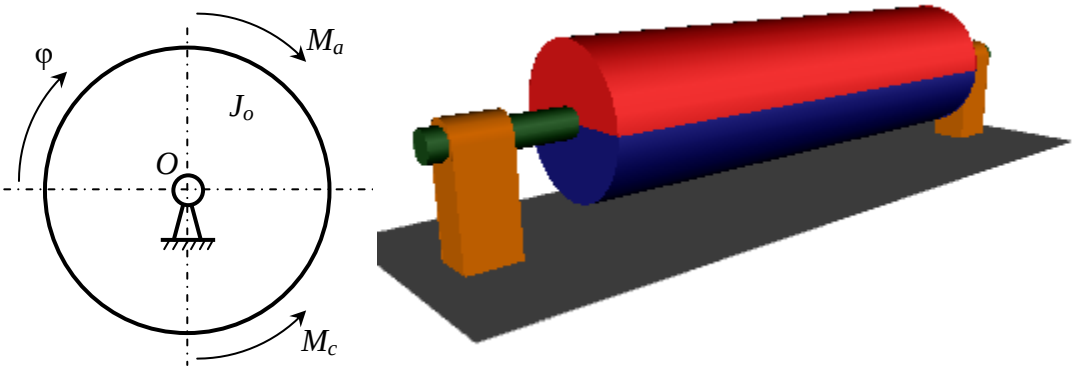


Figure 3.10. Scheme of the model (left) and the mechanical part (right)

Here we will not discuss in details how to create models of mechanical and electrical parts. They are quite simple and it is supposed that the reader can open and study the model of the mechanical part in Universal Mechanism and model of the DC motor in Matlab/Simulink. We will show how to join mechanical and electrical part into one electromechanical system.

The final, ready for simulation, model of the electromechanical model is situated in the **{UM Data}\samples\Simulink\dc_motor_fin**[1] directory.

[1] www.universalmechanism.com/download/90/dc_motor_fin.zip

3.3.1. Matlab/Simulink model of DC motor

Matlab/Simulink model of a DC motor has one input – angular velocity of the rotor and three outputs: (1) electromagnetic moment, (2) current and (3) voltage (V) in the anchor circuit. Screen shots of the DC motor are given in Figure 3.11 and Figure 3.12.

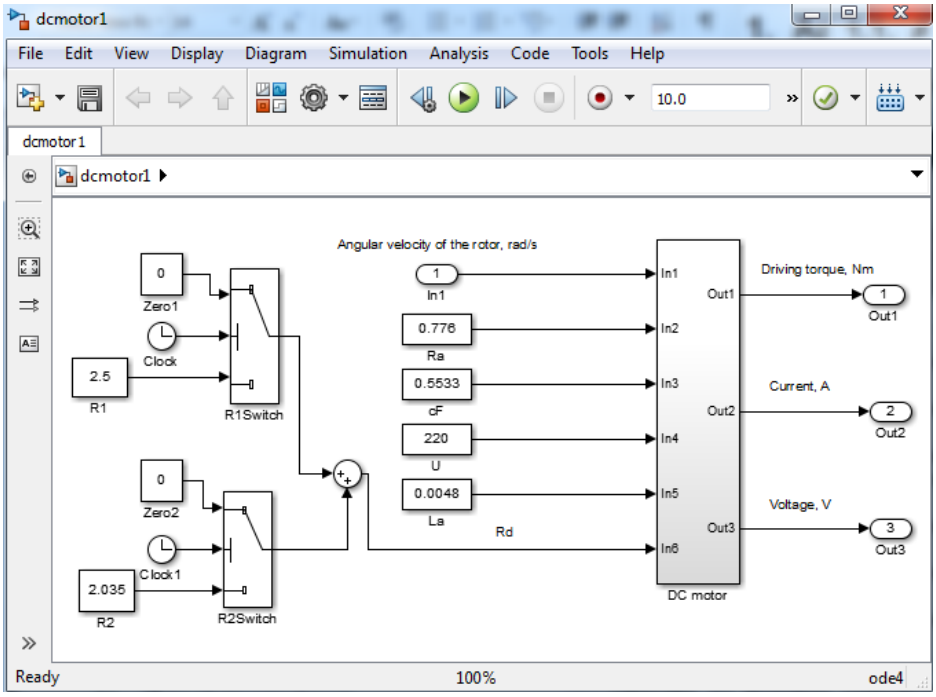


Figure 3.11. DC motor in Matlab/Simulink

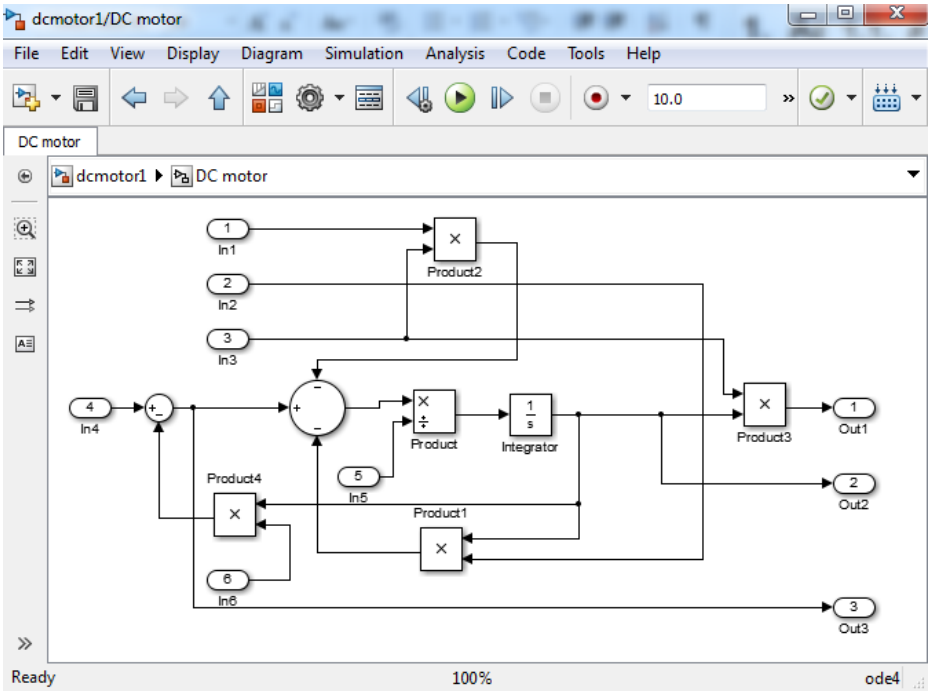


Figure 3.12. DC motor subsystem

3.3.2. Export from Matlab/Simulink

Now we need to compile the source *.mdl file that contains the model of the DC motor with separate excitation and have a *.dll file as a result.

Matlab/Simulink model of the DC motor with separate excitation is in the ../dc_motor/dcmotor1 file. You can compile this *.mdl file according the instructions from the Sect. 3.2.1. Also you can omit the compilation process and use already **prepared dcmotor1.dll.**

3.3.3. Import Matlab/Simulink library in UM

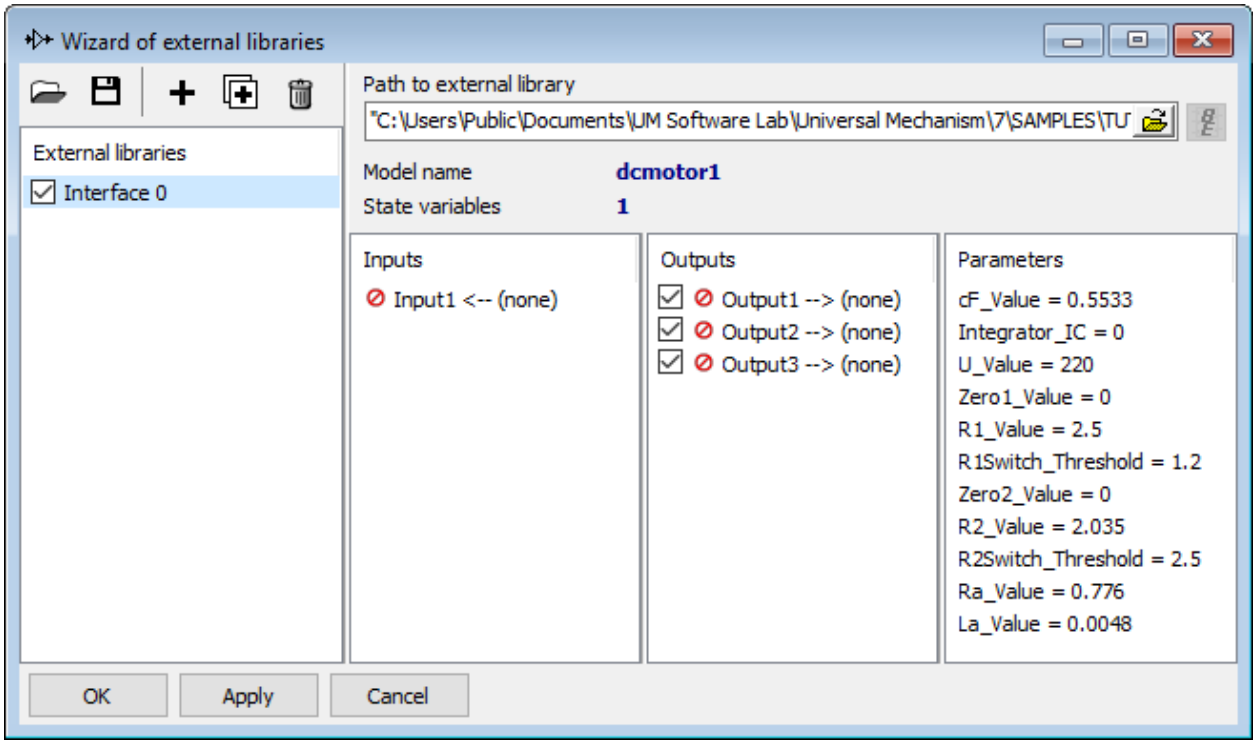
Loading the model

1. Run **UM Simulation** program.
2. Load the {UM Data}\SAMPLES\TUTORIAL\dc_motor model.
3. Create new animation window.

Loading Matlab/Simulink library

1. From **Tools** menu select **External library interface**…. **Wizard of external libraries** appears.
2. To add new Matlab/Simulink library click the **+** button.
3. In the **External library** box select the **dcmotor1.dll**.
4. Turn on **Interface 0**.

Wizard loads the selected model and determines the number of input and output components and parameters of the Matlab/Simulink model. Our example includes one input and three output components. We will set angular velocity of the rotor as an input, assign the **Ma** parameter for the first output, and I and U parameters for the second and the third output values.



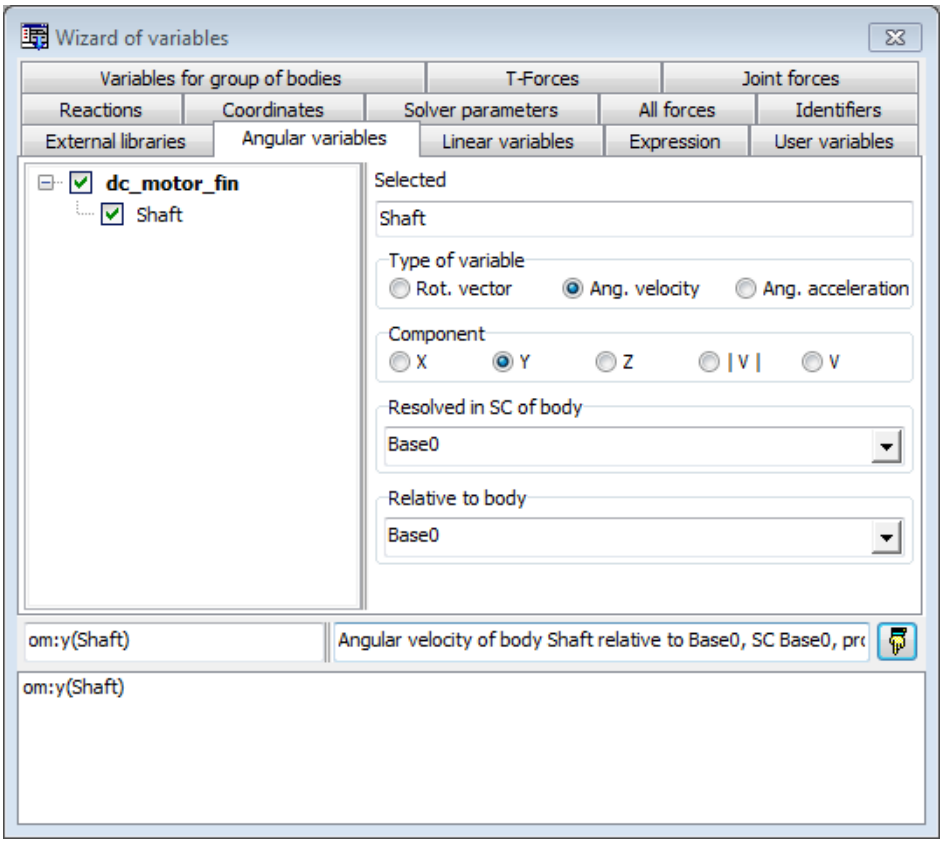
Rename interface

1. In the **External library** list select the item **Interface 0**, open context menu and point to the **Rename…** menu item.
2. Replace **Interface 0** with **DC motor** and press the **Enter** key.

Assign input components of the Matlab/Simulink model

Let's create the *'angular velocity of the shaft' variable* with the help of the **Wizard of variables** and then assign that variable to the Matlab/Simulink model input.

1. Open **Wizard of variables (Tools/Wizard of variables)**.
2. Select the **Angular variables** tab.
3. Select **Shaft** in the list of bodies in the left, set **Type of variable** to **Ang. velocity**, set **Component** to **Y**.
4. Create the new variable with the help of the  button. The new variable **om:y(Shaft)** appears in the container of variables.



5. Drag the **om:y(Shaft)** from the **Wizard of variables** to **Matlab/Simulink interface wizard** to the **Input1**.

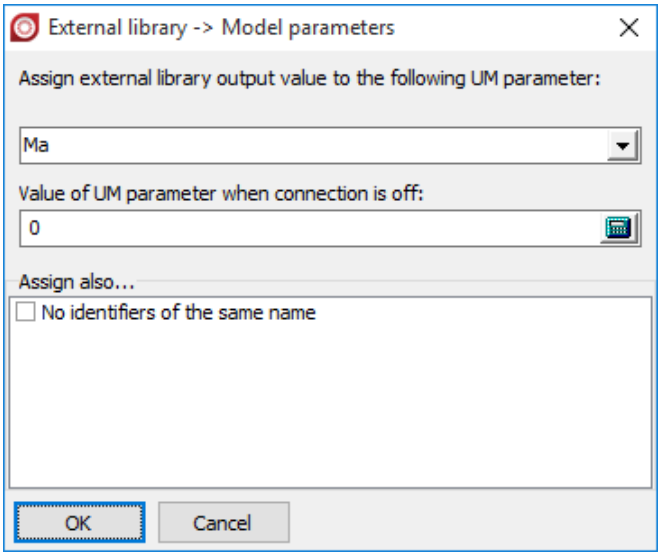
The **Input1** is now selected with green mark, what means that this input is already assigned with a variable.

6. Close the **Wizard of variables**.

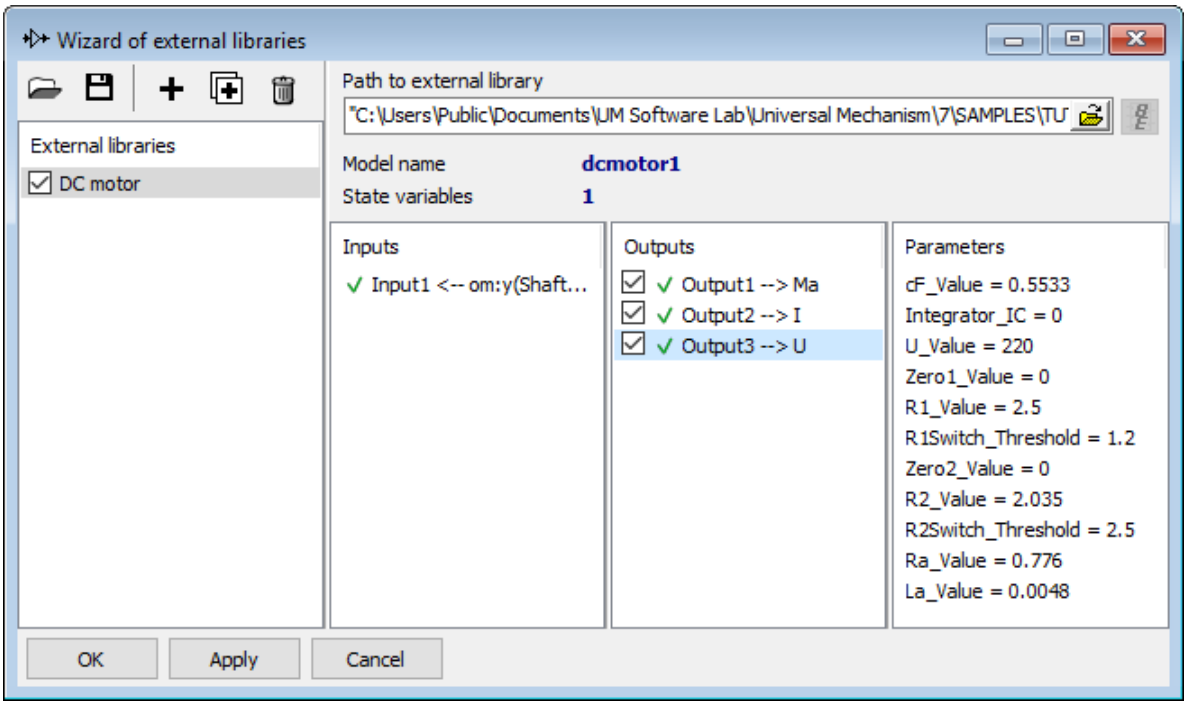
Assign output components of the Matlab/Simulink model

Output components are assigned to UM-model parameters. Such parameters are usually used for force description.

1. Double click on the **Output1** item in the **Output** list. New dialog window for assigning Matlab/Simulink output component and UM parameters appears.
2. Select **Ma** in the field **UM parameter**.
3. Click **OK**.



4. Set **Output2** for **I** parameter.
5. Set **Output3** for **U** parameter.
6. Click **OK** to apply al changes and close the window of the **Wizard of external libraries**.

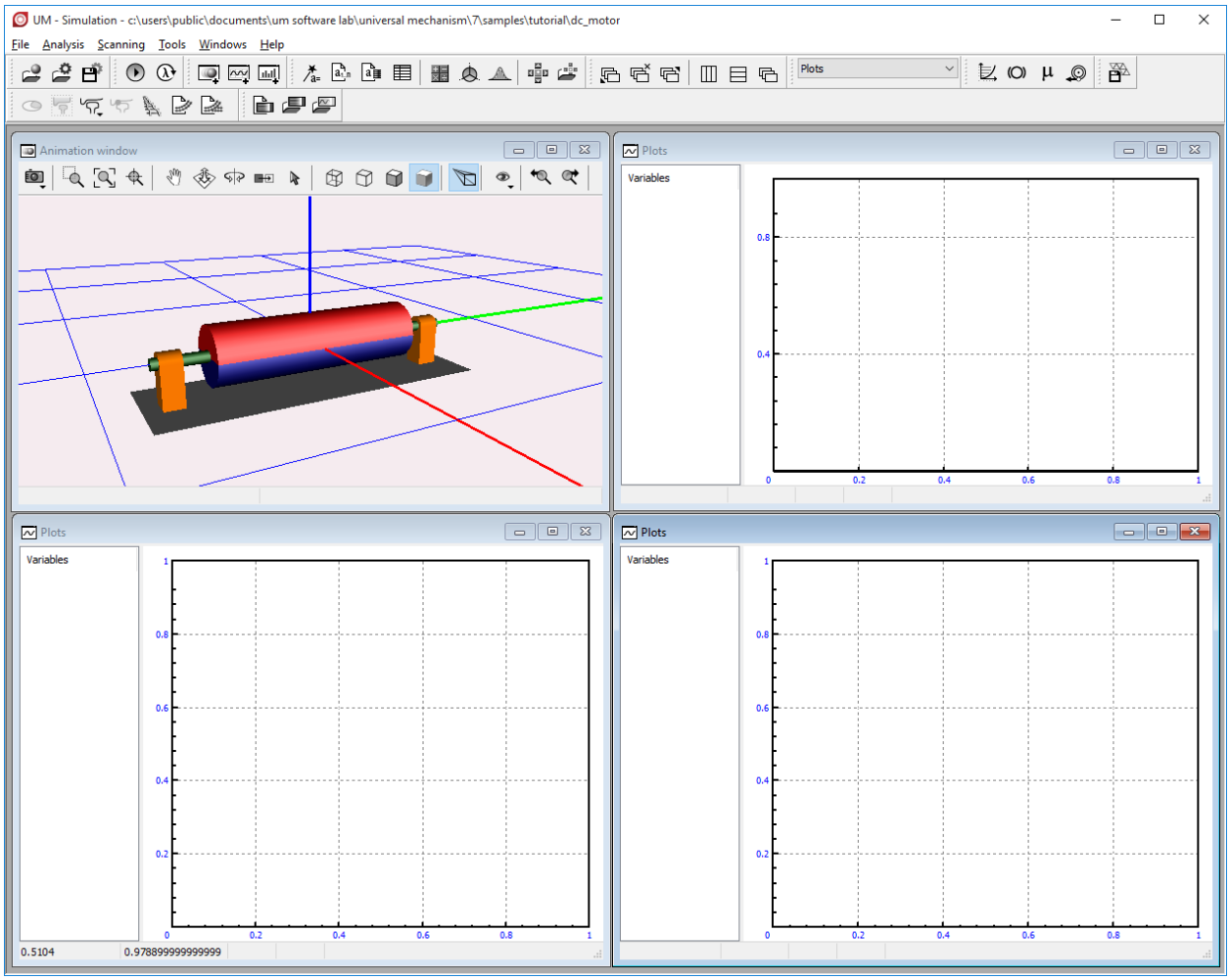


3.3.4. Simulation of motion

Now our model is completely prepared and ready for the subsequent analysis. Now we will create one animation and several graphic windows. We will plot the following variables: active and resistance moments, angular velocity of the shaft, current and voltage.

Preparing for the simulation

1. Create a new animation window, **Tools/Animation window…** menu command.
2. Create three graphical windows, **Tools/Graphical window…** menu command. Arrange windows like it is shown in the figure below.



3. Create new **Wizard of variables (Tools / Wizard of variables)**.

Let us plot a time history of the active moment that acts from the DC motor to the shaft, as well as resistance moment.

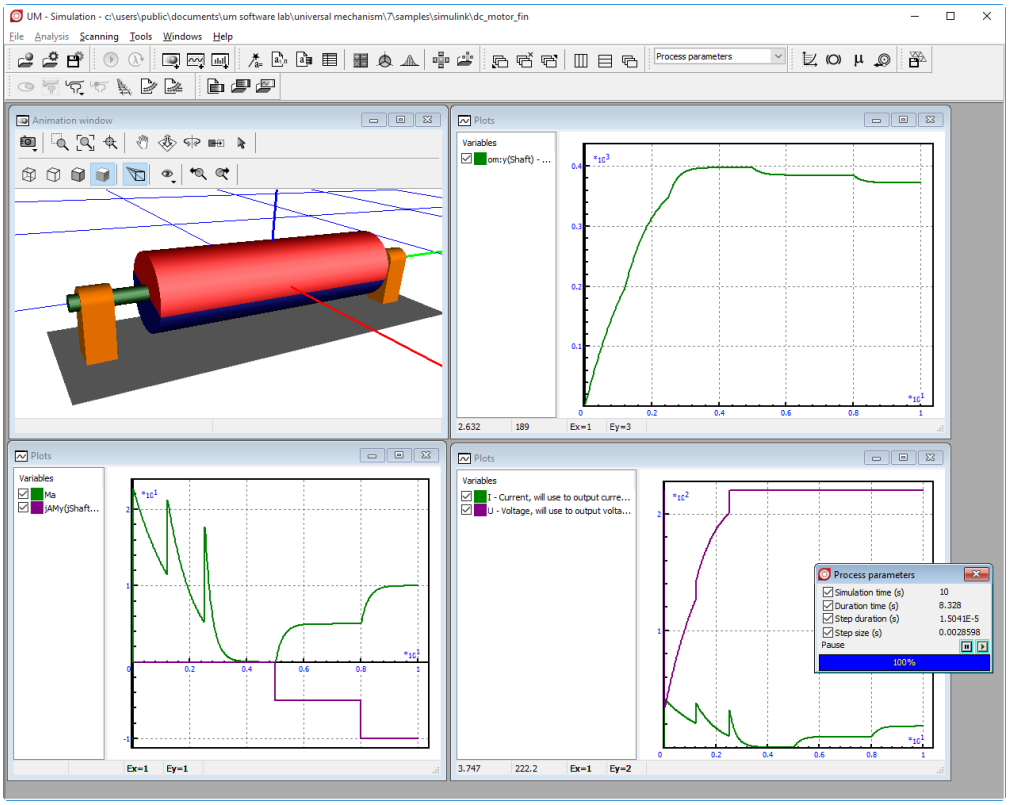
4. In the **Wizard of variables** select the **Identifier** tab. In the list of identifiers select **Ma**. Click  button to create new variable. The new variable **Ma** appears in the container of variables.
5. Select the **Joint force** tab. Select **jShaft** joint. Set **Type** to **Torque**, **Component** to **Y**. In the **Acts on body** list select **Shaft**. Create new variable by clicking  .

6. Now you can see two variables in the container of variables in the **Wizard of variables**: active and resistance moments. Select these variables and drag with the help of the mouse to the left lower graphical window.
7. Create and drag to the right upper graphical window a new window – angular velocity of the shaft (**Angular variables** tab, **Type of variable** = **Ang. velocity**, **Component = Y**).
8. Create and drag to the right lower graphical window two more variables – current and voltage from the Matlab/Simulink DC motor (**Identifier** tab, **I** and **U identifiers**).
9. Close **Wizard of variables**.

Simulation of motion

1. From **Analysis** menu select **Simulation**. **Object simulation inspector** appears.
2. Select the **Solver / Simulation process parameters** tab. Set **Solver** to **BDF**, **Simulation time** to **10**, **Step size for animation** to **0.02**, **Error tolerance** to **0.001**.
3. Click the **Integration** button.
4. During the simulation process plot of variables are shown in the graphical windows, see figure below.

Let us discuss obtained results. We can see leaps in the beginning of the plot of electromagnetic moment that correspond turning off the starting resistors. The shaft reaches the nominal angular velocity during about 4 seconds, whereupon electromagnetic moment becomes to zero. Then at the moment $t=5$ the resistance moment changes stepwise. About in a second electromagnetic moment reaches new steady-state value. Increasing the resistance moment corresponds to increasing the electromagnetic moment and current and decreasing the angular velocity of the shaft.



4. Using CoSimulation tool

Examples of using **UM Control / CoSimulation** tool are considered below. **CoSimulation** supposes that a UM model of mechanical part of the system should be exported from UM and imported in Matlab/Simulink as an S-function.

4.1. Workflow

Simulation of Matlab/Simulink models with imported UM models supposes the following steps to be done.

- Creating a Matlab/Simulink model.
- Including an S-*function* element, that will be treated as a mechanical model, into the Matlab/Simulink model.
- Creating a model of mechanical system in **UM Input**.
- Loading the prepared model in **UM Simulation**. Setting connection between a mechanical part and a control scheme via parameters and variables. Generating the m-file.
- Simulation of dynamics of the complex model under Matlab/Simulink environment.

UM model that is included into Matlab/Simulink model is considered as a black box with some number of input and output signals. Inputs of a UM model are assigned to some UM *parameters* and usually have sense of a force/torque on an executive device. Outputs of the UM model are assigned with *variables* that could be obtained with the help of **Wizard of variables** and usually have sense of some kinematical performances of the mechanical system like positions or accelerations.

4.2. Inverted pendulum

Ready-to-use model with completely prepared .cosim and m files is situated in {UM Data}\SAMPLES\cosimulation\inv_pend_cosim directory. Please check whether such model in the mentioned directory or load it using the following link:

www.universalmechanism.com/download/90/inv_pend_cosim.zip.

In this lesson we will not consider all steps concerning creating UM and Matlab/Simulink models, and will mainly discuss questions concerning connections between UM and Matlab/Simulink models.

4.2.1. Preparing Matlab/Simulink model

The Matlab/Simulink model of the control scheme for the model of the inverted pendulum shown in the Figure 4.1 is very similar to the one considered above in Figure 3.1 but have one significant difference – includes an S-Function. This component can be found in **User-Defined Functions** Simulink library. By default the name of the S *Function* is **System**.

Exactly this S-Function will provide the connection between Matlab/Simulink and UM models. The model of the inverted pendulum we can see the only input of the S-*Function* that corresponds to the control force applied to the cart and the only output that corresponds to the angle of deviation of the pendulum from the vertical position.

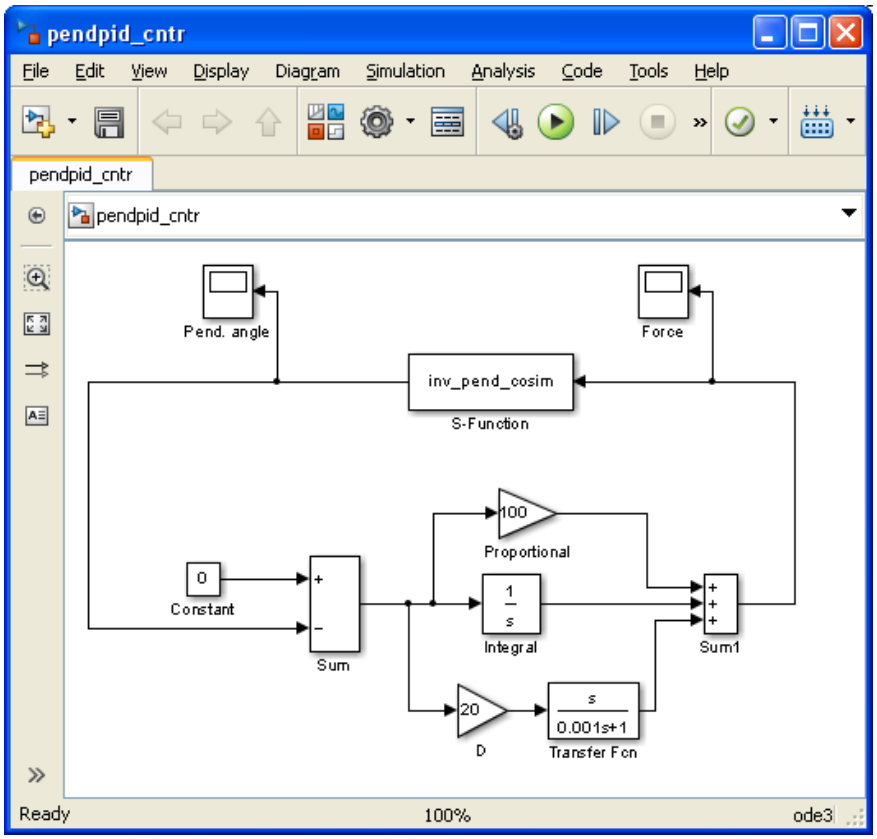


Figure 4.1. Matlab/Simulink model of the control scheme for inverted pendulum with mechanical part given as S-function

Ready-to-use Matlab/Simulink model you can find in the file {UM Data}\SAMPLES\cosimulation\inv_pend_cosim\pendpid_cntr, see Figure 4.1. We will come back to Matlab/Simulink model later and now let us consider basic features of export of UM model for the posterior usage under Matlab/Simulink environment.

4.2.2. Export of UM-model

Now we will describe input and output signals for UM model that correspond to signals of the S Function and save necessary settings and generate a special m file for the S Function.

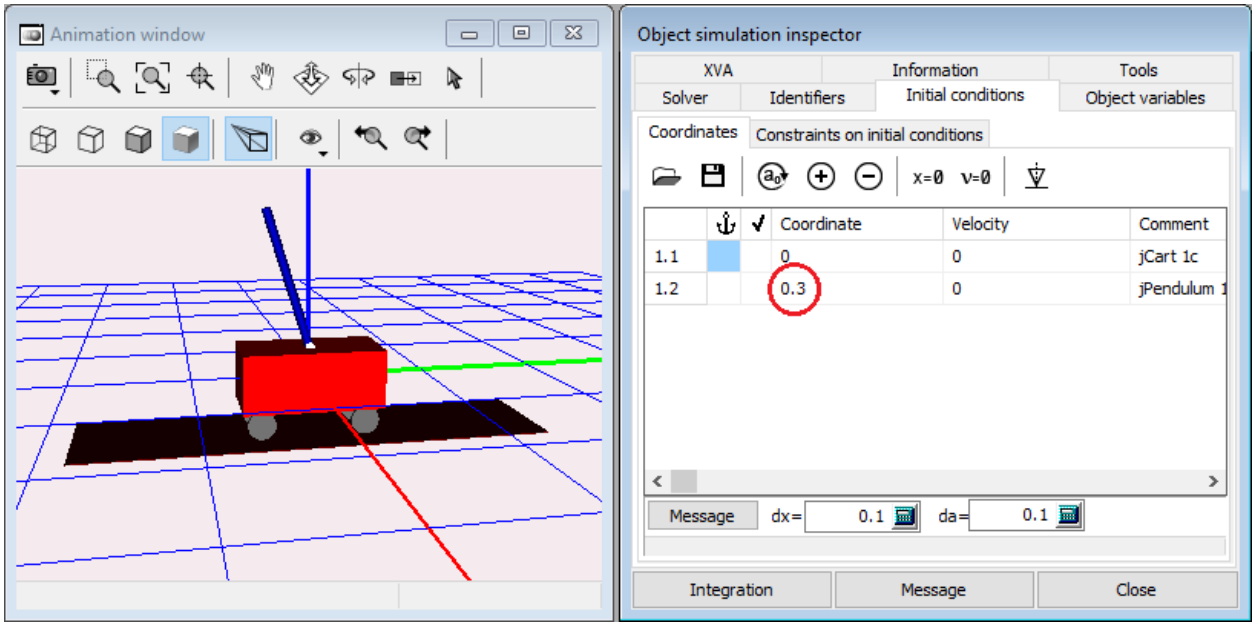
Loading UM model

1. Run **UM Simulation**.
2. Load {UM Data}\SAMPLES\cosimulation\inv_pend_cosim model.

Setting initial condition

Before coming to the rest part of the manual make sure that inverted pendulum is really deflected from ideal vertical position. If not, just deflect the pendulum to see how the control scheme really works.

1. From the **Analysis** menu select **Simulation…. Object simulation inspector** appears.
2. Point to the **Initial conditions/Coordinates** tab.
3. Let's deflect the pendulum with 0.3 radian. Set **Coordinate/1.2** to **0.3**.



Export UM model to Matlab/Simulink

1. Select the menu command **Tools / Wizard of export to Matlab/Simulink….**
   The window of **Wizard of export to Matlab/Simulink** appears.

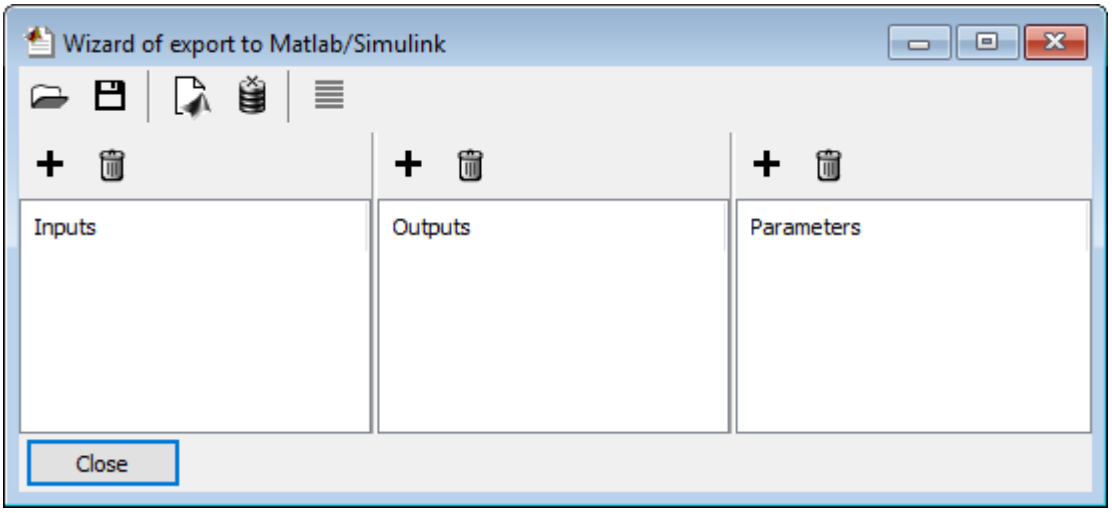
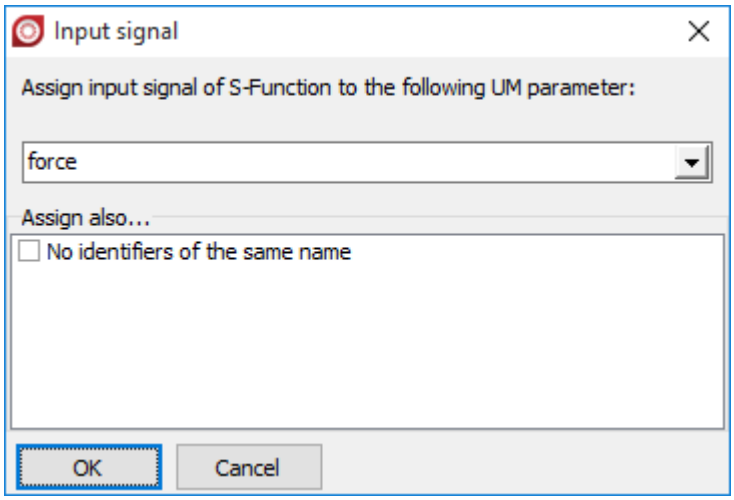


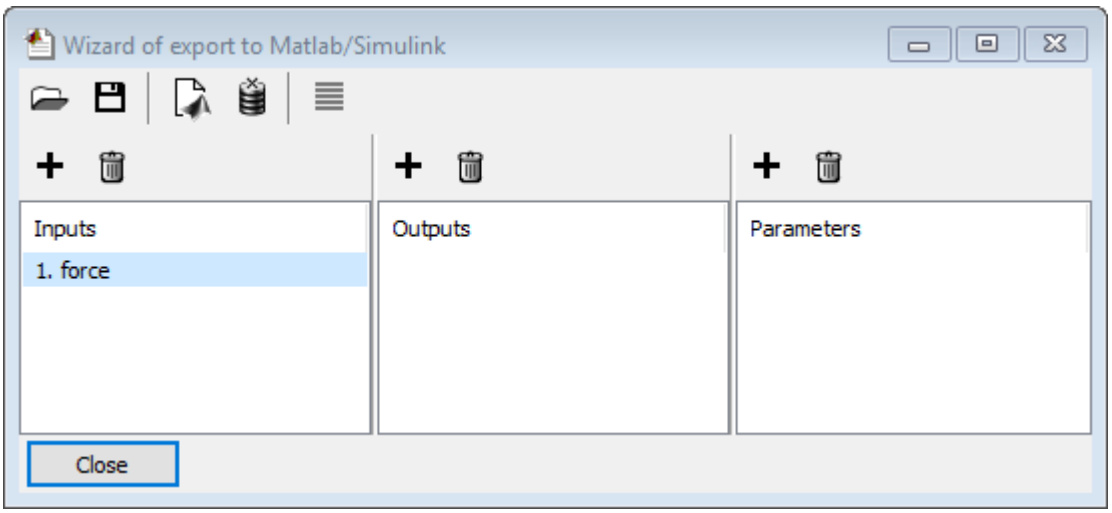
Figure 4.2. Wizard of export to Matlab/Simulink

Now we should create one input signal and assign it to the control force applied to the cart and one output signal parameter that corresponds to the angle of deviation of the pendulum from the vertical position.

Assigning input signals

1. Click + button above the **Inputs** list. New dialog window to select the UM model parameters to be assigned with the S Function input appear.
2. Select **force** parameter in the drop-down list.
3. Click **OK**.

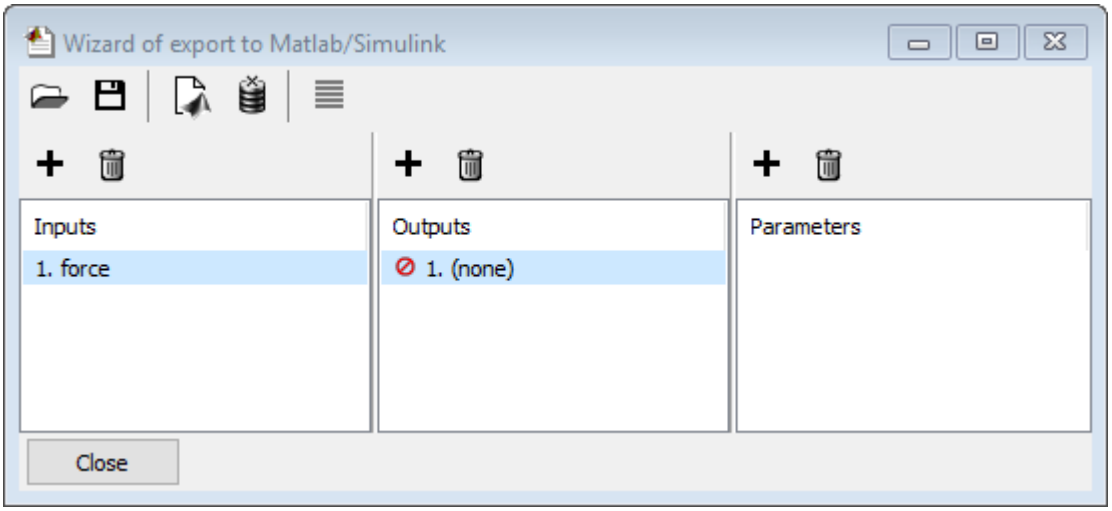




Assigning output signals

Let's create the *'pendulum angle from vertical' variable* with the help of the **Wizard of variables** and then assign that variable with the first output signal from the S Function.

1. Click + button above the **Outputs** list. One more output signal appears in the **Outputs** list.



2. Open **Wizard of variables**.
3. Select the **Angular variables** tab.
4. Select **Pendulum** in the list of bodies in the left, turn off **Use orientation at zero coordinates** flag, set **Type of variable** to **Rot. vector**, set **Component** to X.
5. Create the new variable with the help of the button. The new variable appears in the container of variables.

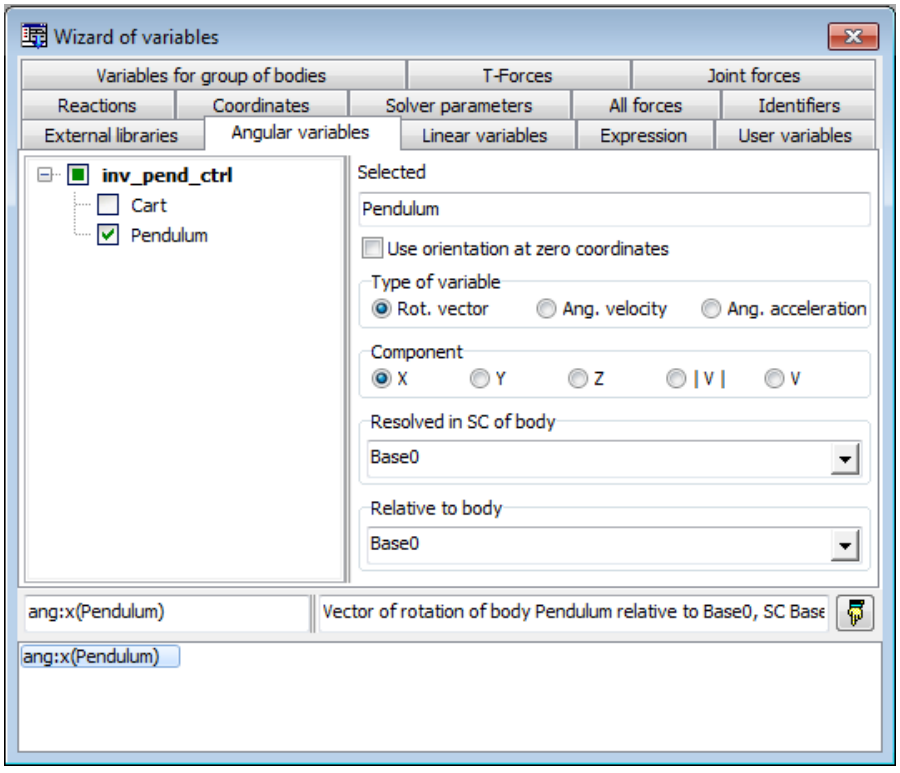
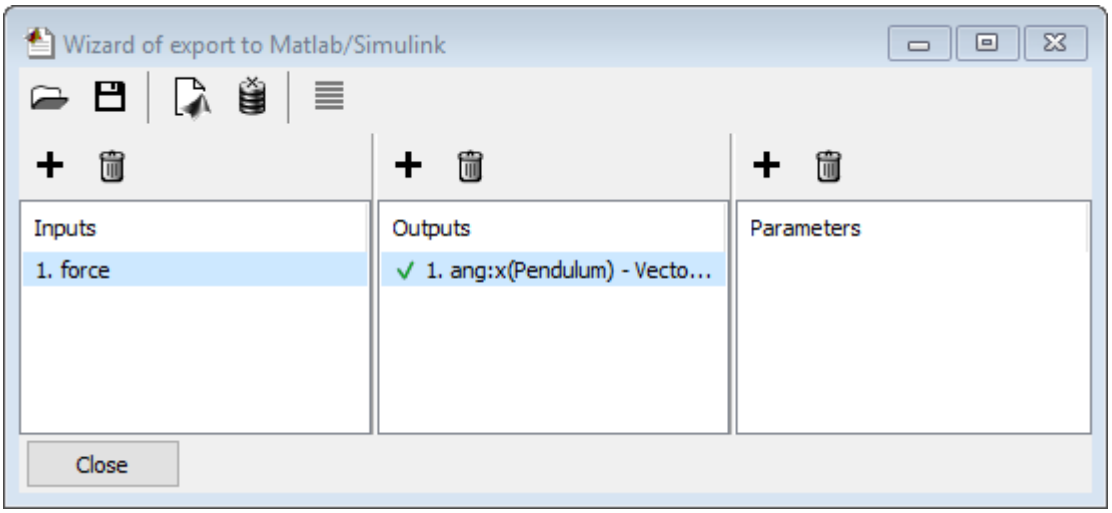


Figure 4.3. Wizard of variables

6. Drag the **ang:x(Pendulum)** from the **Wizard of variables to Matlab/Simulink interface wizard** to the position of the first output.

The first output is now selected with the green mark, what means that this output is already assigned with a variable.



All necessary settings for export of UM model to Matlab/Simulink are now finished. Now let us generate m file that will provide functioning of the UM model as the S-Function and simultaneously save all settings as a *.cosim file.

Generating m file

Now when we described numbers input and output signals, assign input signals with UM model parameter and UM *variables* with output signals. After saving the settings to .cosim file it needs to generate m-file that in fact will support connection between UM and Matlab/Simulink.

1. Click  to generate m-file. Warning message appears. It says that the list of parameters is empty. We can ignore this message by clicking **Yes**.

2. New save dialog appears. Set file name to **inv_pend_cosim** and click **Save. Inv_pend_cosim.cosim** file will be saved along with creating m-file. This file will be used by Universal Mechanism during working under Matlab/Simulink environment.

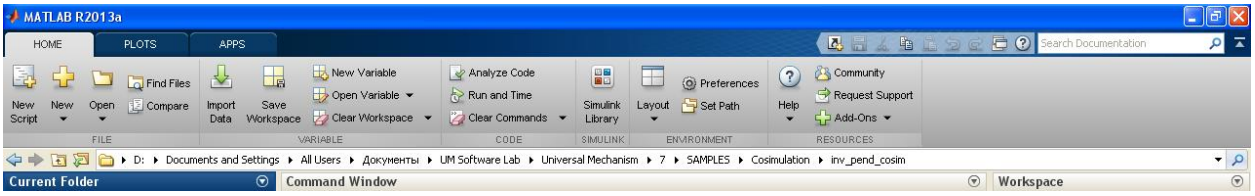
Now all actions under UM environment are finished and let us come to working under Matlab/Simulink environment.

4.2.3. Connection between UM-model and Matlab/Simulink

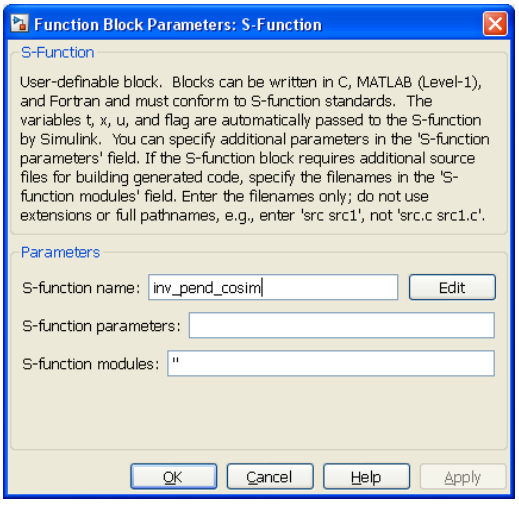
Prepared at the previous step m file by default was saved in the UM model directory. Generally it needs to copy the automatically generated m file to the directory where Matlab/Simulink is situated. For this case Matlab/Simulink model is already in the UM model directory and we do not need to copy the m file.

Now we need to set the m file name as an S-function name for the Matlab/Simulink model.

1. Run Matlab/Simulink.
2. Set **Current Directory** to **{UM Data}\SAMPLES\cosimulation\ inv_pend_cosim**.



3. Load **pendpid_cntr** with the model depicted in Figure 4.1.
4. Double click on **S-function** opens **Function block Parameters** dialog window.
5. Set **S-function name** to **inv_pend_cosim**, in fact the generated m-file name.



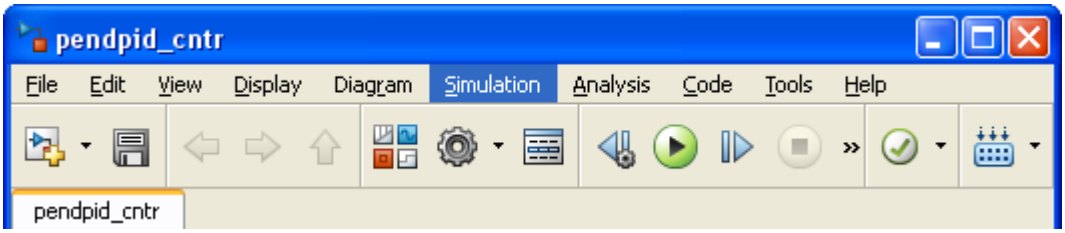
6. Click **OK**.

4.2.4. Simulation of motion

Our model is ready for the simulation. Now we will set simulation parameters and run the simulation of the complex model including mechanical part imported from UM as an S Function and Matlab/Simulink model.

Simulation parameters

1. Click menu command **Simulation/Configuration Parameters**.



2. Select **Solver** tab.
3. Set **Start time** to **0.0**, **Stop time** to **0.3**, in the **Type** list select Fixed-step, set **Fixed-step size** to **1E 4**, **Solver** to **ode3(Bogacki-Shampine).** After all these setting the **Configuration Parameters** dialog window should look like the window in Figure 4.4.

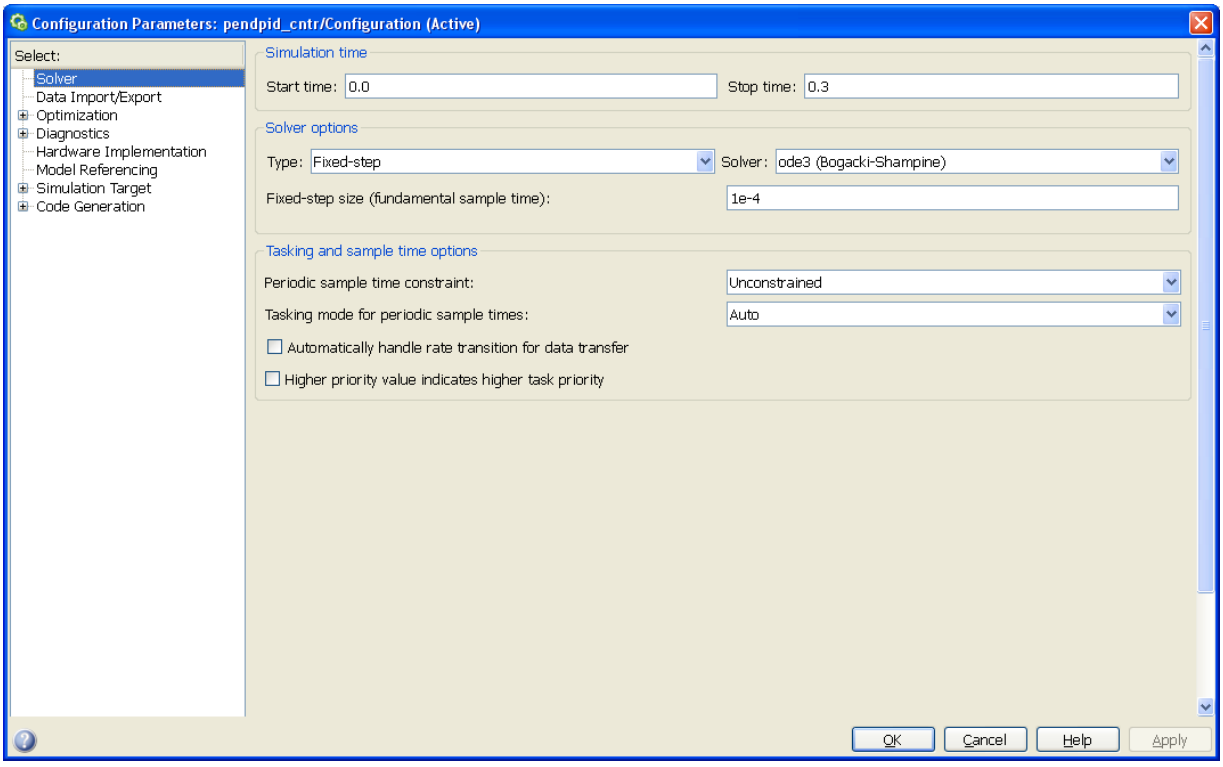


Figure 4.4. Configuration Parameters dialog window (Matlab R2006a)

4. Click **OK** button

Simulation of motion

1. Open all **Scope** windows by double clicking. Scopes will show angle of pendulum and the control force.
2. Run simulation.
3. When simulation stops you can see angle of rotation of the pendulum and the control force like in Figure 4.5 and Figure 4.6.

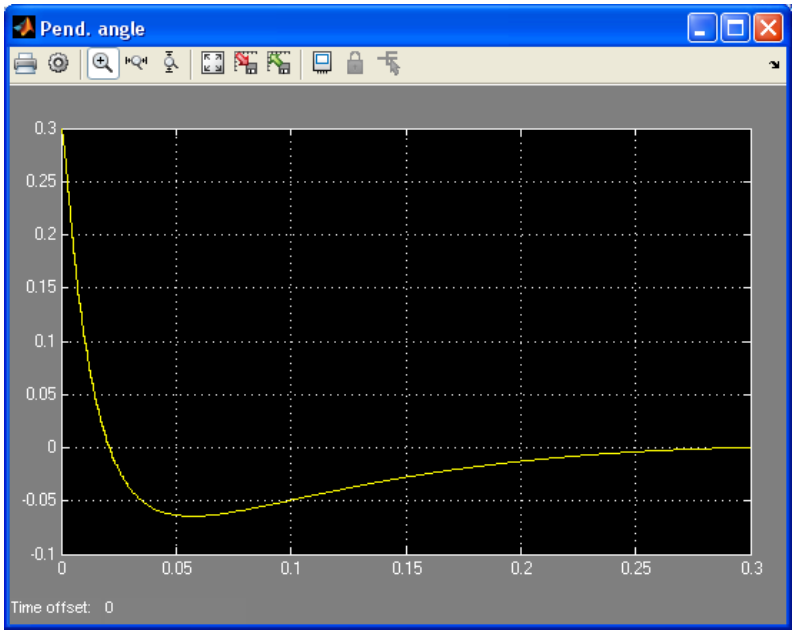


Figure 4.5. Angle of deviation of pendulum from the vertical position

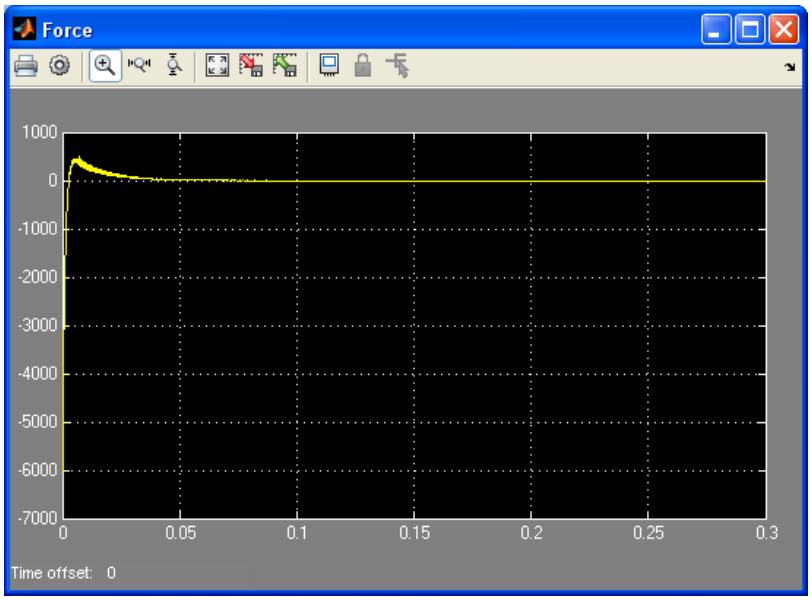


Figure 4.6. Control force

4.3. DC motor

Ready-to-use model with completely prepared .cosim and m-files is situated in {UM Data}\SAMPLES\cosimulation\dcmotor_cosim directory. Please check whether such model in the mentioned directory or load it using the following link: www.universalmechanism.com/download/90/dcmotor_cosim.zip.

4.3.1. Preparing Matlab/Simulink model

Prepared Matlab/Simulink model if the DC-motor with the S Function for connection with mechanical part described as UM model is shown in Figure 4.7. The model of the **DC motor** subsystem is shown in Figure 3.12. S-Function has one input (driving torque) and one output (shaft angular velocity).

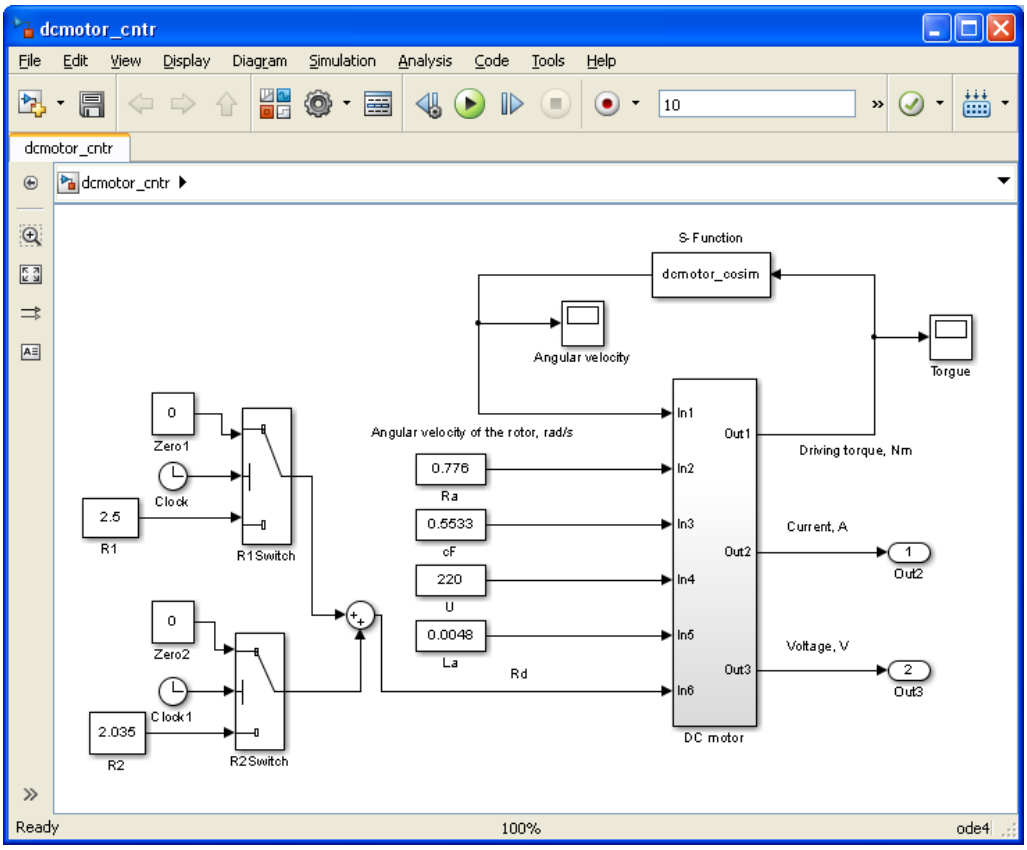


Figure 4.7. Matlab/Simulink model of DC motor with S Function

The model presented in Figure 4.7 is situated here:
{UM Data}\SAMPLES\cosimulation\dcmotor_cosim\dcmotor_cntr. We will come back to Matlab/Simulink a bit later and now let us export UM model first.

4.3.2. Export of UM-model

Loading UM model

1. Run **UM Simulation**.
2. Load {UM Data}\samples\cosimulation\dcmotor_cosim model.

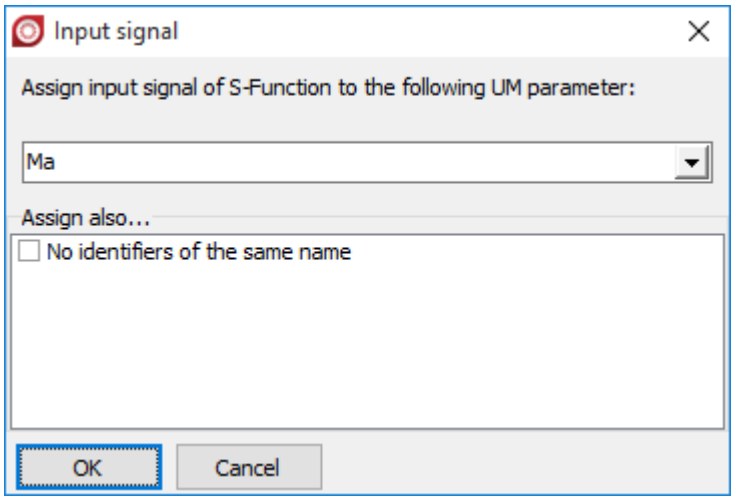
Export UM-model to Matlab/Simulink

1. Select the menu command **Tools / Wizard of export to Matlab/Simulink**. The window of **Wizard of export to Matlab/Simulink** appears.

Now we need to create one input and one output signals. We will assign driving torque as the input signal and the angular velocity of the shaft as the output signal.

Assigning input signals

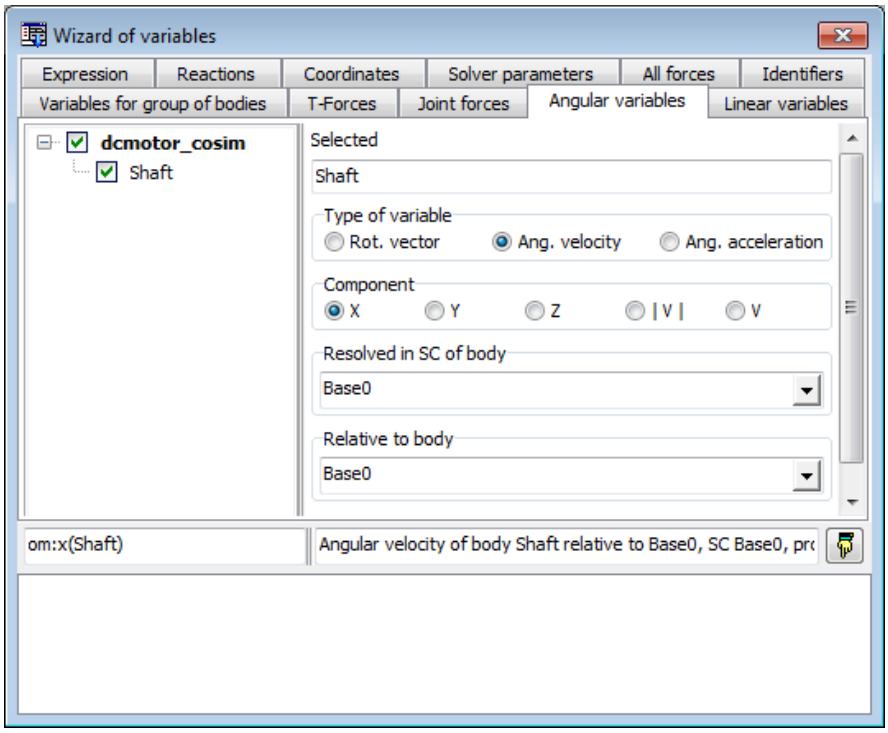
1. Click **+** button above the **Inputs** list. New dialog window to select the UM model parameters to be assigned with the S-Function input appear.
2. Select **Ma** parameter in the drop-down list and click **OK**.



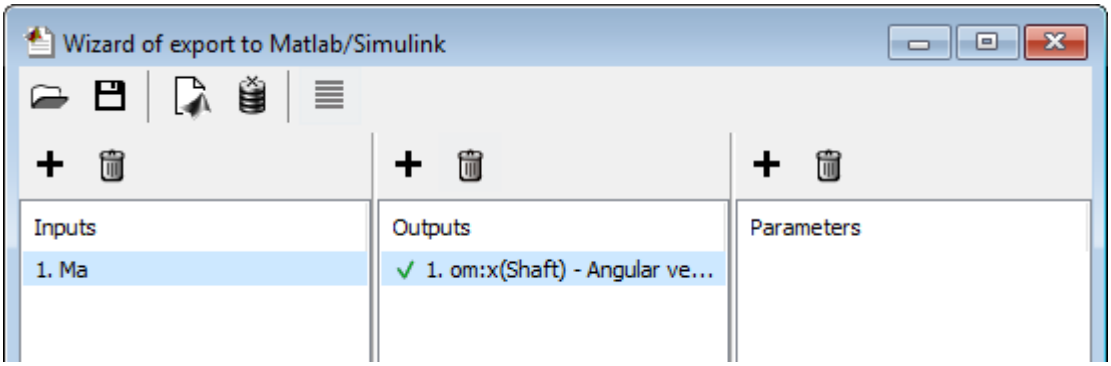
Assigning output signals

Let's create the *'angular velocity of the shaft'* variable with the help of the **Wizard of variables** and then assign that variable with the first output signal from the S-Function.

1. Click + button above the **Outputs** list. One more output signal appears in the **Outputs** list.
2. Open **Wizard of variables**.
3. Select the **Angular variables** tab.
4. Select **Shaft** in the list of bodies in the left, set **Type of variable** to **Ang. velocity**, set **Component** to **Y**. Create the new variable with the help of the  button. The new variable **om:y(Shaft)** appears in the container of variables.



5. Drag the **om:y(Shaft)** from the **Wizard of variables** to **Matlab/Simulink interface wizard** to the output signal. The first output is now selected with the green mark, what means that this output is already assigned with a variable.



Generating m file

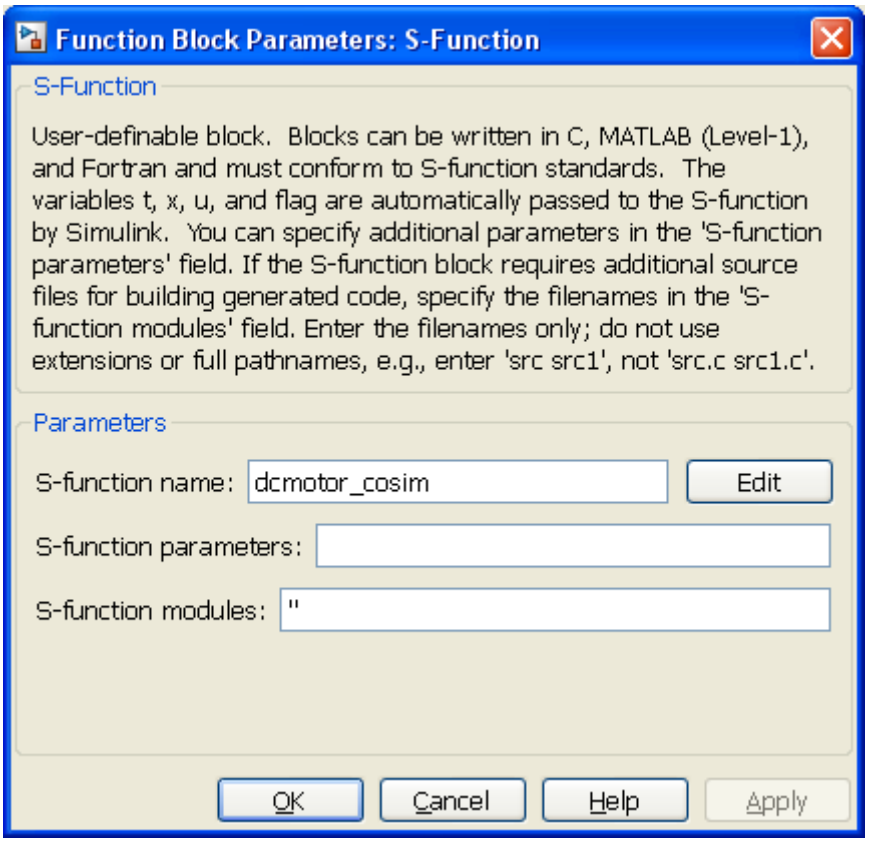
1. Click  to generate m-file. Warning message appears. It says that the list of parameters is empty. We can ignore this message by clicking **Yes**.
2. New save dialog appears. Set file name to **dcmotor_cosim** and click **Save**.

Now all actions under UM environment are finished and let us come to working under Matlab/Simulink environment.

4.3.3. Connection between UM-model and Matlab/Simulink

Now we need to set the m file name as an S-function name for the Matlab/Simulink model.

1. Run Matlab/Simulink.
2. Load **dcmotor_cosim** with the model depicted in Figure 4.7.
3. Double click on **S-function** opens **Function block Parameters** dialog window.
4. Set **S-function name** to **dcmotor_cosim**, in fact the generated m-file name.



5. Click **OK**.

The model is ready for simulation.

4.3.4. Simulation of motion

Simulation of motion

1. Open all **Scope** windows by double clicking. Scopes will show angular velocity of the shaft and active electromagnetic torque.
2. Run simulation.
3. When simulation stops you can see plots like in Figure 4.8 and Figure 4.9.

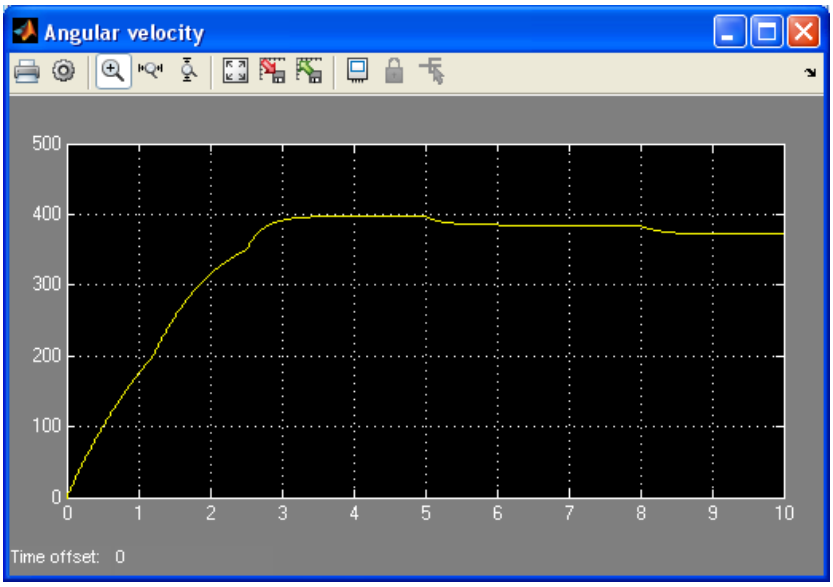


Figure 4.8. Angular velocity of the shaft

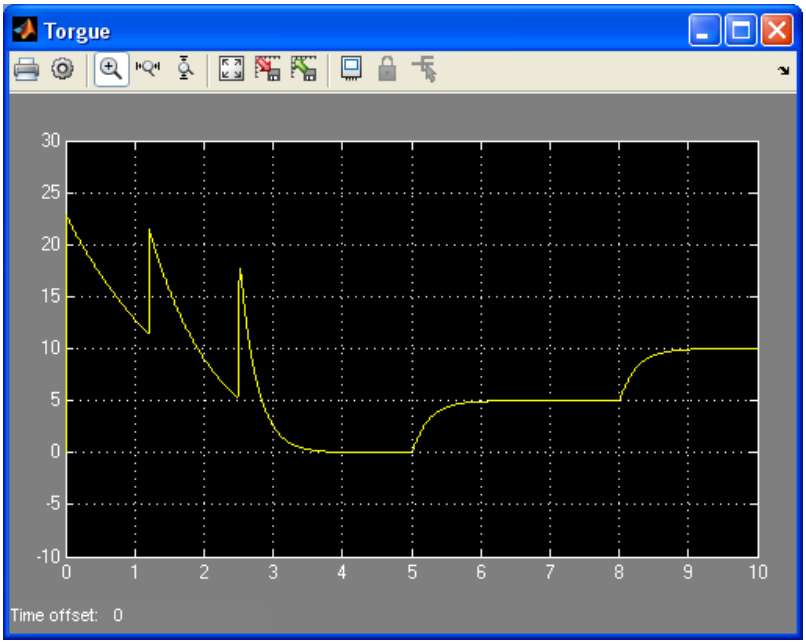


Figure 4.9. Driving torque

4.4. Anti-lock braking system of car (ABS)

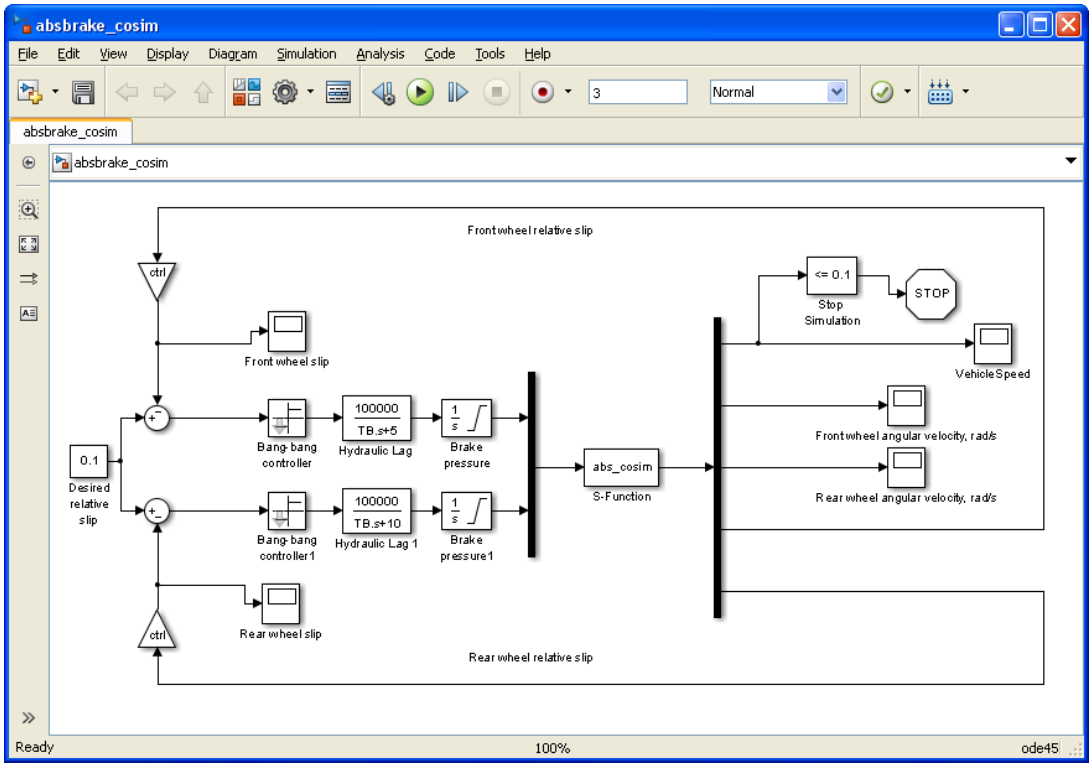
Ready-to-use model with completely prepared .cosim and all rest setting files is situated in {UM Data}\SAMPLES\cosimulation\vaz21_09abs directory. Please check whether such model in the mentioned directory or load it using the following link: www.universalmechanism.com/download/90/vaz21_09abs.zip.

In this lesson we will not consider all steps concerning creating UM and Matlab/Simulink models, and will mainly discuss questions concerning connections between UM and Matlab/Simulink models.

4.4.1. Matlab/Simulink model of ABS

An anti-lock braking system, or ABS is a safety system which prevents the wheels on a motor vehicle from locking up while braking. A rotating road wheel allows the driver to maintain steering control under heavy braking by preventing a skid and allowing the wheel to continue interacting tractively with the road surface as directed by driver steering inputs. While ABS offers improved vehicle control, and may decrease stopping distances on dry and especially slippery surfaces, it can also increase braking distance on loose surfaces such as snow and gravel.

Let us consider a simple ABS model with 2-channel ABS, one channel per one axle. The considered ABS model has two inputs that are in fact frictional torques on front and rear wheels and a number of outputs such as vehicle velocity, angular velocity and normalized slip of front and rear wheel.

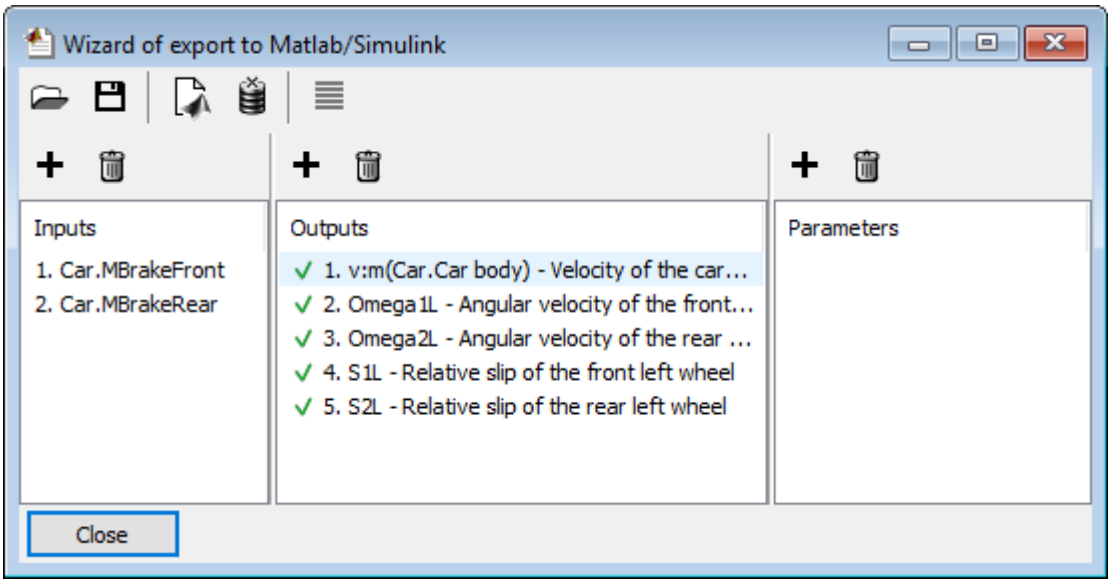


4.4.2. Export of UM-model

Let us to skip detailed description of all settings and present just some screen shots to illustrate the basic features.

Wizard of export to Matlab/Simulink with all input and output signals is shown below. To simulate the model under Matlab/Simulink environment you have to re-generate m-file.

1. Run **UM Simulation**.
2. Load the model {UM Data}\SAMPLES\cosimulation\vaz21_09abs.
3. Run **Wizard of export to Matlab/Simulink (Tools/ Wizard of export to Matlab/Simulink…** menu command)
4. In the **Wizard of export to Matlab/Simulink** load **abs_cosim.cosim** file.
5. Click **Yes** on prompt to load all configuration files.
6. To re-generate the m file click  button.



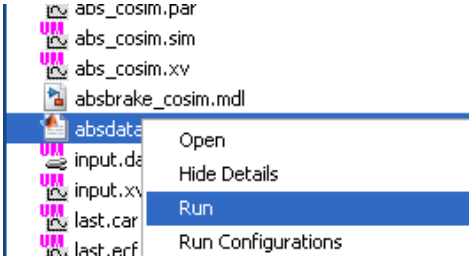
Now model is ready to be connected with Matlab/Simulink.

4.4.3. Simulation of motion

Let us consider simulation of motion firstly with ABS and then without ABS and compare the obtained results. Plots of vehicle velocity, angular velocities of front and rear wheels, as well as normalized relative slip vs. time for both with/without ABS cases are presented below.

4.4.3.1. Loading the model

1. Run **Matlab**.
2. Set **Current Directory** to {UM Data}\SAMPLES\cosimulation\ vaz21_09abs.
3. Now we need to initialize some Matlab constants. Select **absdata.m** file in the list of files in the main Matlab window and click **Run** from the context menu.



4. Load **absbrake_cosim.mdl** model of the ABS.
5. Run simulation.

4.4.3.2. Simulation of motion with ABS

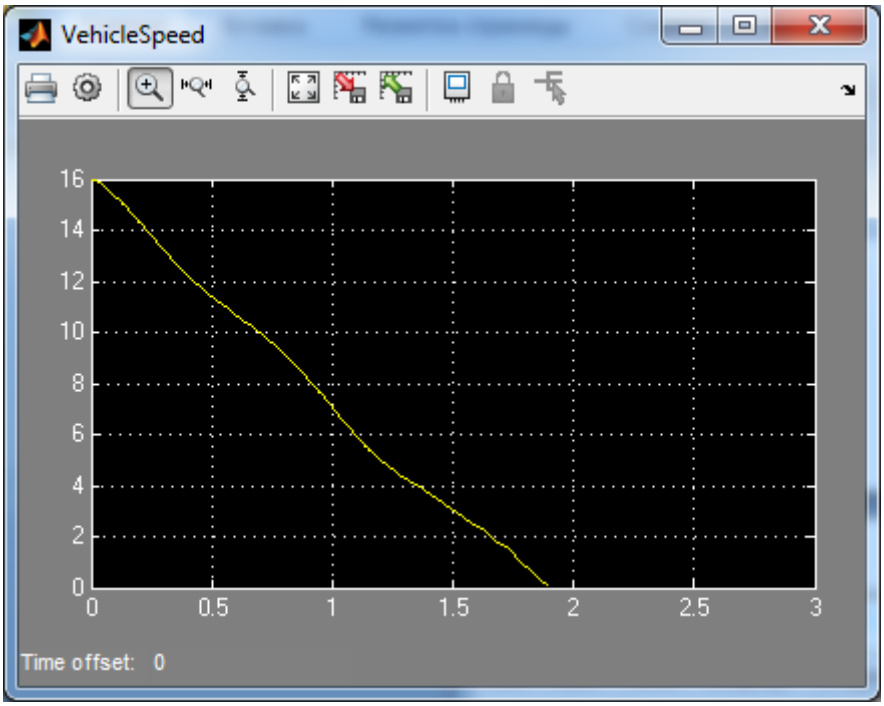


Figure 4.10. Vehicle velocity for braking with ABS, m/s

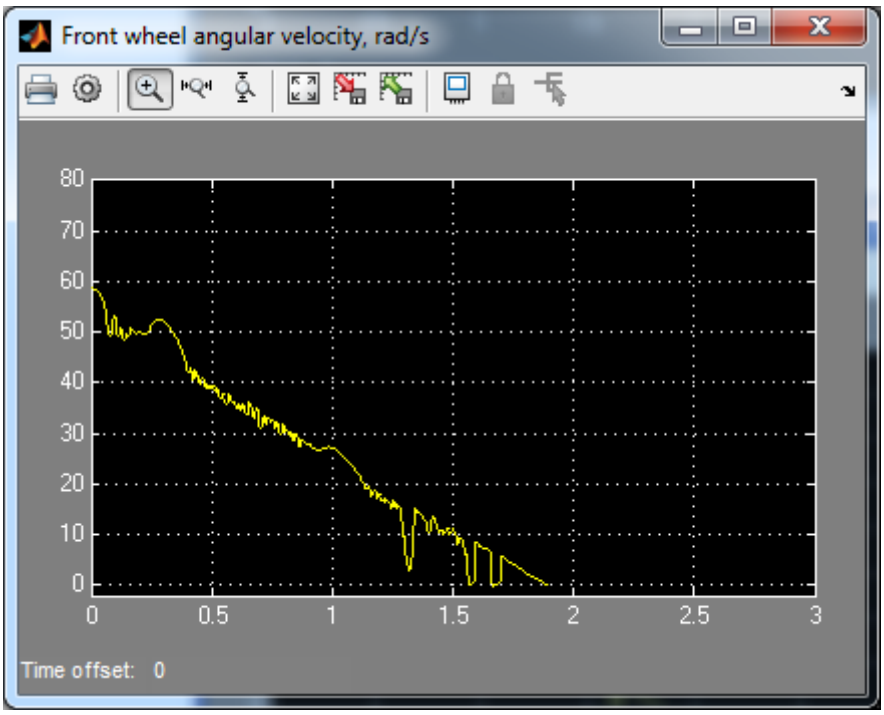


Figure 4.11. Angular velocity of a front wheel, rad/s

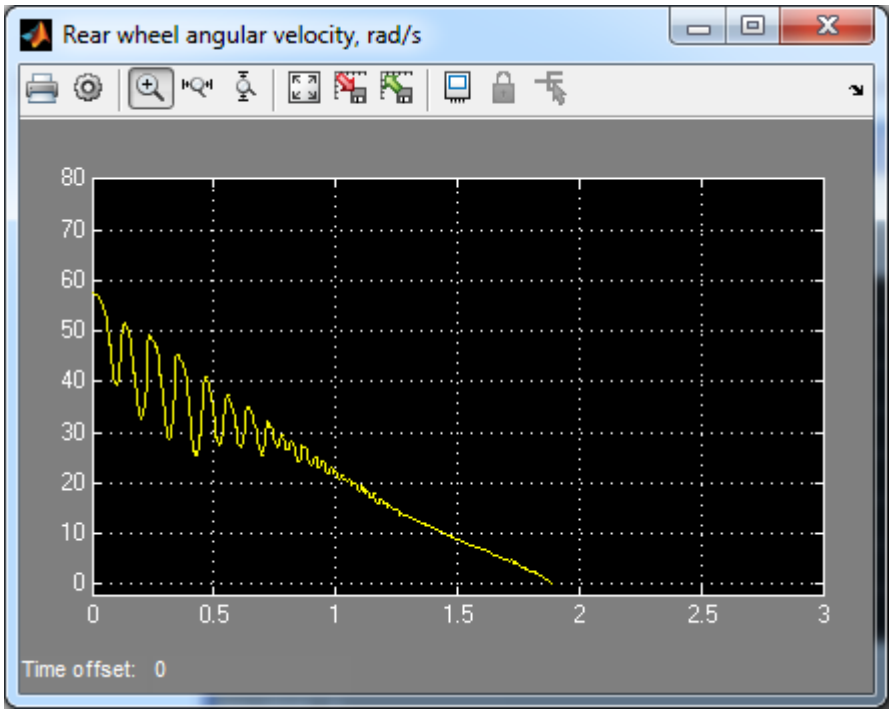


Figure 4.12. Angular velocity of a rear wheel, rad/s

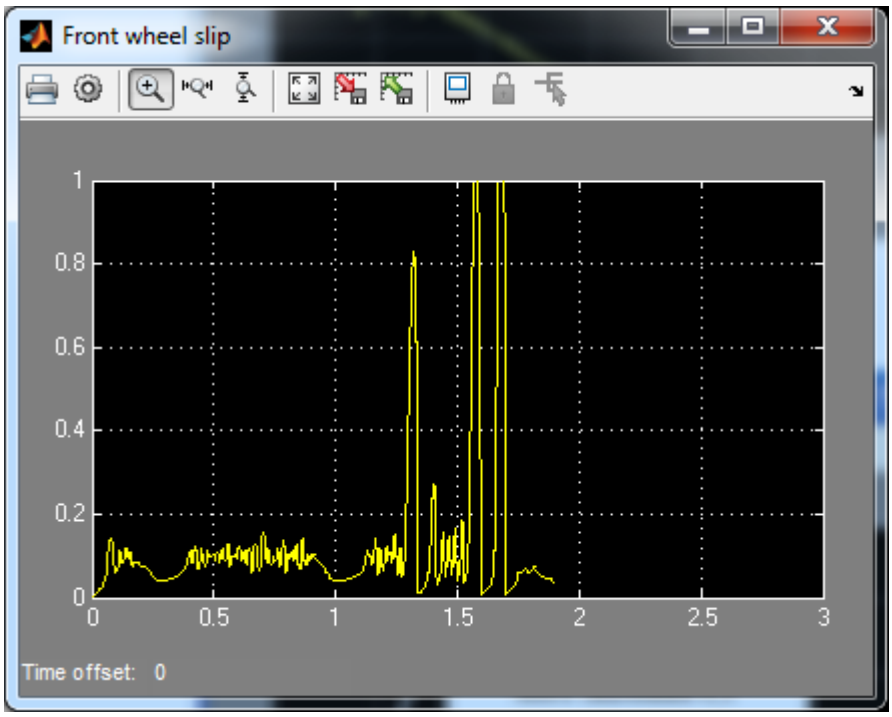


Figure 4.13. Normalized relative slip of front wheels

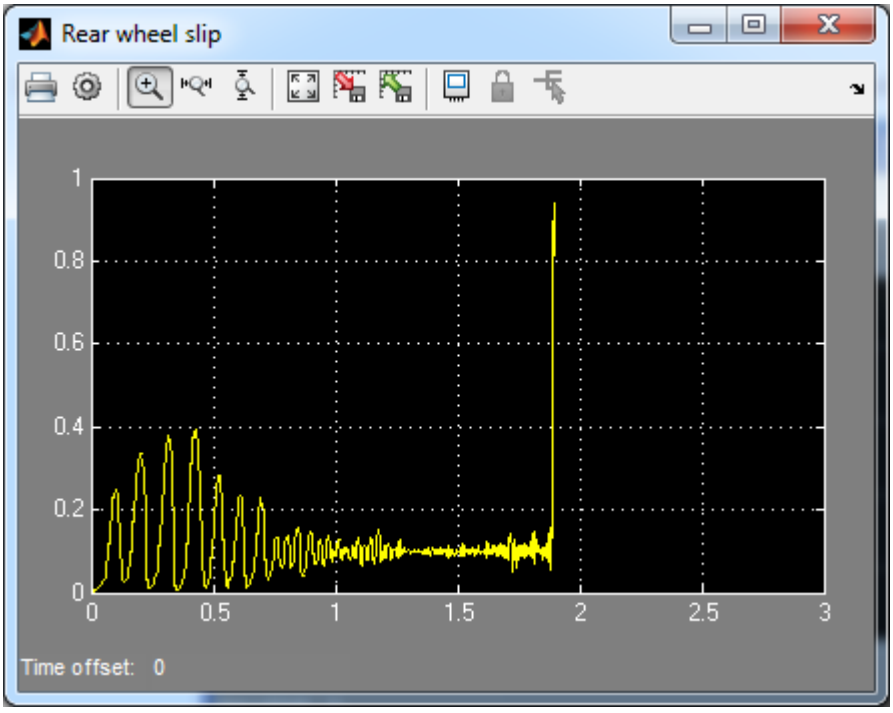


Figure 4.14. Normalized relative slip of rear wheels

4.4.3.3. Simulation of motion without ABS

Let us consider simulation results without ABS.

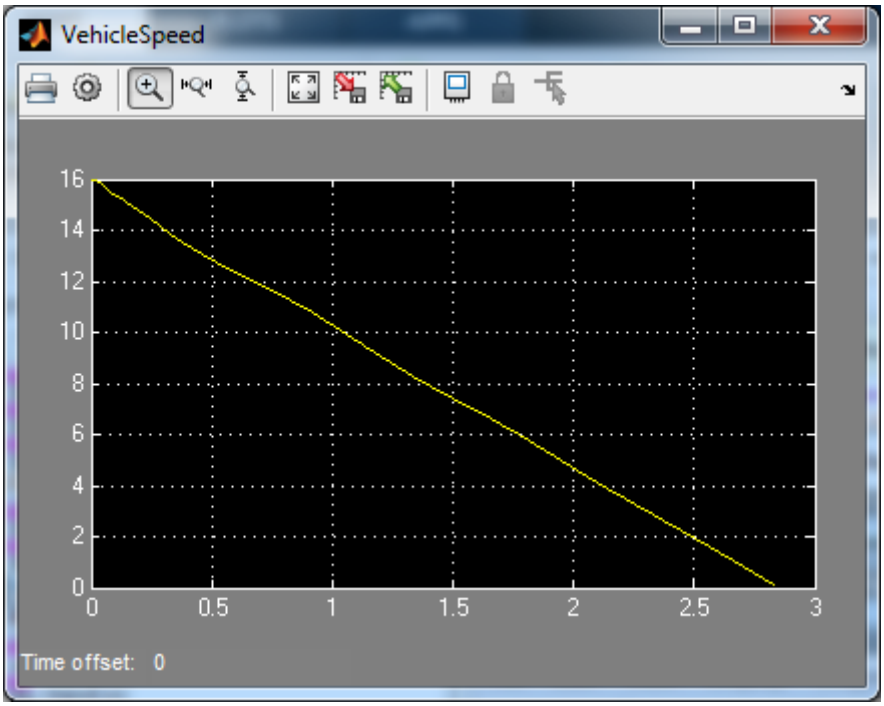


Figure 4.15. Vehicle velocity for braking without ABS, m/s

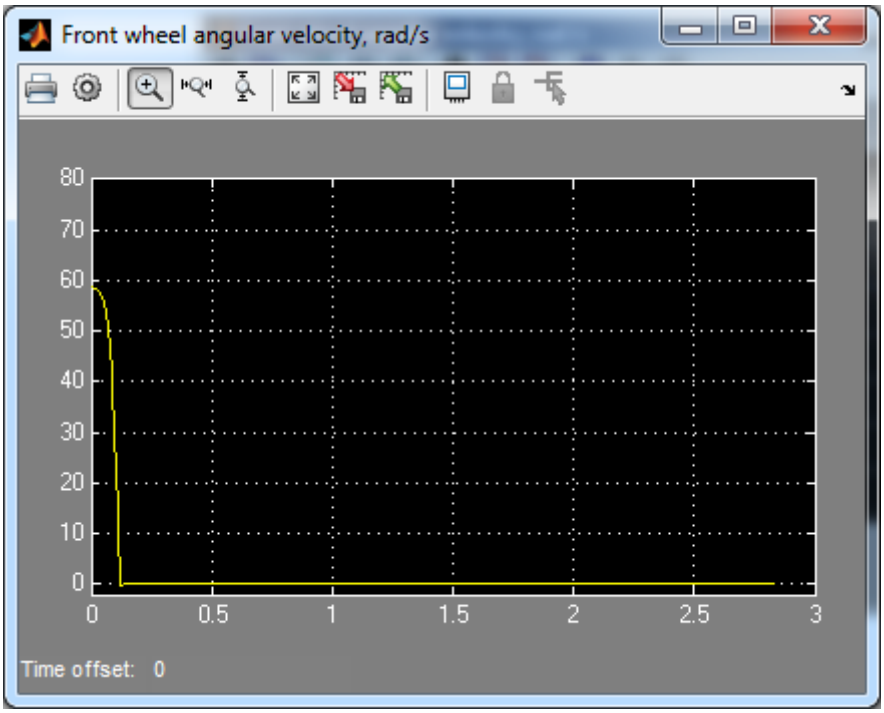


Figure 4.16. Angular velocity of a front wheel without ABS, rad/s

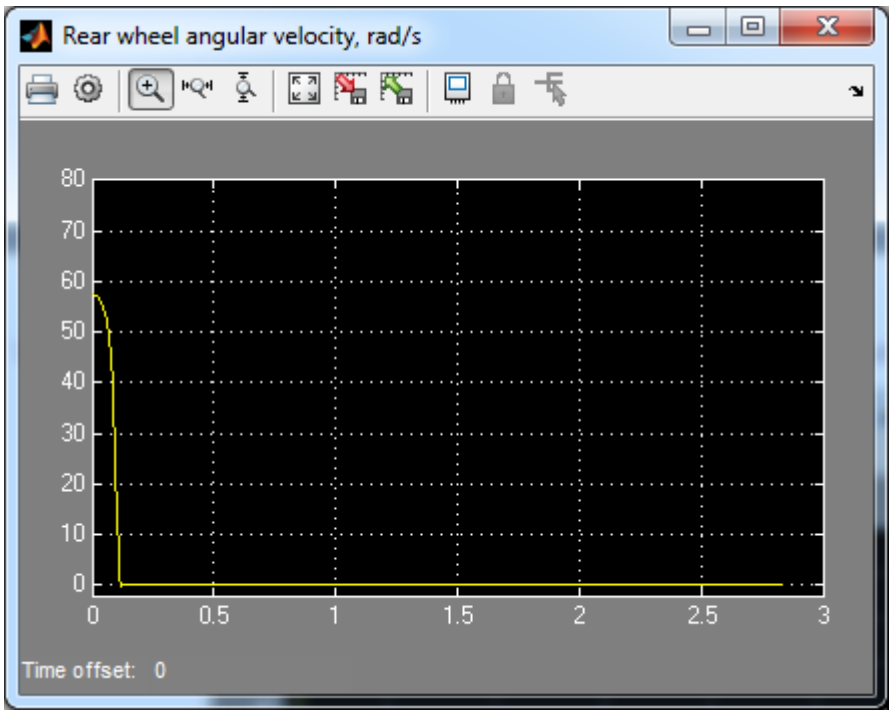


Figure 4.17. Angular velocity of a rear wheel without ABS, rad/s

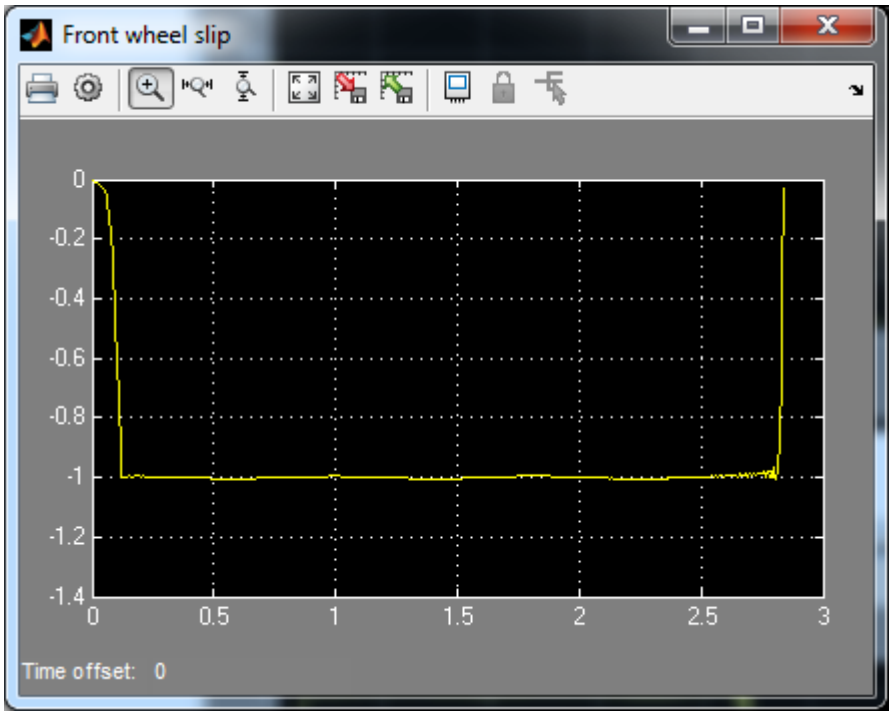


Figure 4.18. Normalized relative slip of front wheels

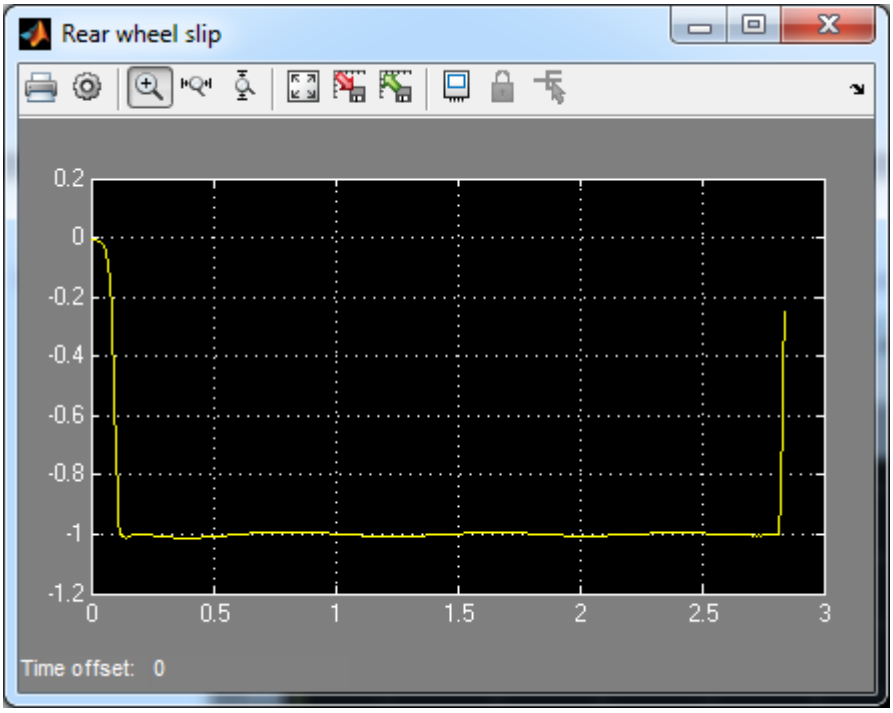


Figure 4.19. Normalized relative slip of rear wheels

Obtained results clearly showed that ABS reduces stopping time and distance correspondingly: 2.7 s with ABS vs. 1.75 s without ABS. It is seen that high braking moments block wheels and the vehicle start skidding.