universal mechanism

User`s manual

UM CoSimulation Tool

2020

Contents

19. CoSimulation tool

The **CoSimulation** tool from the **UM Control** module helps you to export a UM model from UM for posterior integration into Matlab/Simulink model. To import UM models into Matlab/Simulink you need to use so-called *S-Functions* that in fact provide data exchange between Matlab/Simulink and UM models.

S-Function generally has several input and output signals, as well as several parameters. Input signals of *S-Function* are assigned with UM model *parameters* that usually describe control forces or torques (control action). So UM model gets control forces that are calculated at the Matlab/Simulink side according to control algorithms. Output signals of *S-Function* are assigned with UM *variables* that are created with the help of **Wizard of variables** and usually refer to kinematics of a UM model. *S-Function* parameters are assigned with UM *parameters* that are assigned in the beginning of a numerical experiment and are not changed during the simulation process, for example it could be inertia and geometry parameters, stiffness and damping coefficients and so on.

S-Function should be used along with specially generated *m-file* in Matlab programming language that loads UM COM-server, a UM model and control all data exchange between applications, see Sect. 19.2. *"Structure of m-file"*, p. 19-8.

19.1. Work under UM environment

M-file for an *S-Function* is generated automatically with the help of **Wizard of export to Matlab/Simulink** under the **UM Simulation** environment.

19.1.1. Wizard of export to Matlab/Simulink

Wizard of export to Matlab/Simulink generates *m-file* as well as *.cosim* file that includes all information concerning input and output signals and *S-Function* parameters. The detailed manual how to work with **Wizard of export to Matlab/Simulink** you can find in the [Getting started: Matlab/Simulink interface] chapter of UM User's Manual. The most recent edition of the chapter you find using the link below:
www.universalmechanism.com/download/90/eng/gs_um_control.pdf.

Using **Wizard of export to Matlab/Simulink (UM Simulation | Tools | Wizard of export to Matlab/Simulink** menu item) you can create and describe necessary input and output signals and *S-Function* parameters, see Figure 19.1.

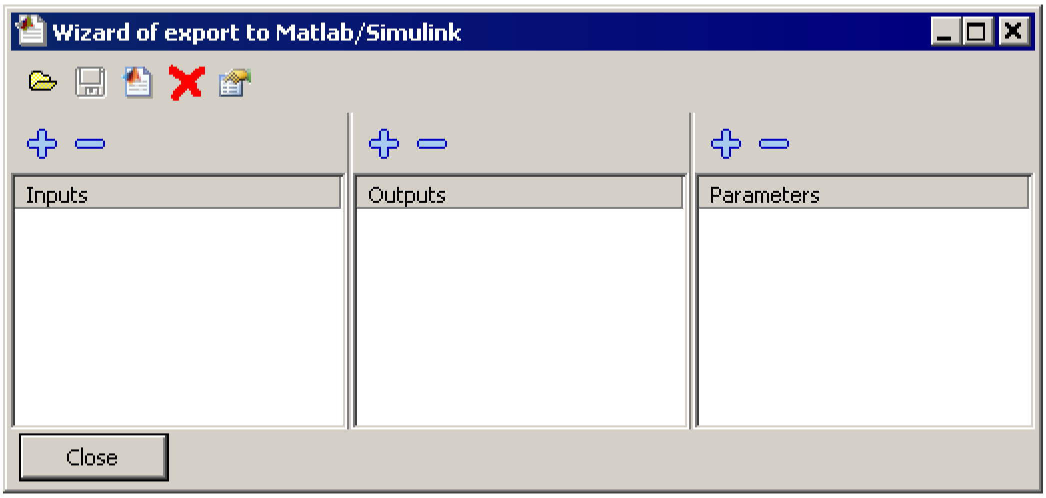

Figure 19.1. Wizard of export to Matlab/Simulink

Input signals and parameters of *S-Function* are assigned with UM-model parameters. That is why it needs to include all necessary parameterized force elements at the stage of the model description in **UM Input**. To assign input signal to a UM model parameter, double click on the necessary item in the **Inputs** list (see Figure 19.1) and select the parameters in the dialog window, see Figure 19.2.

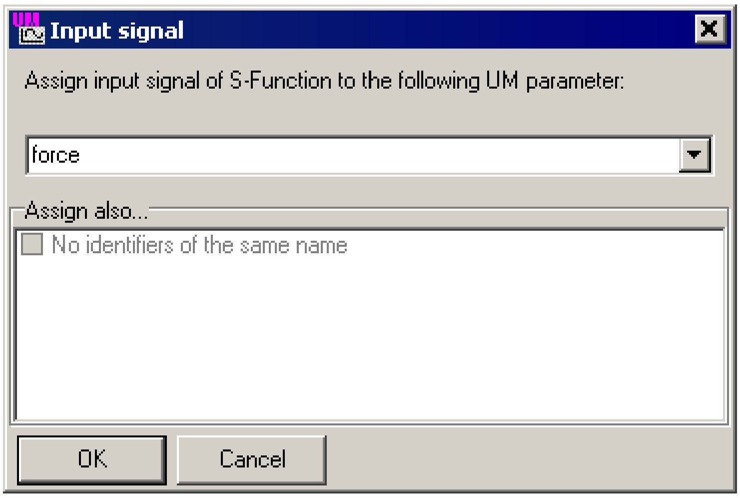

Figure 19.2. Connection input signals with UM model parameters

Output signals of the *S-Function* are assigned with UM *variables*. Use **Wizard of variables** to create necessary variables and simply *Drag-and-Drop* them to assign with the certain output signal. So, any *variable* that can be created with the help of **Wizard of variables** can be assigned as the *S-Function* output signal.

19.1.2. Including several UM models into Matlab/Simulink model

If you want to include several UM models into the same Matlab/Simulink model you have to create *S-Function* and create separate *m-file* for each UM model even for the same UM model.

Firstly you need to describe input and output signals and *S-Function* model parameters as usual and prior to generating the *m-file* you should open the **Settings of m-file generation** dialog box by clicking button, see Figure 19.3.

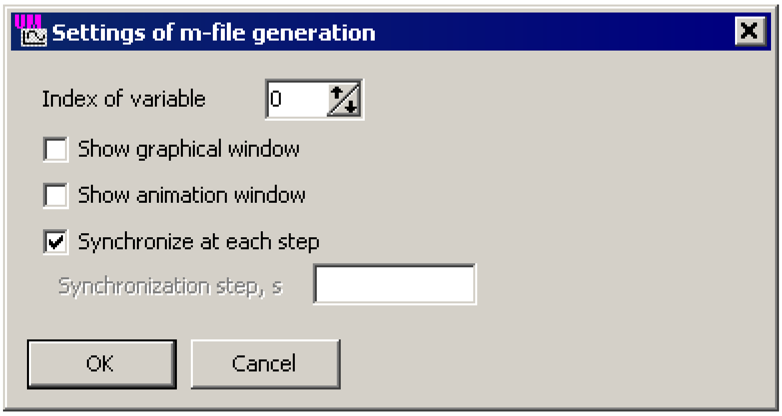

Figure 19.3 Settings of m-file generation

Every model should have the unique global variable in Matlab terms, see *h1* variable in *m-file* code in Sect. 19.2.1. *"Initialization"*, p. 19-8. All global variables are named as *h[Index of variable]*. So the user has to provide uniqueness of names of global variables in all *m-files* used in the same Matlab/Simulink model. If you use the only UM model in Matlab/Simulink model you need not to care about that. So you have to set **Index of variable** to **1** for the first mode, to **2** for the second one and so on. The only important issue is providing the uniqueness of names of Matlab/Simulink global variables. After setting the unique index, you have to save *.cosim* file and regenerate *m-file* with also unique names.

19.1.3. Settings of m-file generation

The dialog window with settings of *m-file* generation is shown in Figure 19.3. Let us consider these settings in details.

Index of variable

This setting is used if one Matlab/Simulink model includes several UM models, see Sect. 19.1.2. *"Including several UM models into Matlab/Simulink model"*, p. 19-5 for details.

Show graphical (animation) window

Turning on these flags leads to including special code into *m-file* that will show graphical or animation UM windows under Matlab/Simulink environment. It will give you a possibility to visualize UM model and processes. It might be extremely useful especially on the stage of testing models and their interaction.

Synchronize at each step

Turned on **Synchronize at each step** flag means that data exchange between Matlab/Simulink and UM will be initiated at each step of numerical method of Matlab/Simulink. Turn off the flag to set a value to **Synchronization step**, in fact the data exchange step between Matlab/Simulink and UM.

In some cases, when the frequency of processes in a Matlab/Simulink model is much higher than the frequencies of the mechanical part, it is recommended to carry out data exchange with a step-size larger than the integration step-size, which is set (or selected automatically) in Matlab/Simulink environment. In such cases, it makes sense to integrate high-frequency (for example in electrical part) and low-frequency (in mechanical part) processes with different step-size. This significantly reduces the computational efforts and accordingly accelerates the process of numerical simulation.

A typical example of such a system is a model of a control system of an asynchronous motor described in Matlab/Simulink and connected with a mechanical system modeled in UM. The frequencies of an electrical part (PWM) in this case would be several orders of magnitude higher than the highest frequency of mechanical part. In particular, this means that the step of modeling the electrical and mechanical parts will be different about the same proportion.

In practice, using rational **synchronization step** usually speeds up the simulation of such models by several orders almost without any significant errors in simulation results. For most practical purposes, the recommended interval of the synchronization step is [0.0001; 0.01]. However, prior to using this approach it is strongly recommended firstly to check its influence on the simulation results. To do this, carry out a few test calculations with different **synchronization step**, starting with the **synchronization step** close to the step model in Matlab / Simulink, with a gradual increase the step, and thus choose the optimal step synchronization step from accuracy / simulation time point of view.

19.2. Structure of m-file

Here you can see syntax of *m-file*:

[sys, x0, str, ts]=f(t, x, u, flag, p1, p2, ...), where

f is the *S-Function* name;

t is the current (simulation) time;

x is the vector of state variables;

u is the vector of input signals of the S-Function;

flag is the variable that dispatches the current stage of the simulation processes (initialization, calculation of output values or termination);

p_1, p_2 are S-Function parameters.

During the simulation process Matlab/Simulink calls *f* function one time on the initialization and termination stages and repeatedly on each time-step for calculation of output values.

Let us consider the content of *m-file* for every stage:

- initialization **(mdlInitializeSizes)**;
- calculation of output values **(mdlOutputs)**;
- termination **(mdlTerminate)**.

19.2.1. Initialization

When *flag* is equal to 0 the initialization **mdlInitializeSizes** function is called. Here the basic characteristics of *S-Function* like number of input and output signals are described. Below you can see the source code of the initialization procedure.

```
function [sys,x0,str,ts] = mdlInitializeSizes()
sizes = simsizes;
sizes.NumOutputs=1;
sizes.NumInputs=1;
sys = simsizes(sizes);
x0 = []; % No continuous states
str = []; % No state ordering
ts = [0]; % Inherited sample time
global h1
h1=actxserver('UMCosimulation.UMMatlab');
h1.LoadObjectFromFile('C:\UM\samples\tutorial\inv_pend_cosim \input.dat');
h1.LoadMatlabSettings('C:\UM\samples\tutorial\inv_pend_cosim
\inv_pend_cosim.cosim');
h1.ReadTotalConfiguration('C:\UM\samples\tutorial\inv_pend_cosim\inv_pend_cos
im');
h1.PrepareIntegration();
% End of mdlInitializeSizes.
```

Please note that the simulation of dynamics of UM models is supported by UM COM server that includes UM mathematical core. At the initialization stage, an instance of the COM server is created and some of its methods are called.

Creating instance of COM server

```
h1=actxserver('UMCosimulation.UMMatlab');
```

Loading the model

```
h1.LoadObjectFromFile('C:\UM\samples\tutorial\inv_pend_cosim\input.dat');
```

Loading settings

Besides UM model itself it is necessary to load UM-Matlab/Simulink settings that are saved as *.cosim* file and all UM model settings from *.par*, *.sim*, *.xv*, *.icf* files with the same name.

```
h1.LoadMatlabSettings('C:\UM\samples\tutorial\inv_pend_cosim
\inv_pend_cosim.cosim');
h1.ReadTotalConfiguration('C:\UM\samples\tutorial\inv_pend_cosim\inv_pend_cos
im');
```

Initializing the numerical methods

```
h1.PrepareIntegration();
```

The initialization stage is finished.

19.2.2. Calculation of output values

When *flag* is equal to 3 the **mdlOutputs** is called. This function provides transferring input signals to a UM model, execution of one integration step of a UM model under UM COM environment

```
function sys = mdlOutputs(t,x,u)
global h1
value=[u(1) 0.0];
h1.SetValues(value);
h1.DoIntegrationInterval(t);
sys=h1.GetValues();
% End of mdlOutputs.
```

Transferring input signals to UM model

h1.SetValues(value);

Here you can see the trick that is used to pass the only parameter to a UM model as a vector. The second empty (0.0) value is formally passed but not used. It is used just to fulfill formal syntax.

Integration step

h1.DoIntegrationInterval(t);

Transferring output signals from UM model

sys=h1.GetValues();

19.2.3. Termination

Subroutine **mdlTerminate** is called when flag is equal 9 and finalize simulation process in UM COM server and remove it from PC memory.

```
function sys = mdlTerminate(t,x,u)
global h1
h1.FinishIntegration();
h1.delete;
sys=[];
% End of mdlTerminate.
```

Finalization of integration process

```
h1.FinishIntegration();
```

Destroying the COM server

```
h1.delete;
```

19.3. Work under Matlab/Simulink environment

19.3.1. Matlab/Simulink configuration parameters

It is strongly recommended to turn off **Preferences | Simulink | Launch Simulink Preferences | Optimization | Implement logic signals as boolean data (vs. double)** flag, see Figure 19.4. Besides that, it also ensures compatibility with models created by earlier versions of Simulink software.

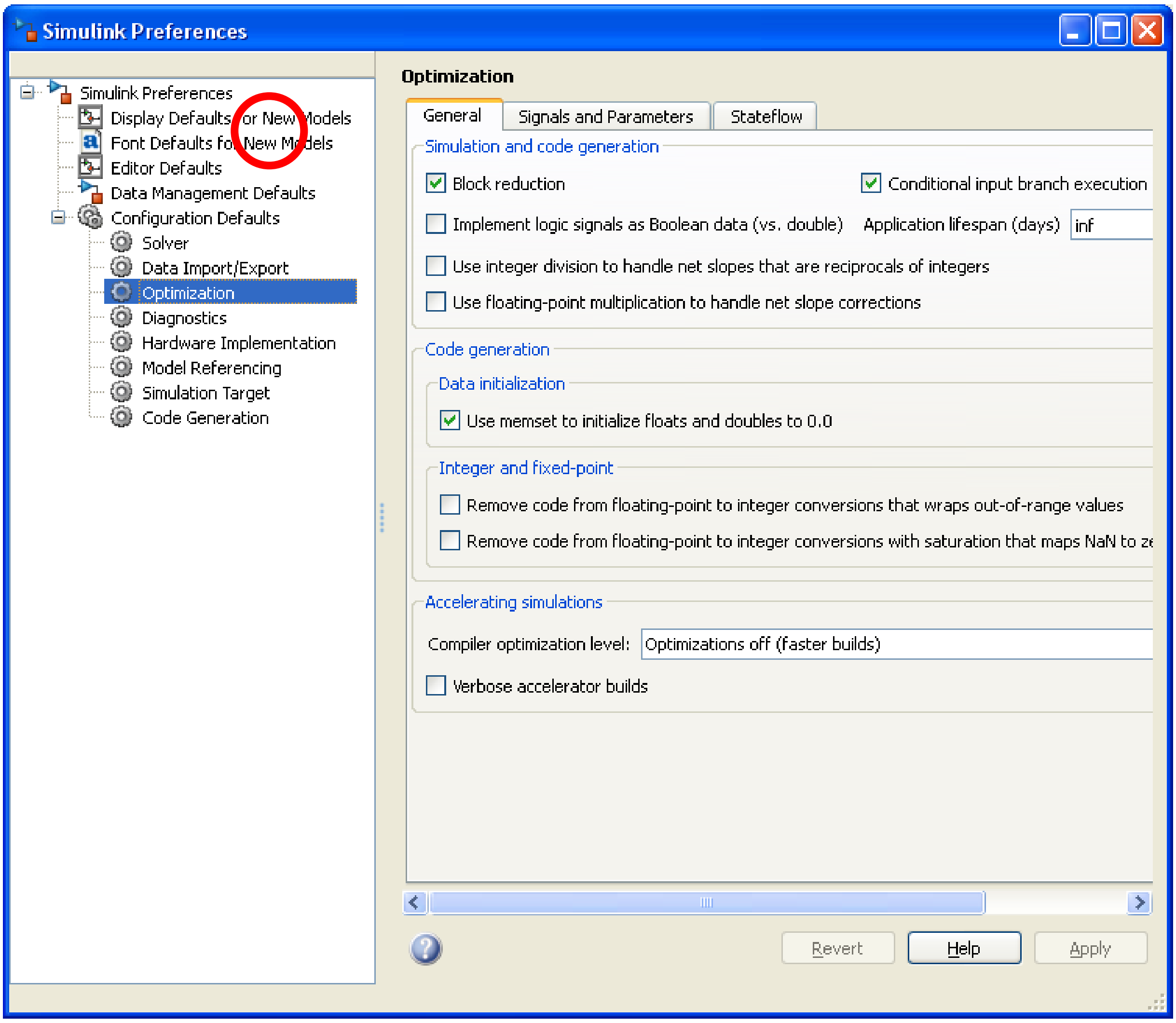

Figure 19.4. Matlab/Simulink configuration parameters

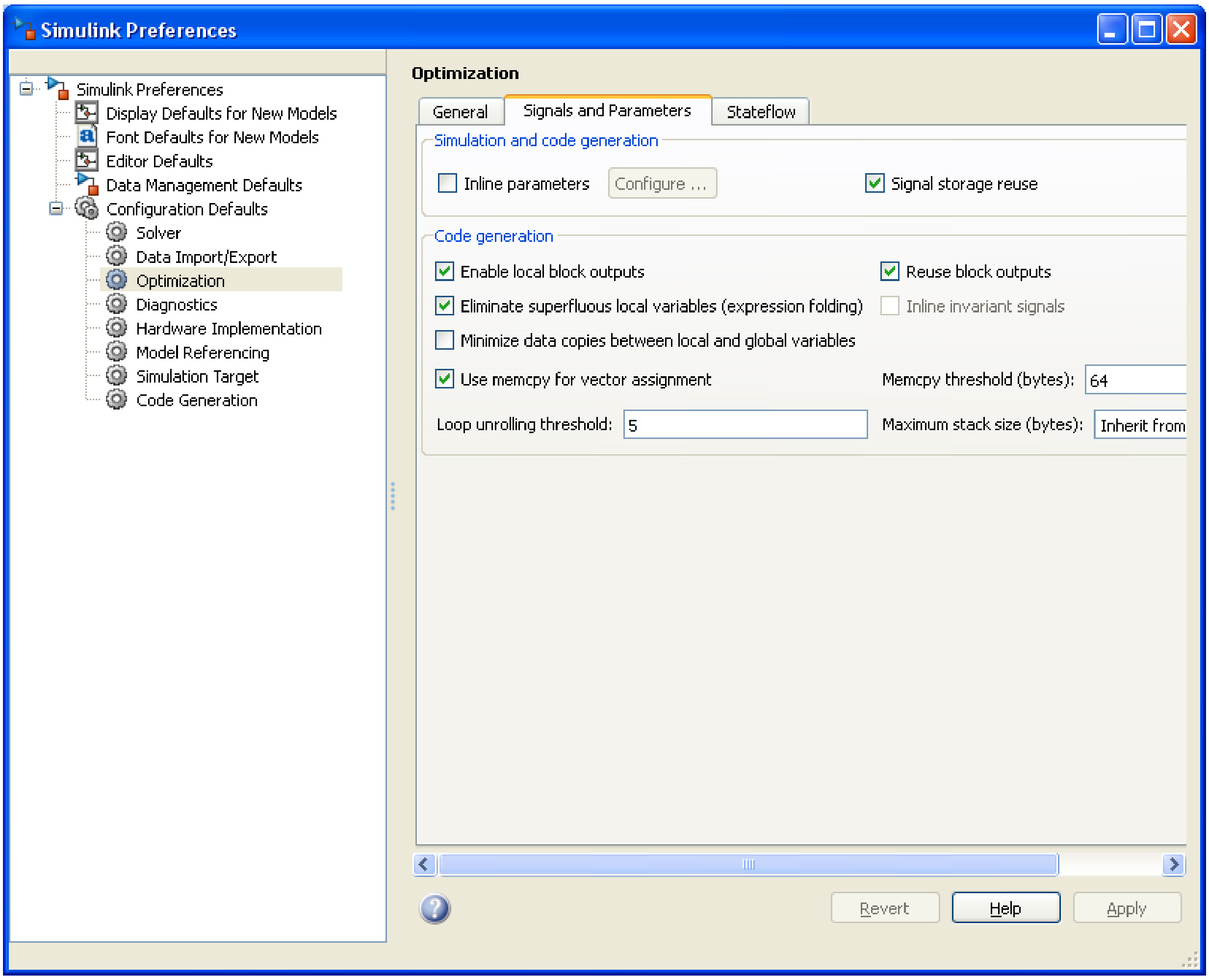

Figure 19.5. Matlab/Simulink configuration parameters

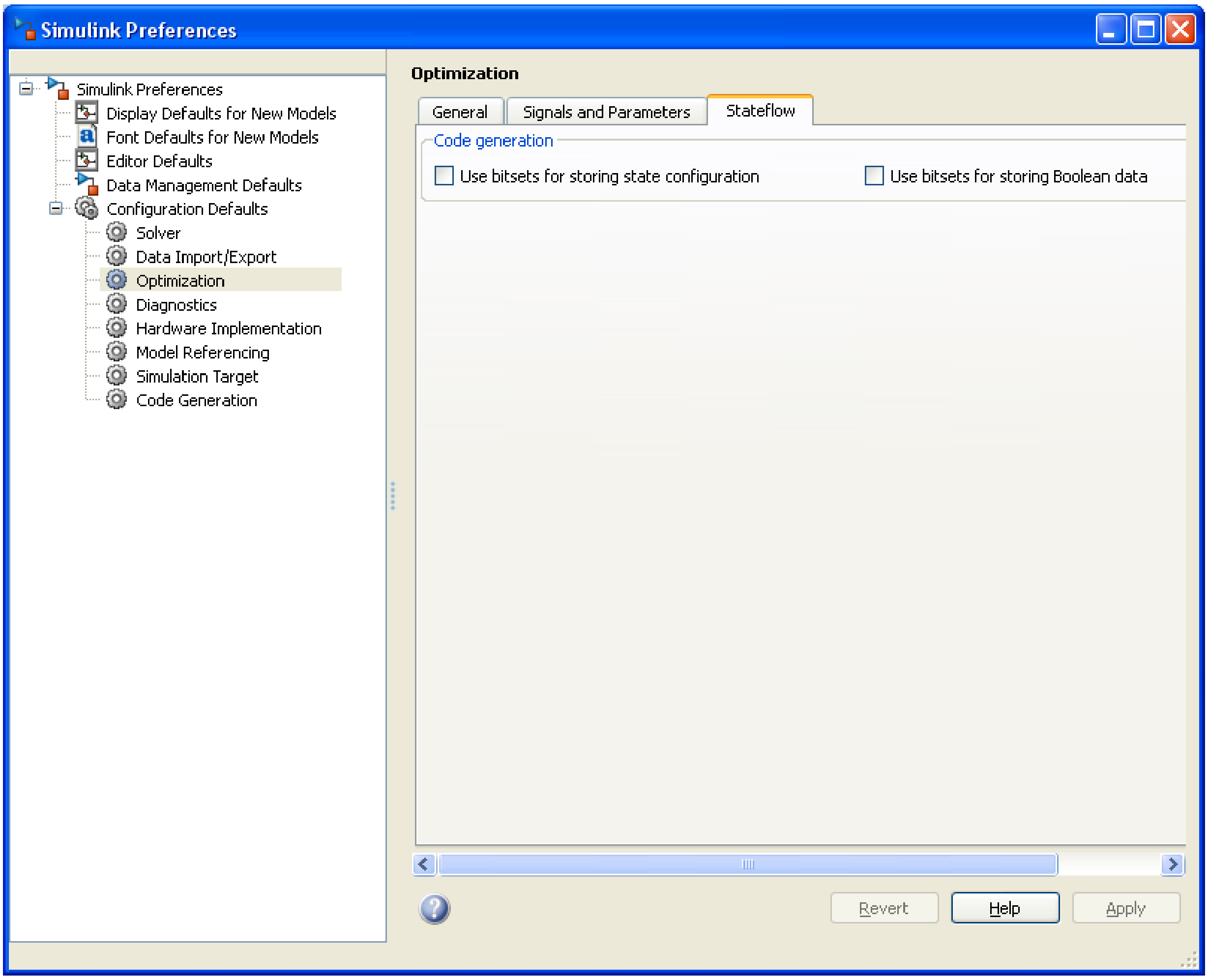

Figure 19.6. Matlab/Simulink configuration parameters

19.3.2. Initialization of parameters of S-Function

The connection between *S-Function* parameters and UM model parameters is also established with the help of **Wizard of export to Matlab/Simulink**. They are indicated in the **Parameters** list. After the generation of source code of *m-file* the list of S-Function parameters are added to formal parameters.

[sys, x0, str, ts]=f(t, x, u, flag, p1, p2, ...), where

p_1, p_2 are *S-Function* parameters.

Initialization of actual values of parameters takes place in the **Function Block Parameters: S-Function** dialog box, see Figure 19.7. Input actual values in the **S-function parameters** box in the same order how they are listed in the **Wizard of export to Matlab/Simulink**. You can input as actual parameters both numbers and declared variables from Matlab workspace.

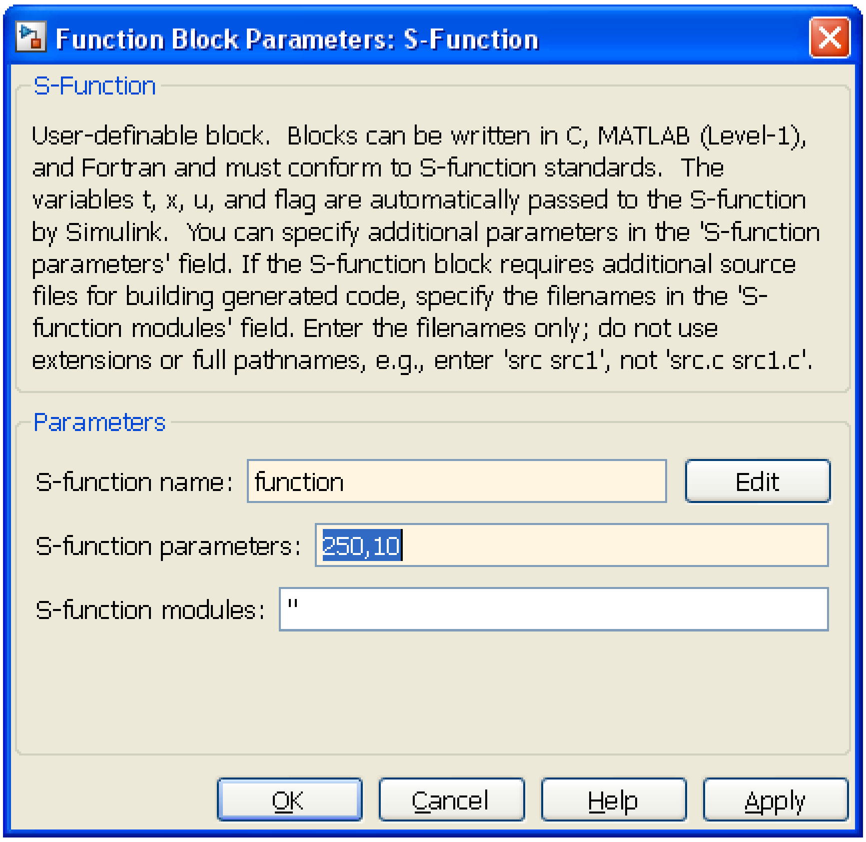

Figure 19.7. S-Function parameters

19.3.3. S-Function with several inputs and outputs

S-Function block has the only input and output port. If it is supposed that *S-Function* should have several input and output signals they should be combined with the help of **Mux** (for input signals) and **Demux** (for output signals) blocks like it is shown in Figure 19.8. You can find that blocks in the **Simulink Commonly Used Blocks** library.

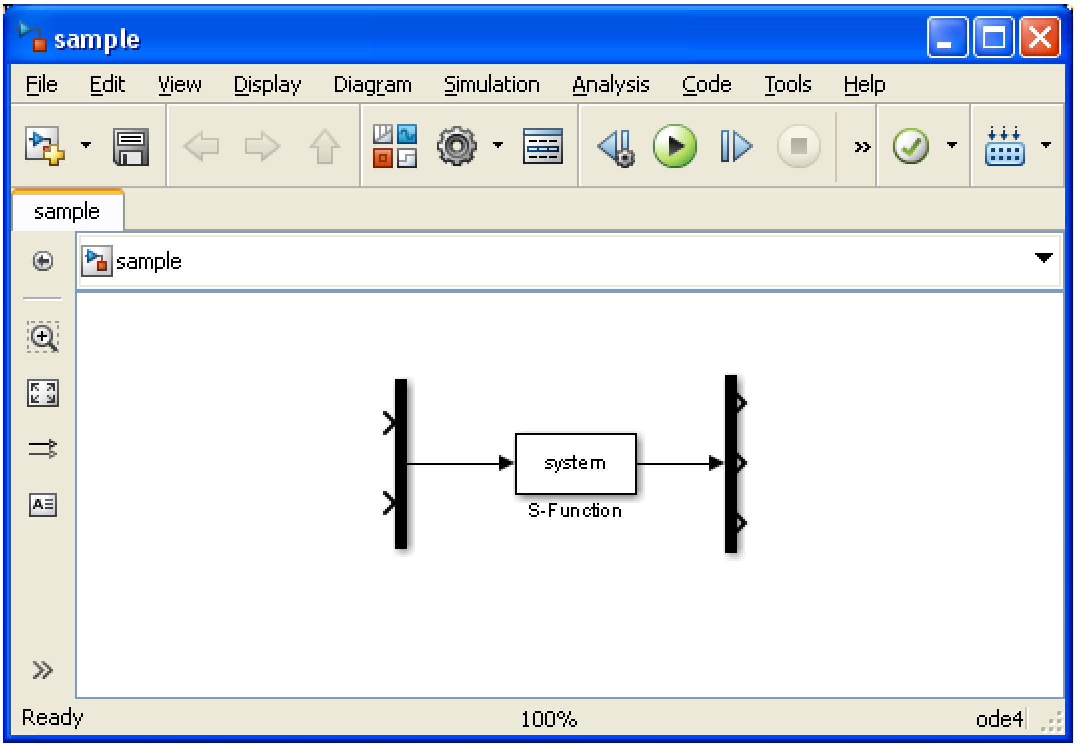

Figure 19.8. Mux and Demux blocks

Mux and **Demux** blocks are connected with input and output signals correspondingly. The number of signals for **Mux** and **Demux** blocks are specified in **Mux** and **Demux Block Parameters**, see Figure 19.9, Figure 19.10.

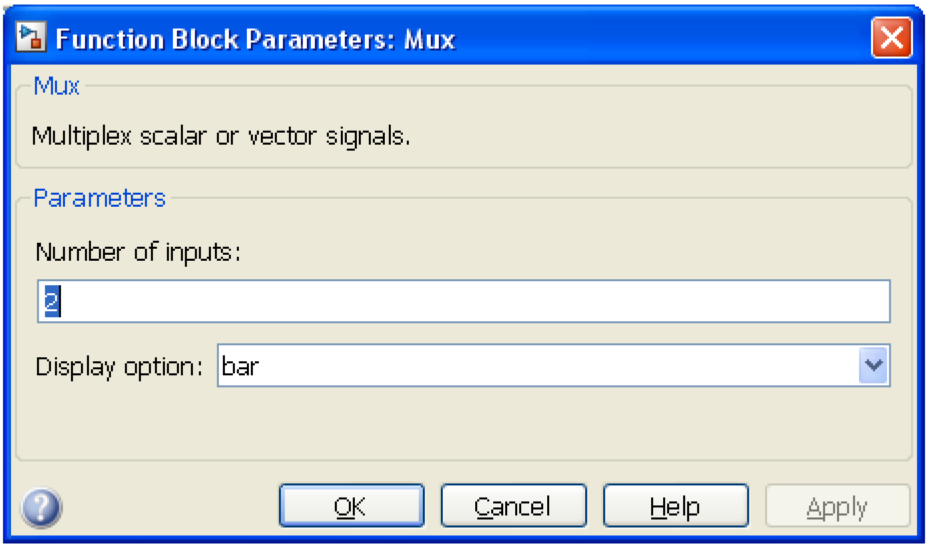

Figure 19.9. Mux parameters

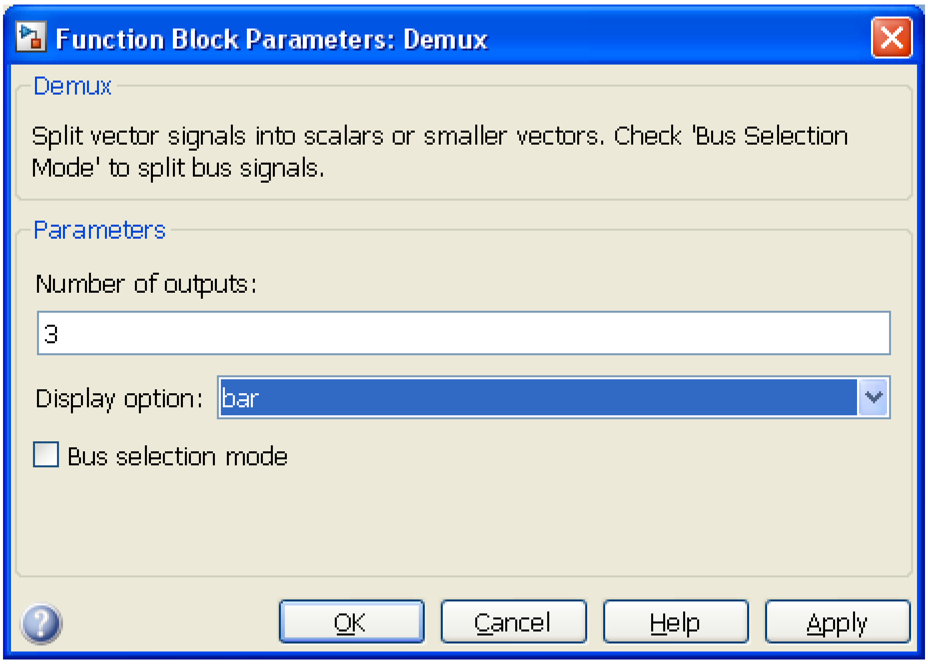

Figure 19.10. Demux parameters

19.4. UM model parameters

Saving all settings of UM model (initial conditions, parameters of integration method, identifier values, railway, road and trucked vehicle configuration) are fulfilled along with saving *m-file*. All files are saved automatically with the *m-file* name and specific extension (*.par*, *.icf*, *.car*, etc.). So you need to determine all settings prior to generating *m-file*.

Loading settings of UM model under Matlab/Simulink environment is fulfilled with the help of the **ReadTotalConfiguration** method of UM COM server, see Sect. 19.2.1. *"Initialization"*, p. 19-3 for details.

Let us consider the sequence of assignment of values of UM model parameters during all stages under Matlab/Simulink environment.

1. Loading values of parameters from *.par* file that were saved along with the generation of *m-file* at the initialization stage, see Sect. 19.2.1. *"Initialization"*, p. 19-8.
2. Then the actual *S-Function* parameters are assigned to connected UM model parameters, see Sect. 19.3.1. *"Matlab/Simulink configuration parameters"*, p. 19-12. Thereby values of those parameters are reassigned.
3. Finally new values of UM model parameters that are connected with input signals of *S-Function* are assigned on each step of numerical method.

19.5. Model portability

Please note that procedure for the automatic generation of *m-file* includes direct paths to files, see Sect. 19.1.1. *"Wizard of export to Matlab/Simulink"*, p. 19-4. That is why after moving or copying UM model to another directory, previously generated *m-file* will contain incorrect paths and it is necessary to regenerate *m-file*.

To regenerate *m-file* you need to load a UM model in **UM Simulation** program, open **Wizard of export to Matlab/Simulink**, load *.cosim* file and click . Then the *m-file* will be generated and correct current paths will be used.

Please also note that during the generation of *m-file* all files with UM settings (initial conditions, parameters of integration method, identifier values, railway, road and trucked vehicle configuration) will be rewritten. So prior to regenerating *m-file* you need to set up or load all model settings, see Sect. 19.4. *"UM model parameters"*, p. 19-18 for more details.