



Forschungs- und Entwicklungs GmbH - Research Center Wels - Stelzhamerstraße 23 - A-4600 Wels

Documentation

ModOPS - Scilab/XCos Model Library

Erstellt von:

FH-Wels

Contents

1. General Remarks	3
1.1. Homepage and Live-DVD	3
1.2. Installation Instructions (Windows OS)	3
2. Overview on Tool-chain	3
3. Introduction of SCILAB and XCOS	4
3.1. General	4
3.2. ATOMS	7
3.2.1. COSELICA	7
3.2.1.1. COSELICA functionality	8
3.3. HeuristicLab	9
4. Position servo	11
4.1. Model description	11
4.2. Simulation of the system	12
4.3. Identification of system	14
4.3.1. Recursive Least Square Algorithm	14
4.3.2. HeuristicLab (HL)	16
4.4. Regulation of system	19
5. Position servo with rod pendulum	20
5.1. Model description	20
5.2. Simulation of the system	22
5.3. Identification of system	23
5.3.1. Recursive Least Square Algorithm	23
5.3.2. HeuristicLab (HL)	25
5.4. Regulation of system	26
6. Two mass oscillator	27
6.1. Model description	27
6.2. Simulation of the system	28
6.3. Identification of system	30
6.3.1. Recursive Least Square Algorithm	30
6.3.2. HeuristicLab (HL)	33
6.4. Regulation of system	33
7. Two mass oscillator with rod pendulum	34
7.1. Model description	34
7.2. Simulation of the system	36

7.3. Regulation of system	36
-------------------------------------	----

1. General Remarks

1.1. Homepage and Live-DVD

The software presented here is provided for two different operating systems. On the one hand there exists a Windows-Version. Due to licensing reasons, this version cannot be offered as a Live-DVD and is only listed in this document. On the other hand, there exists a Linux-Live-DVD with limited functionality.

1.2. Installation Instructions (Windows OS)

If a Windows OS is used, the following tools must be installed:

- ▷ Scilab 5.4.1 (32bit or 64bit): <http://www.scilab.org/download/5.4.1>
 - Coselica Toolbox (ATOMS)
 - Plotting Library (ATOMS)
- ▷ C-Compiler (Visual Studio 2008/2010/2012/2013 Professional)
- ▷ HeuristicLab 3.3.9: <https://dev.heuristiclab.com/download>

Open Scilab and type “haveacompile()” in the command line. If the return value equals “T”, Scilab has found a C-compiler. After this, the Coselica Toolbox has to be installed, see 3.2.

2. Overview on Tool-chain

Fig.2.1 shows the basic steps from system modelling up to the closed loop experiments. First, the system model has to be determined. This can be done by mathematical equations (ODEs,...) or by using the Modelica based XCos add-on Coselica. After the system behavior of the two modeling approaches has been verified, the system parameters can be identified with appropriate methods like Recursive-Least-Square algorithm or HeuristicLab. If the system parameters are known from identification, a suitable state feedback controller can be designed.

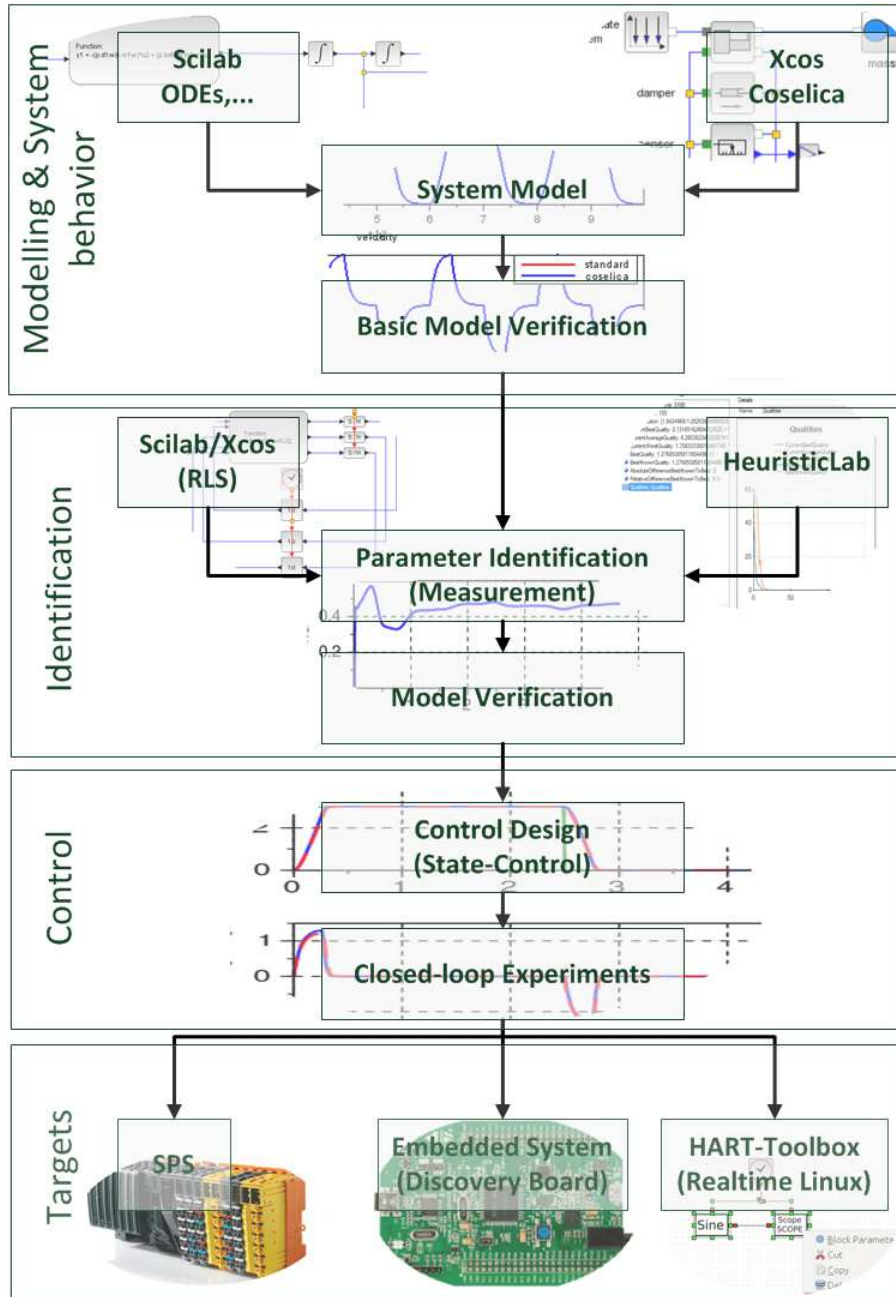


Figure 2.1: Tool-chain

3. Introduction of SCILAB and XCOS

3.1. General

SCILAB is an open source software for numerical computations. It is very similar to MATLAB. XCOS, which is part of SCILAB, is a graphical dynamical system modeler and simulator, and the counterpart of SIMULINK. A lot of MATLAB commands also work in SCILAB, some are only

named differently. Also, the graphical user interface (GUI) is very similar to MATLAB. The screenshot in Fig. 3.1 shows the GUI. On the left side, there is the “File Browser” which shows the current working directory. The window in the middle is the “Console” where the commands are executed, but it’s also possible to create a command script by clicking the note symbol on the left side of the toolbar. On the right side, there are the “Variable Browser”, which shows stored variables, and the “Command History”, which logs all the executed commands. The XCos and the “Palette Browser” of XCos are launched by clicking the screen symbol in the toolbar.

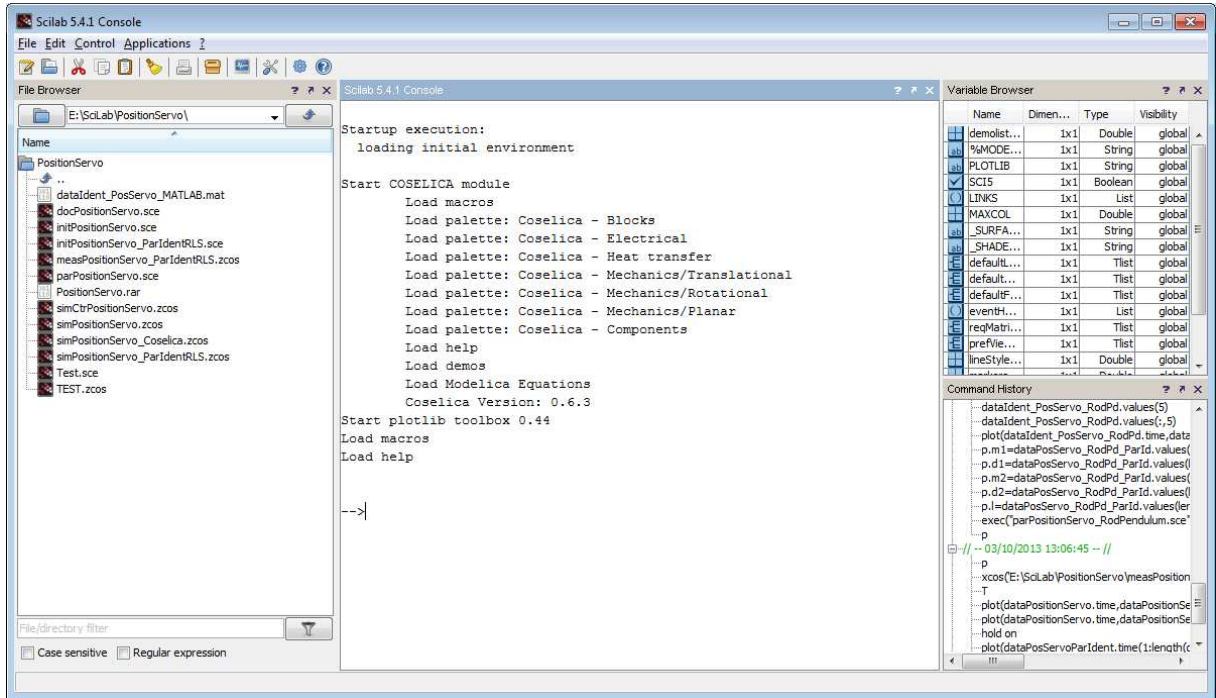


Figure 3.1: GUI of SCILAB

The palettes of the “Palette Browser”, shown in Fig. 3.2 are almost identical to those of SIMULINK. The blocks are inserted into the XCos diagram by drag and drop. The two most important buttons in the toolbar of XCos are shown in Fig. 3.3. The “play” symbol which executes the simulation and the red x-symbol which aborts it. The parameters of the simulation-solver can be edited by /Simulation/Setup in the menu bar.

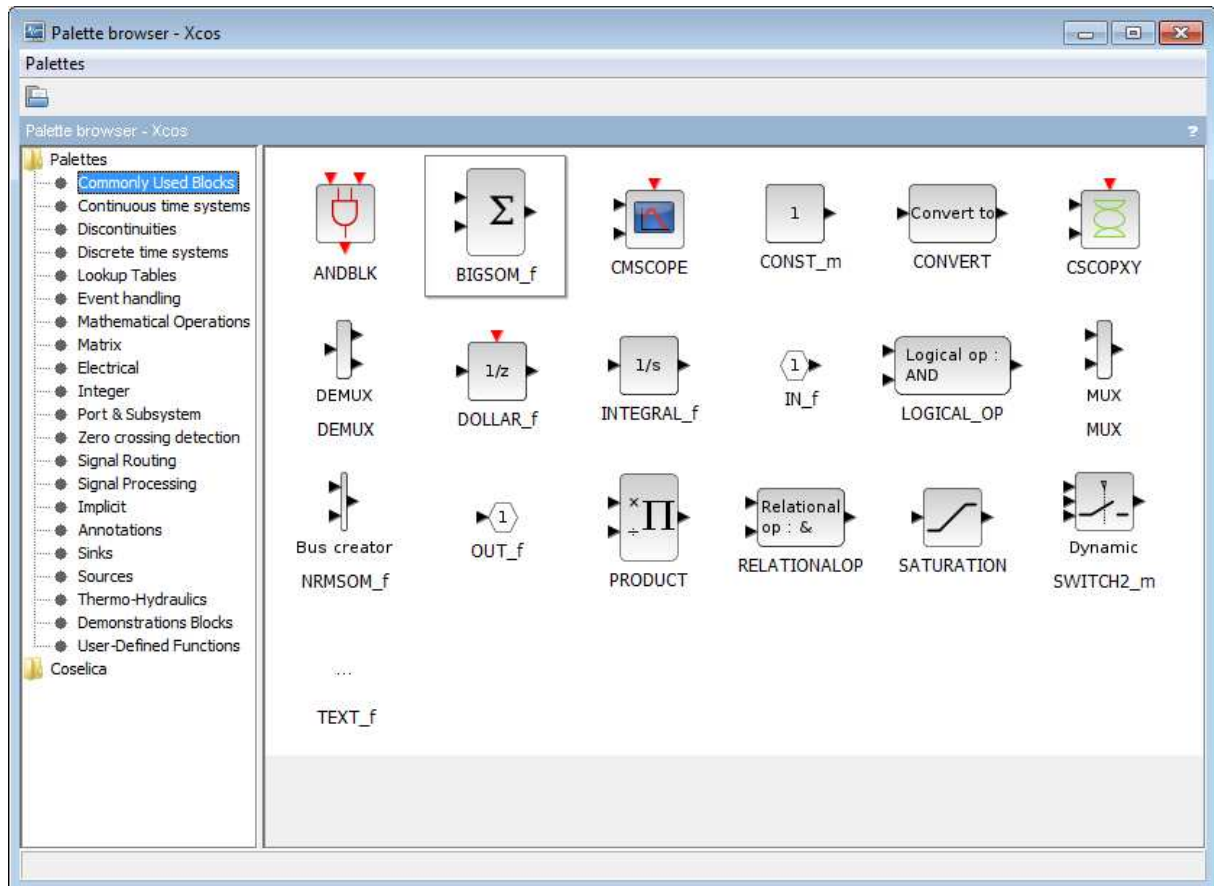


Figure 3.2: Palettes Browser

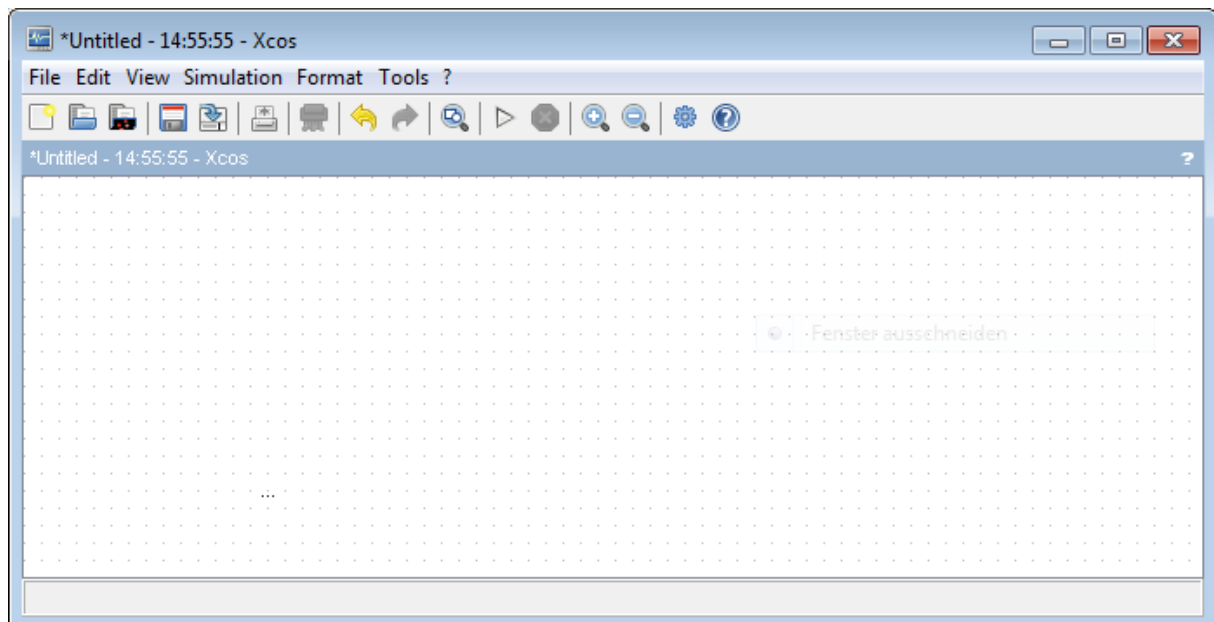


Figure 3.3: XCos window

3.2. ATOMS

To extend the functionality of Scilab, there are a number of additional packages available, which can be downloaded. These packages are available directly from the Scilab console via the AuTo-matic mOdules Management for Scilab, called ATOMS.

To search for add-ons, open the ATOMS catalog by clicking the cardboardbox-symbol in the main window of SCILAB, see Fig. 3.4. To get full access to all examples in this guide, install the toolboxes “Coselica” and “Plotting library”. The Coselica toolbox can be found in the category XCos and the “Plotting library” is located in Graphics. “Plotting library” is supposed to help you make plots as if you were using Matlab. The Coselica toolbox is based on the modeling language Modelica and provides some basic mechanical, electrical and thermodynamical modeling and simulation blocks.

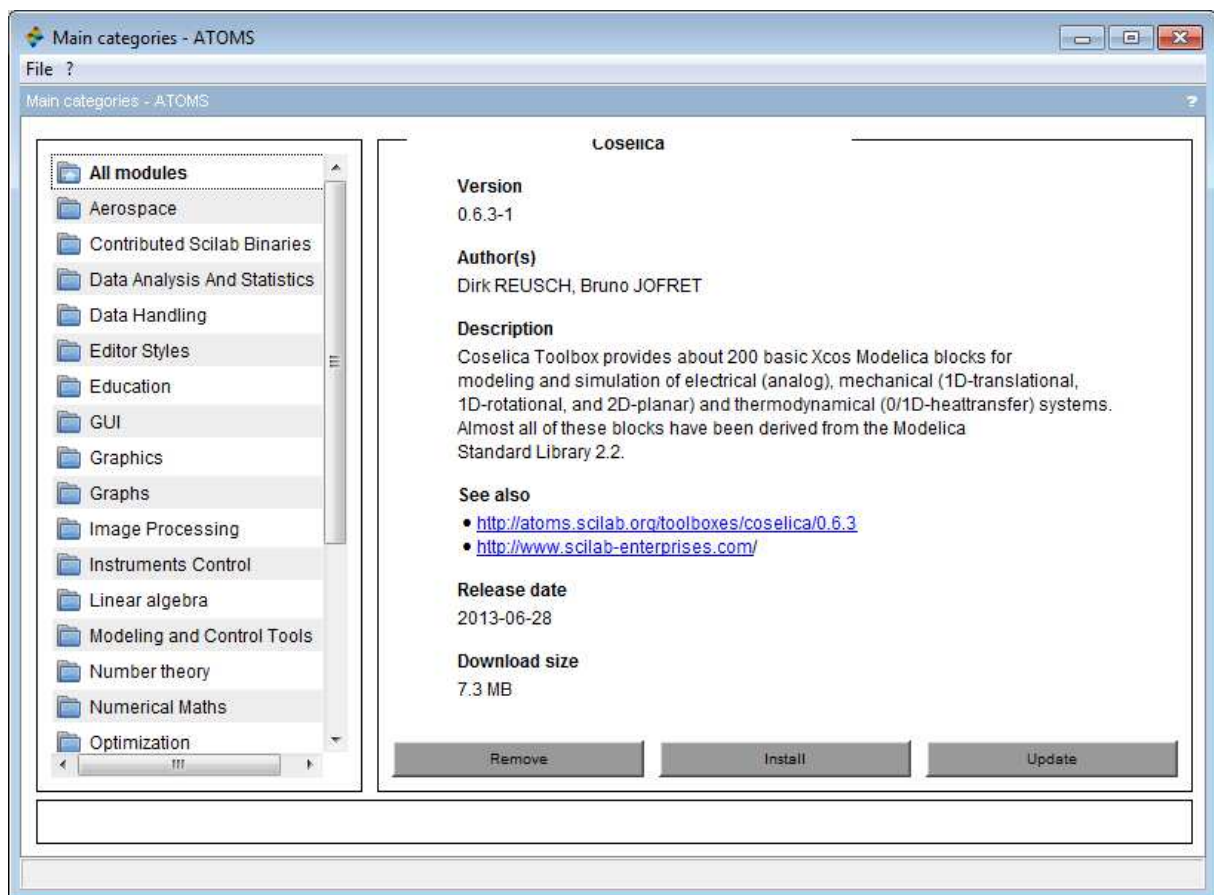


Figure 3.4: ATOMS catalog

3.2.1. COSELICA

After you have installed Coselica in the ATOMS catalog, an additional palette appears in the “Palettes Browser” of XCos. For the examples in this guide only COSELICA blocks from “Blocks” and “Mechanics” are used. The folder “Blocks” provides some essential blocks like signal sources,

math operators, interfaces (data type conversion) etc. The folder “Mechanics” provides blocks for translational, rotational and planar movements of mechanical systems. Some of the translational blocks are also used for planar simulations.

3.2.1.1. Coselica functionality COSELICA uses a very intuitive way of describing physical systems. The simulation models consist of basic physical elements. They can be combined and connected without substantial knowledge about mathematical background. SIMULINK and XCOS imply substantial mathematical knowledge about their physical backgrounds compared to COSELICA.

The demo examples in this guide are only mechanical systems, so the basic functionality of COSELICA will be shown by the mechanical model of a position servo. The screenshot in Fig. 3.5 shows the simulation model of a simple position servo (cart with force as input).

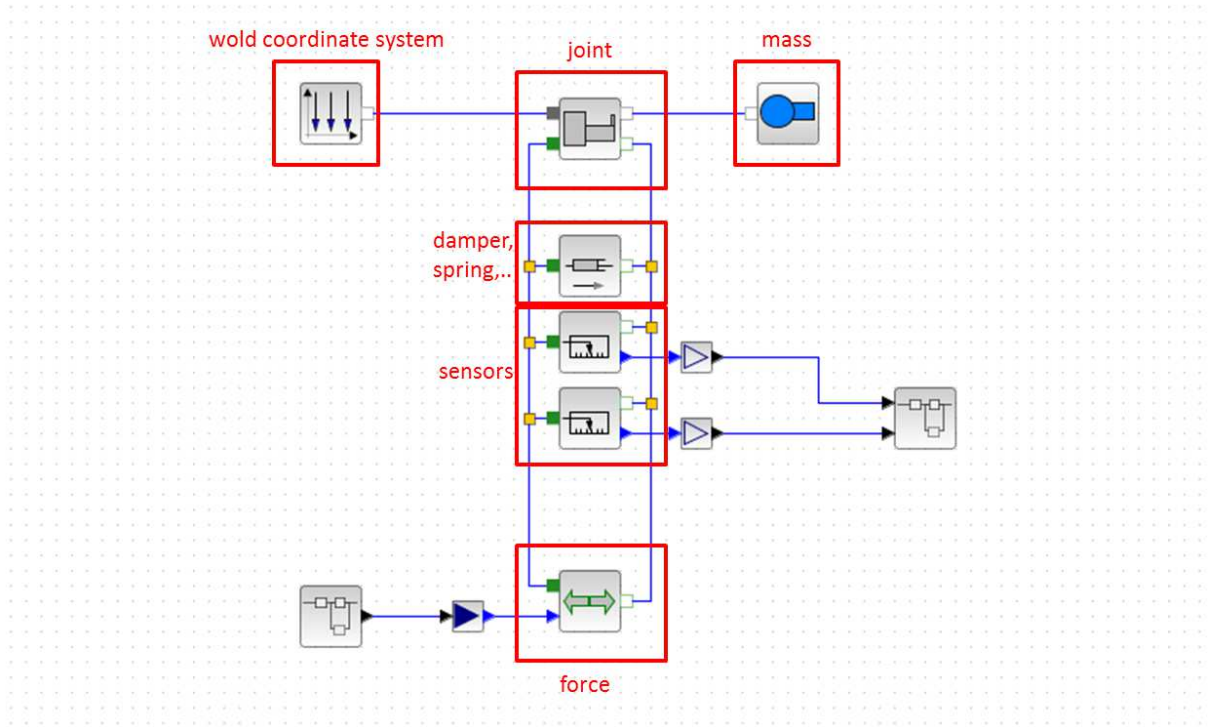


Figure 3.5: basic COSELICA elements shown at the example of a position servo

The basic block of all planar simulations is a joint. For different degrees of freedom there are different joints (translational, rotational). On one side, the joint is connected to a world coordinate system (or to other components) to define its reference position. Beyond, masses are connected to the joint. All forces that affect the joint are connected parallel to it. These forces can derive from dampers, springs or external forces for example. To measure the states of the model, sensor blocks are used. There are different sensor blocks for position, velocity and acceleration. To connect signals from normal XCOS blocks to Coselica blocks and vice versa, interface blocks are necessary. How to realise different mechanical systems in Coselica will be illustrated in the following examples.

3.3. HeuristicLab

HeuristicLab [1] is an open source framework for heuristic and evolutionary algorithms that is developed by members of the Heuristic and Evolutionary Algorithms Laboratory (HEAL) since 2002 and is freely available at <http://dev.heuristiclab.com/Download>. HeuristicLab is shipped with a wide variety of ready-to-use algorithms and problem definitions.



Figure 3.6: Sample screenshot of the HeuristicLab GUI

In the context of this project, HeuristicLab is used to identify the parameters of a simulation model, which is implemented in Scilab/Xcos. Therefore, a generic coupling between HeuristicLab and Scilab has been implemented that allows the execution of arbitrary Scilab scripts. When a parameter identification problem (Fig. 3.7) for simulation models should be solved, the script is responsible for executing the simulation model with suggested parameter values and calculating a fitness value. The fitness value expresses the accordance between the results of the simulation model with the currently used parameter values and the observed measurements in the real world. Most of the times the sum of the absolute errors at predefined time steps is calculated and used as the fitness value (cf. Fig. 3.7 last line of the evaluation script). In addition to the Scilab script, a few other parameters have to be adapted before the parameter identification problem can be solved. First of all, the problem dimension (e.g., how many parameters should be optimized) must be configured. The next step is the configuration of the names of the parameters (ParameterNames) and the name of the variable (QualityVariableName) which contains the quality value. Afterwards the Scilab script (ScilabEvaluationScript) for evaluation of a parameter vector has to be chosen and whether the quality should be maximized or not (Maximization); in the case of the sum of absolute errors as the fitness function, Maximization should be set to false.

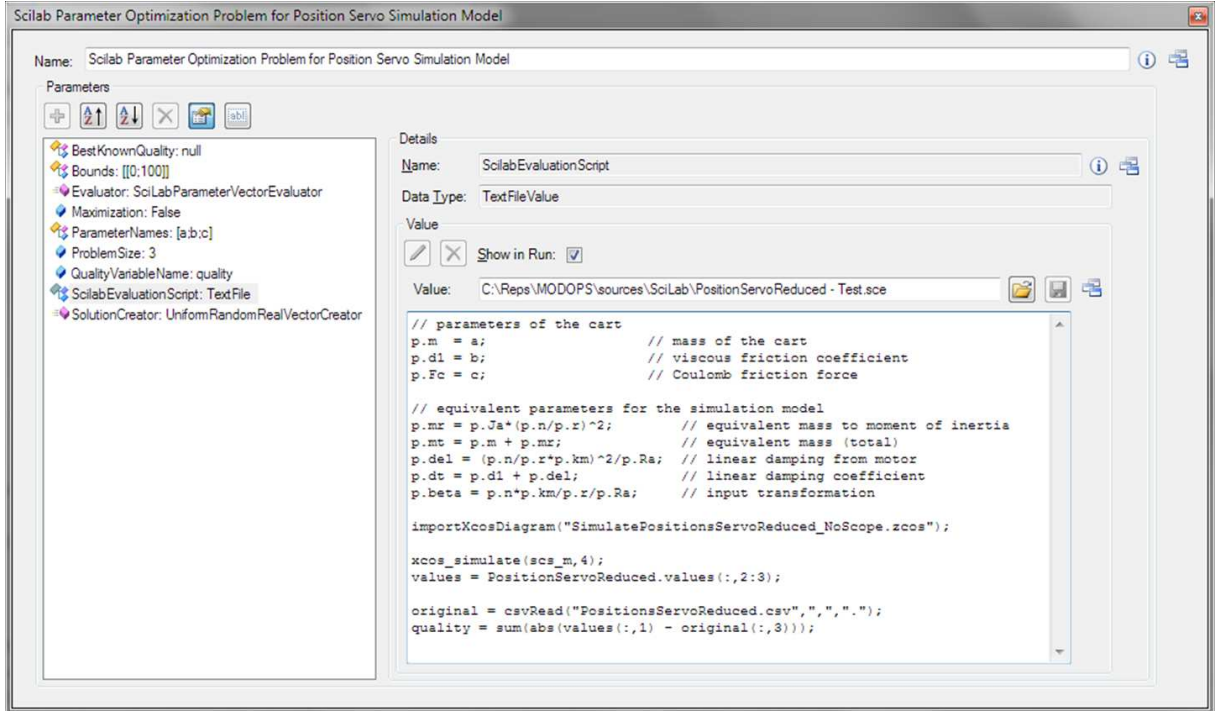


Figure 3.7: Parameter Identification Problem in HeuristicLab

After the problem configuration is finished every HeuristicLab algorithm that is able to deal with real vectors can be used to solve the problem (e.g., evolution strategies or genetic algorithm). However, several algorithm tests have shown that the covariance matrix adaption evolution strategy (CMA-ES) yields the best results for parameter identification problems of simulation models. The results of an exemplary run of the CMA-ES on a parameter identification problem are shown in Fig. 3.8. In the following sections a HeuristicLab file for every described simulation model can be found in the according folder. The file contains the preconfigured CMA-ES algorithm as well as the parameter identification problem (the evaluation script has to be set manually, before the algorithm can be executed) and the achieved results.

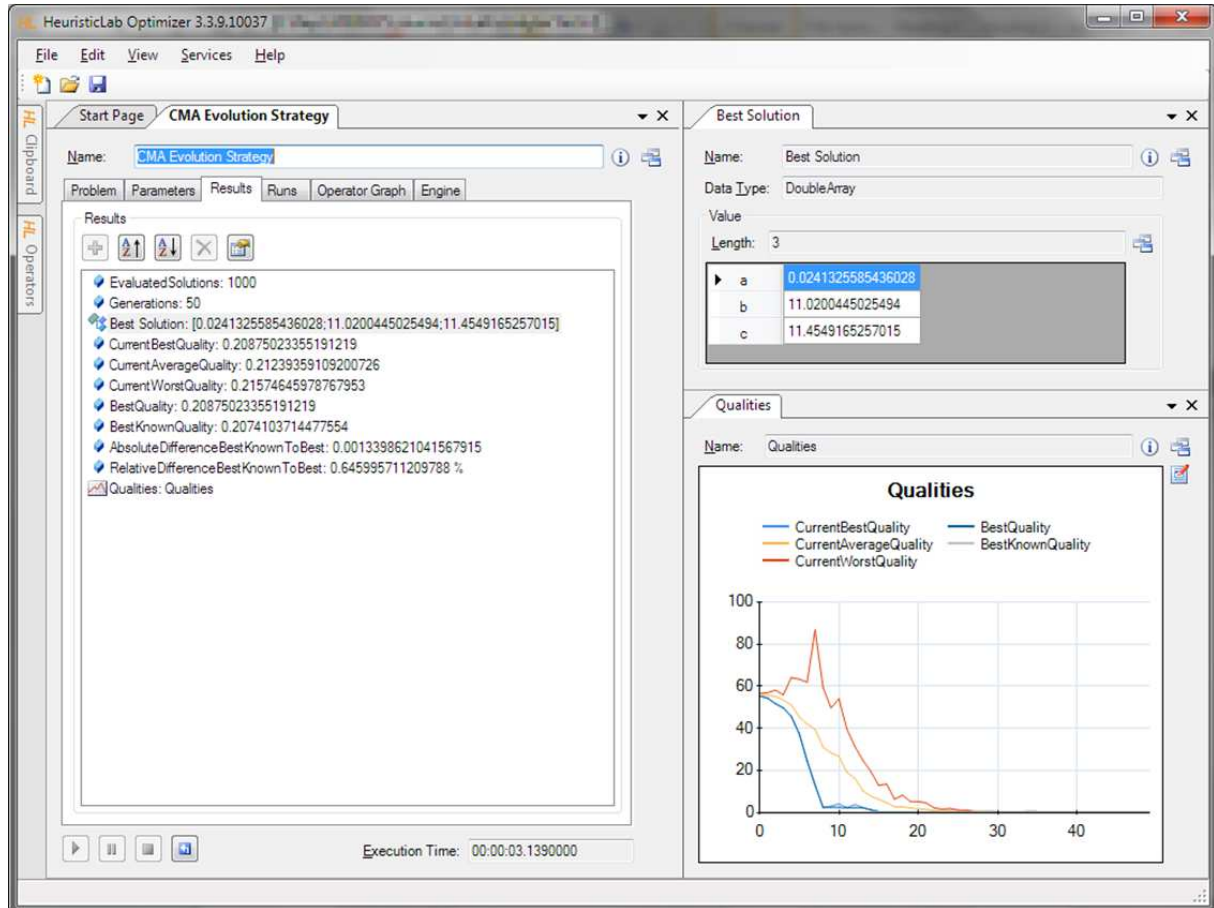


Figure 3.8: Results of an exemplary CMA-ES run on a parameter identification problem

4. Position servo

4.1. Model description

The position servo is basically a cart with one degree of freedom that is driven by a DC motor. The drawing in Fig. 4.1 shows the schematic structure of the model.

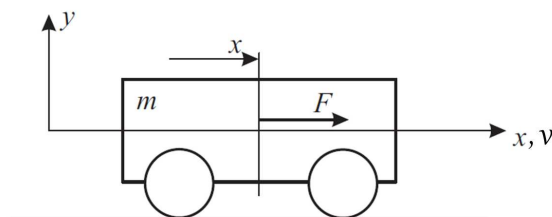


Figure 4.1: schematic of the position servo

The mathematical equations to describe this mechanical system are:

$$\dot{x} = v$$

$$\dot{v} = \frac{\tilde{d}_1}{\tilde{m}_1}v + \frac{\beta}{\tilde{m}_1}u_a$$

Name	Description
x	position of cart
v	velocity of cart
\tilde{m}_1	accumulated mass of cart
\tilde{d}_1	accumulated friction of cart
β	force factor of drive

Table 4.1: Abbreviations cart system

The dynamic of the electrical circuit is neglected because of its high dynamic compared to the mechanical part.

4.2. Simulation of the system

The XCos simulation file (“simPositionServo.zcos”) and the COSELICA simulation file (“simPositionServo_Coselica.zcos”) can be found in the folder “PositionServo\Simulation”. The screenshots in Fig. 4.2 and Fig. 4.3 show the XCos diagram and the COSELICA diagram of the position servo.

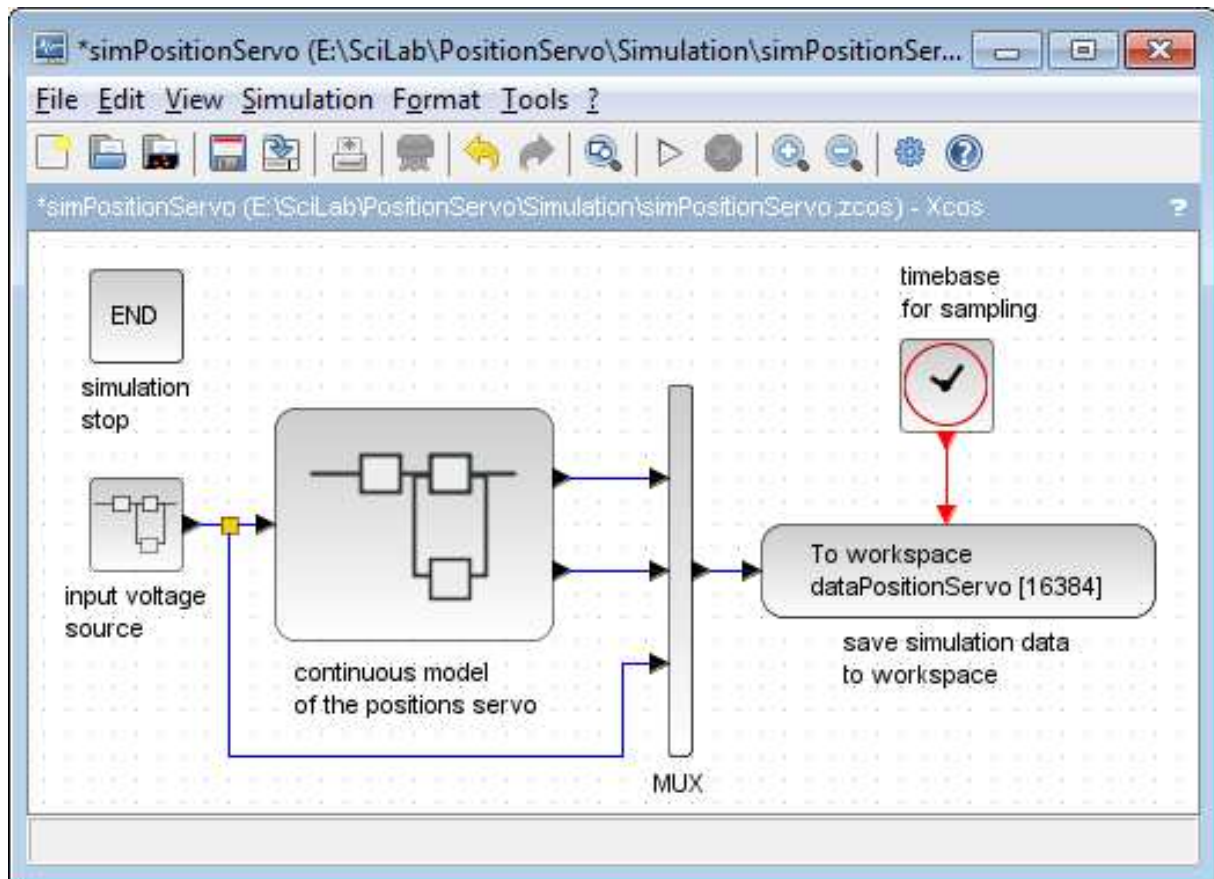


Figure 4.2: Simulation of the position servo with standard XCos blocks

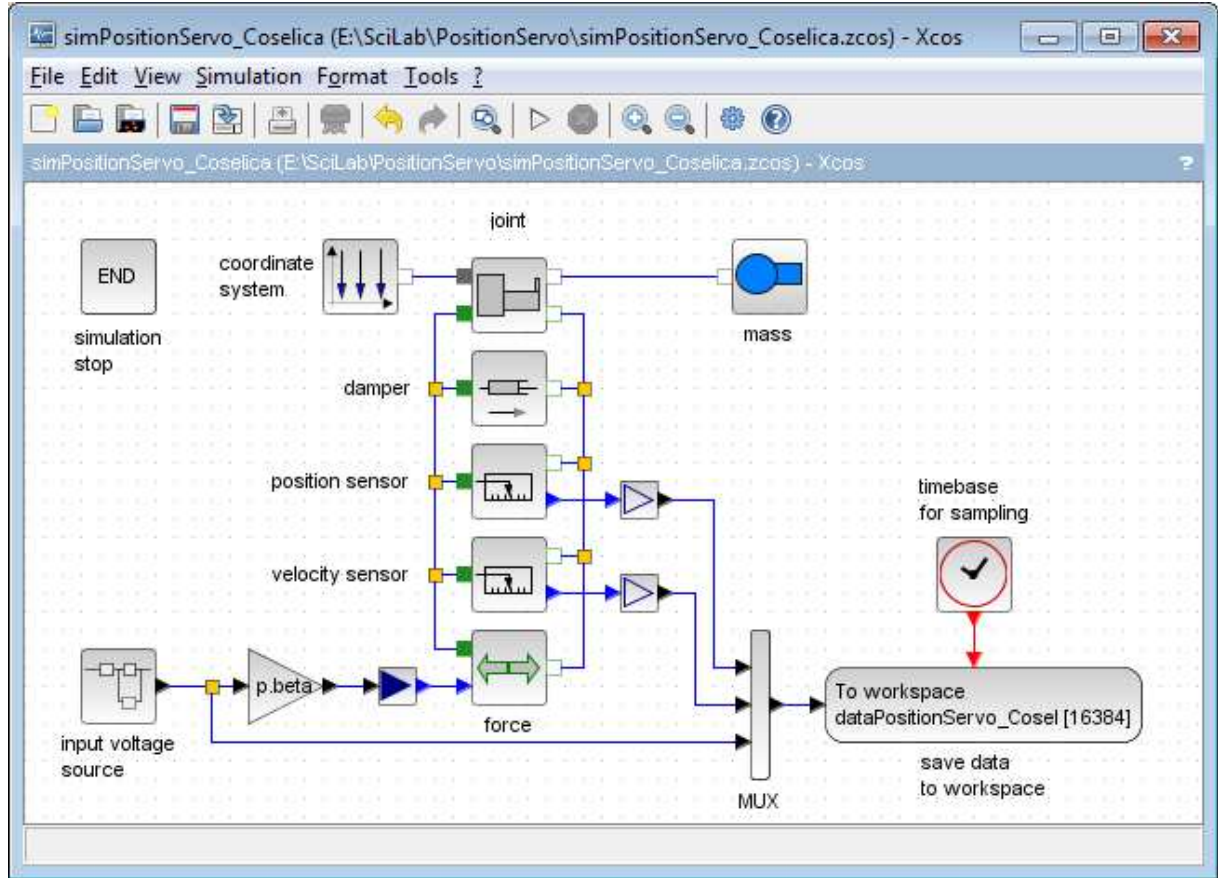


Figure 4.3: Simulation of the position servo with COSELICA blocks

The XCos simulation of the system in Fig. 4.2 is implemented by a so-called “SCILAB function block”, very similar to the MATLAB-function block in SIMULINK. The COSELICA simulation consists of basic mechanical components, as described in Chapter 3.2.1.

To start the simulation, execute the script file “initPositionServo”. This script initializes all necessary parameters and executes all simulations related to the position servo. To visualize the output of the simulation, execute the file “docPostionServo”.

4.3. Identification of system

Because the parameters of a mechanical system are often not known, a parameter identification, based on real measurements of the system, is necessary. One possible way of identifying a system is the RLS identification (recursive least square identification), which is used for this example.

4.3.1. Recursive Least Square Algorithm

This method assumes that the mathematical model of the system is known, furthermore the input signal and the output signal of a measurement are required.

The file “initPositionServo_ParIdentRLS.sce” in the folder \PositionServo\Identification initializes and executes the identification that is carried out by the identification program “measPositionServo_ParIdentRLS.zcos”. The screenshot in Fig. 4.4 shows the identification program. The RLS-subsystem is provided with the measurement data of the position servo and calculates the parameters of the system. Afterwards, the identification output is stored to the workspace.

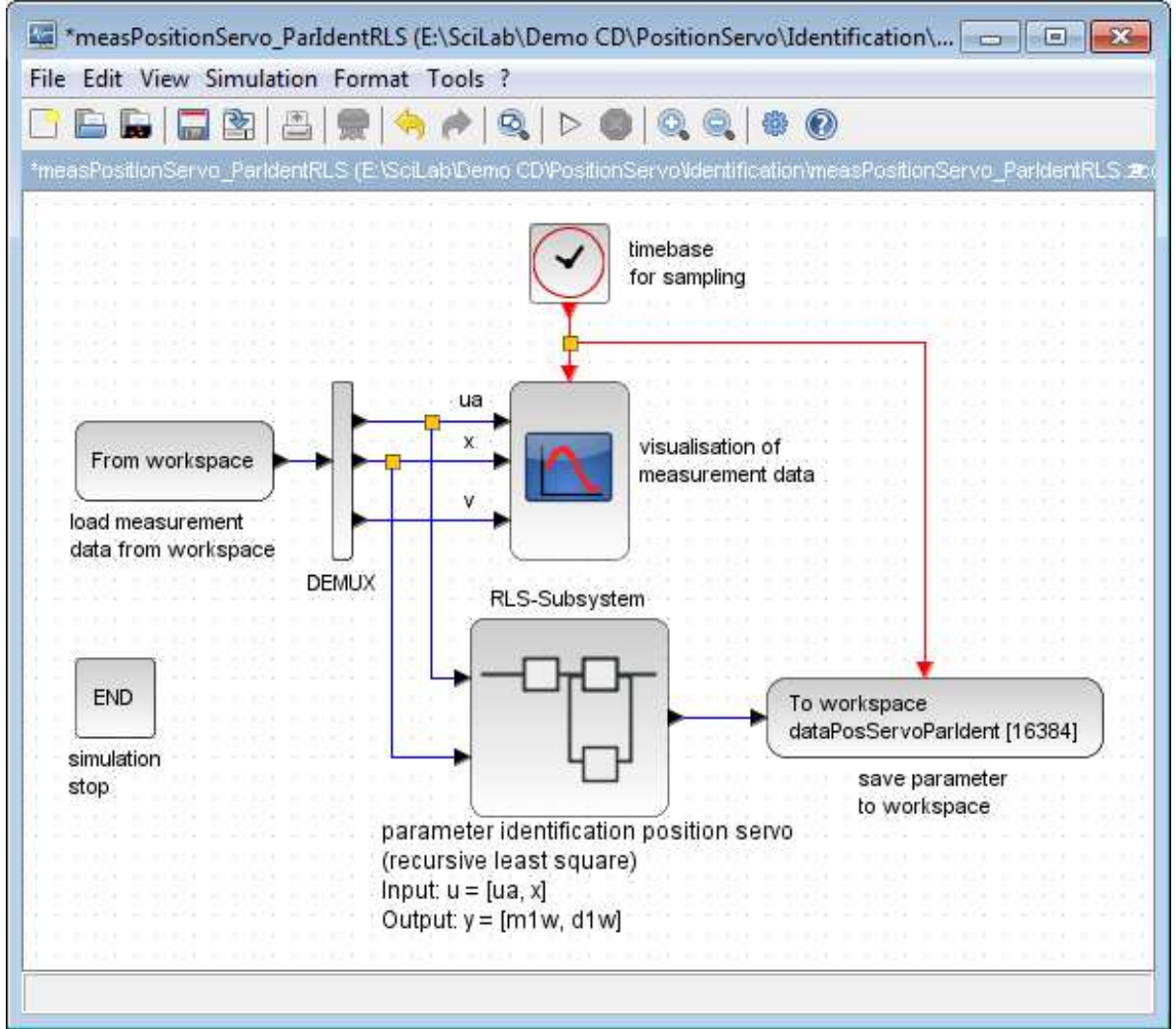


Figure 4.4: Identification program of position servo

The prepared measurement data of a real position servo is stored in “dataIdent_PosServo_MATLAB.mat”. By executing the file “docPositionServo_ParIdentRLS.sce” the identification will be visualized (Fig. 4.5). The graph shows the input signals (u_a and x) output signals (\tilde{m}_1 and \tilde{d}_1) of the RLS-block. The valid identification parameters are the stationary values of the parameter signal. They are stored in the structure pIdent at the workspace.

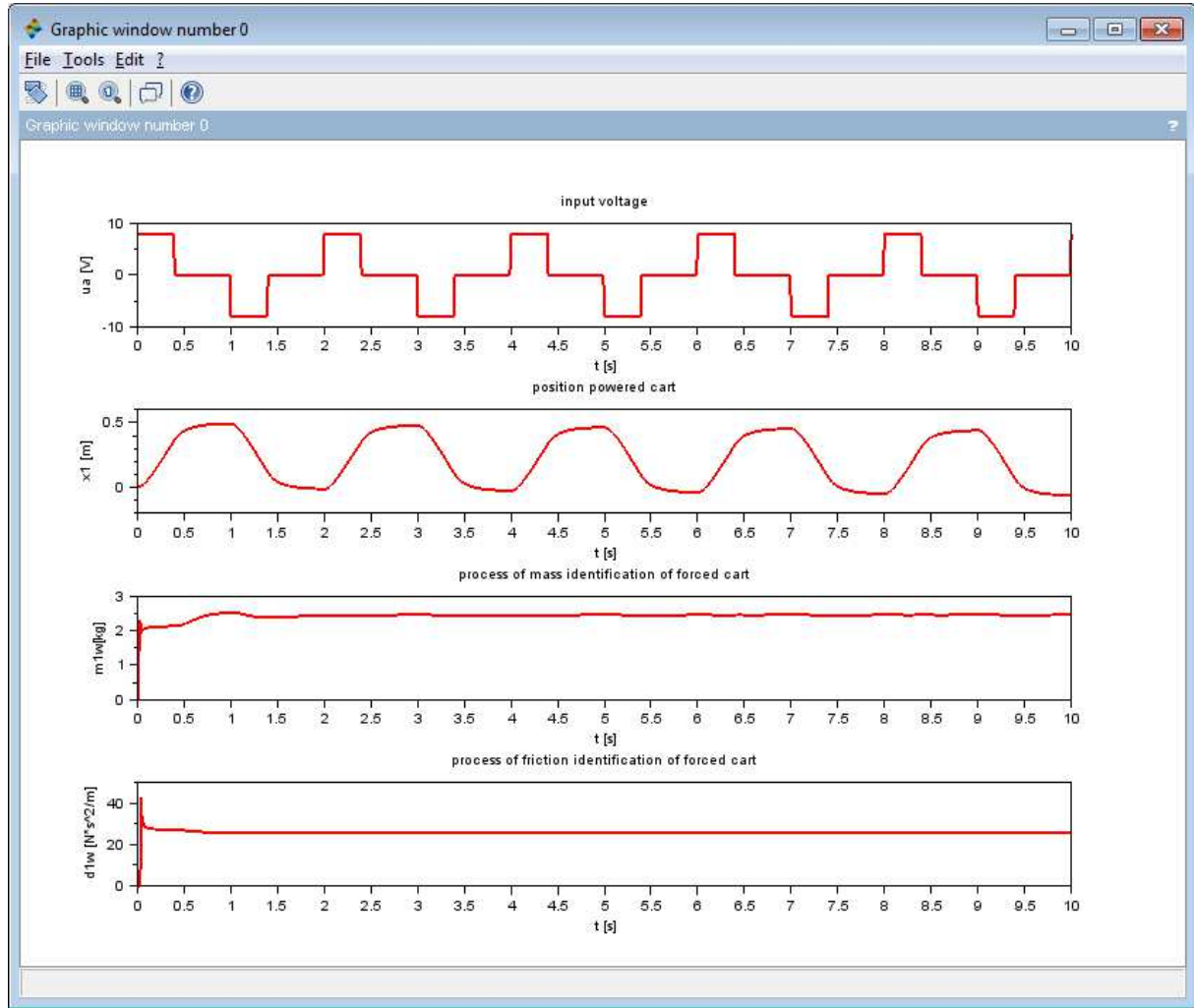


Figure 4.5: Visualisation of the parameter identification of the position servo

After executing the file “docPositionServo_ParIdentRLS.sce” it is possible to visualize a comparison between the measurement and the simulation with the identified parameters. To visualize the comparison, the file “initCompareMeasSim.sce” has to be executed.

4.3.2. HeuristicLab (HL)

The file “ParIdent_HL_PositionServo.hl” has to be executed with HeuristicLab. To do this, the executable file (...\\HeuristicLab 3.3.9\\HeuristicLab 3.3.exe) has to be started. In the window that appears, the button “Optimizer” must be selected, after which the following user interface should be visible, see Fig. 4.6. Open “ParIdent_HL_PositionServo.hl”, which is located in “...\\ModelLibrary\\PositionServo\\Identification\\HL”.

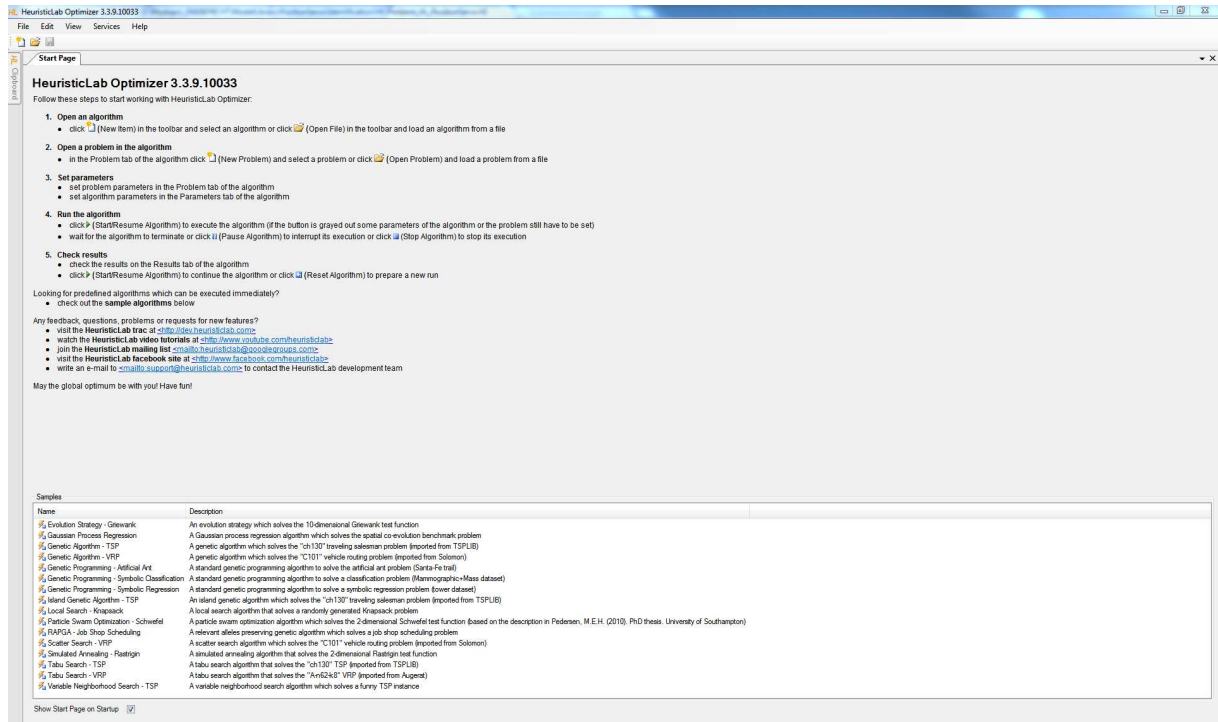


Figure 4.6: User Interface HeuristicLab Optimizer

In the tab “Results”, the results of the last identification runs are displayed. As you can see, the identified parameters (\tilde{m}_1, \tilde{d}_1) in the field “Best Solution” match perfectly with the real values. The deviation between measurement and simulation (with identified parameters) is in the order of 10^{-6} . Duration time of the identification process is in the range of 8 minutes, see Fig. 4.7.

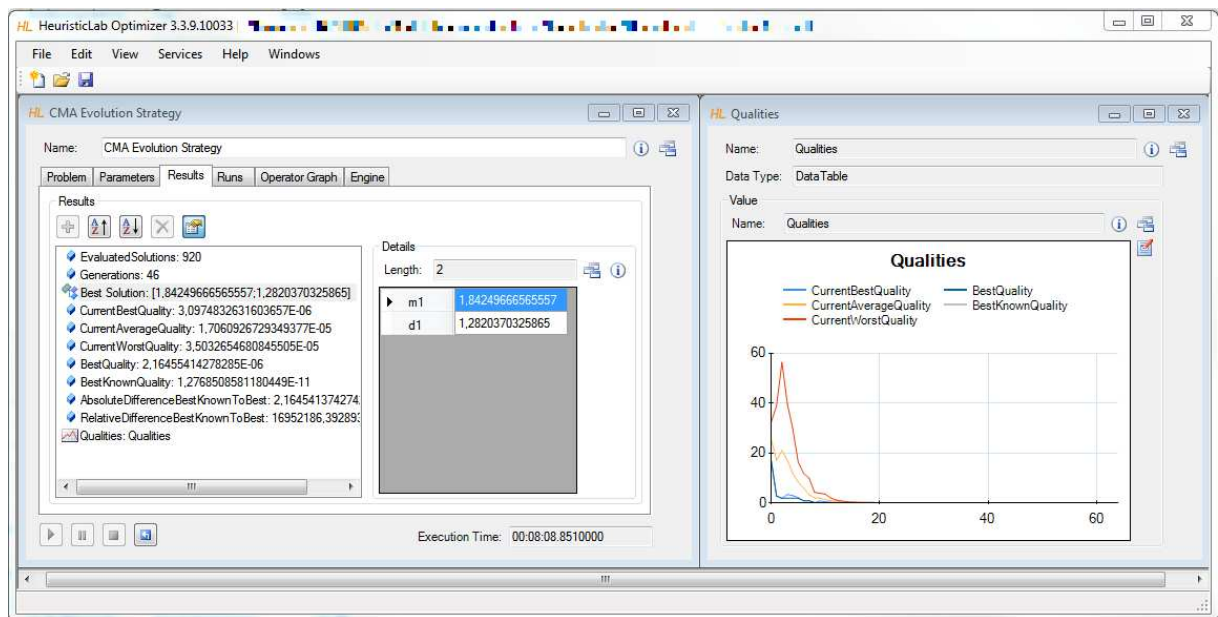


Figure 4.7: Results of an CMA-ES identification run using the example of the position servo

If you want to perform a separate run, you have to do following:

1. Verify the paths of the two Scilab scripts (“PositionServoReducedInitScript.sce” and “PositionServoReducedCyclicScript.sce”), see Fig. 4.8 and Fig. 4.9.
2. Reset Algorithm, see Fig. 4.10 at the lower left.
3. Start/Resume Algorithm, see Fig. 4.10 at the lower left.

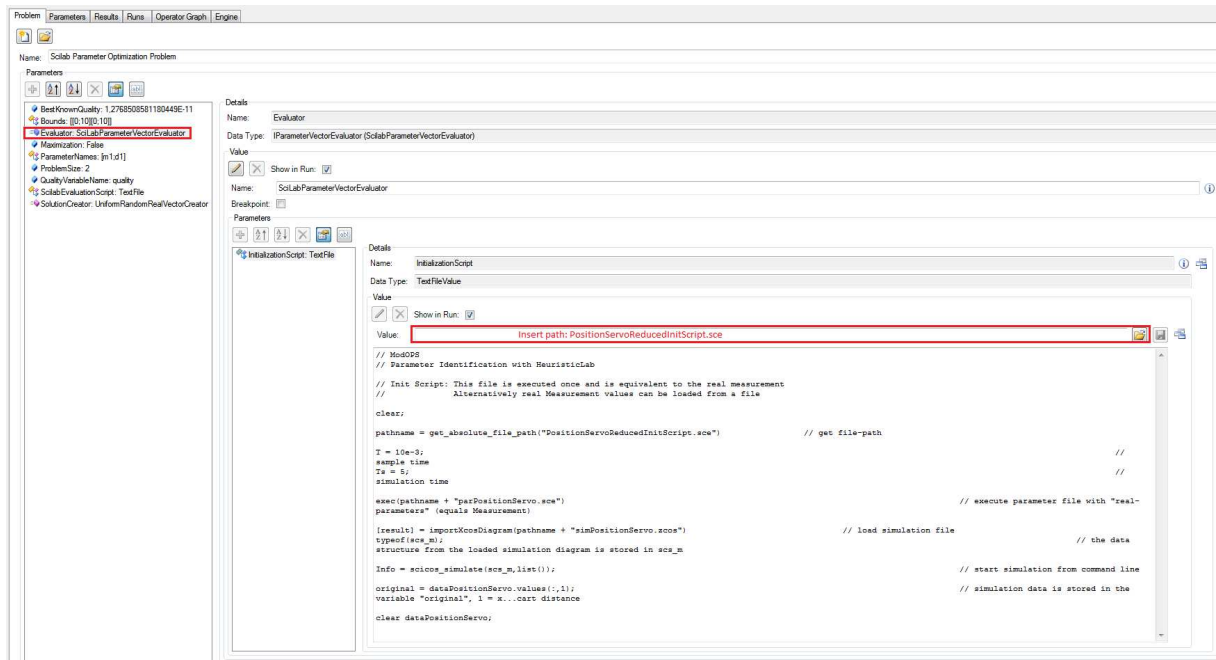


Figure 4.8: Verify path Init-script

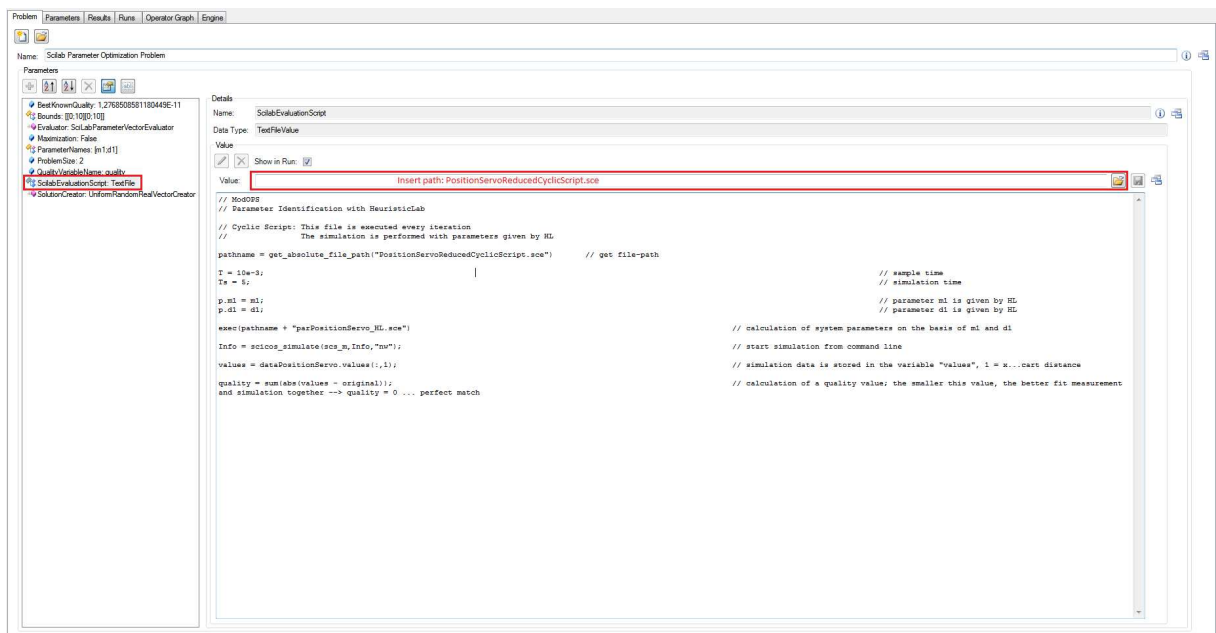


Figure 4.9: Verify Path Cyclic-script

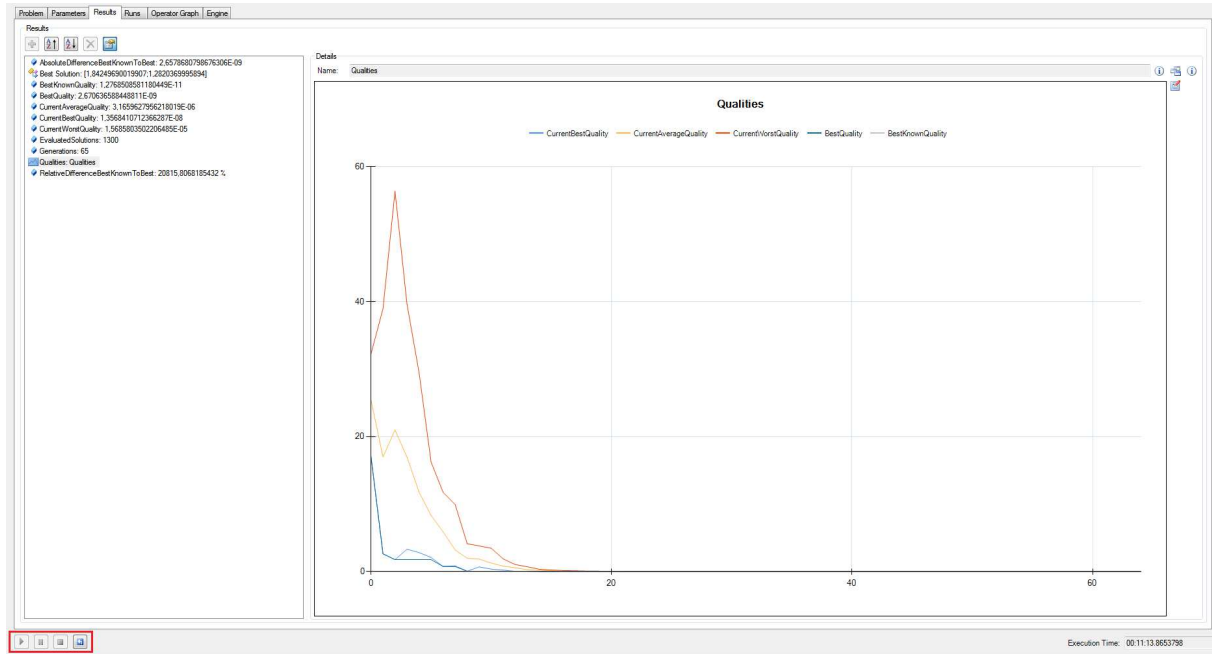


Figure 4.10: Reset and start new identification run

4.4. Regulation of system

The regulation of the model is shown by the example of a state controller. The state controller uses the inner states of the system to control the model by feedbacking them. This enables you to force the system dynamic range.

The regulation simulation file “simCtrPositionServo.zcos” (Fig. 4.11) can be found in the folder \PositionServo\Regulation. The model of the system is built out of standard XCos blocks. It would also be possible to regulate a model based on COSELICA blocks. The feedback loop with the feedback vector “kt” is shown. The setpoint for the position of the cart can be set in the superblock “setpoint presetting”.

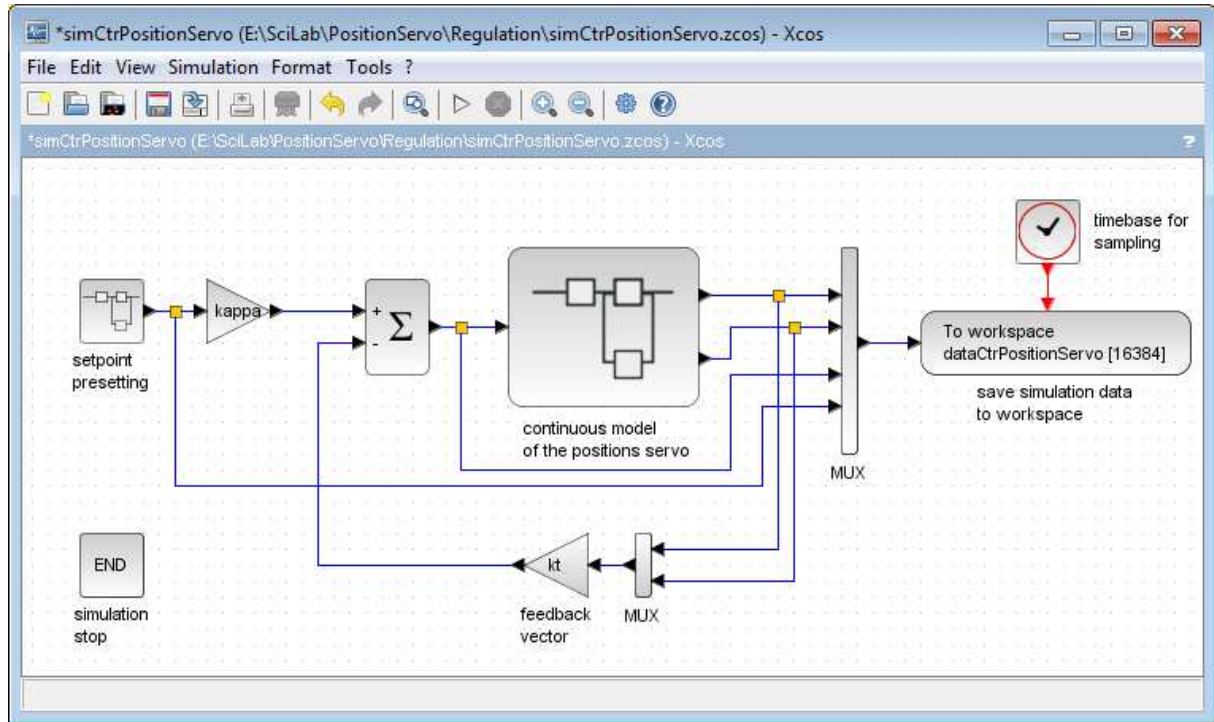


Figure 4.11: regulation of position servo

The file “initCtrPositionServo.sce” computes the feedback vector, initializes and executes the simulation. The outcome of the regulation simulation will be visualized by executing the file “docCtrPositionServo.sce”.

5. Position servo with rod pendulum

5.1. Model description

The position servo with rod pendulum is a cart with one degree of freedom, with a rod pendulum installed on it. The cart is driven by a DC motor. The drawing in Fig. 5.1 shows the schematic structure of the model.

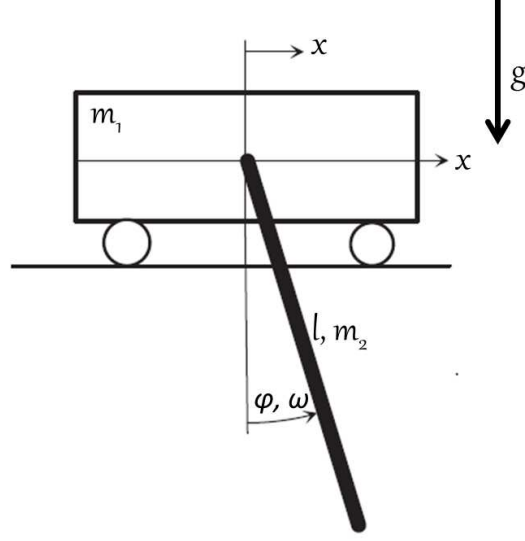


Figure 5.1: schematic of the position servo with rod pendulum

The nonlinear mathematical equations to describe this mechanical system are:

$$\dot{x} = v$$

$$\dot{\varphi} = \omega$$

$$\dot{v} = -\frac{3g m_2 \cos(\varphi) \sin(\varphi) + 4\beta u_a - 4\tilde{d}_1 v + 2l m_2 \sin(\varphi) \omega^2}{3m_2 \cos(\varphi)^2 - 4\tilde{m}_1 - 4m_2}$$

$$\dot{\omega} = -\frac{6g \tilde{m}_1 \sin(\varphi) + 6g m_2 \sin(\varphi) + 6\beta \cos(\varphi) u_a - 6\tilde{d}_1 \cos(\varphi) v + 3l m_2 \cos(\varphi) \sin(\varphi) \omega^2}{4l \tilde{m}_1 + 4l m_2 - 3l m_2 \cos(\varphi)^2}$$

Name	Description
x	position of cart
φ	angle of pendulum
v	velocity of cart
ω	angular velocity of pendulum
\tilde{m}_1	accumulated mass of cart
m_2	mass of pendulum
\tilde{d}_1	accumulated friction of cart
β	force factor of drive
g	acceleration of gravity

Table 5.1: Abbreviations cart with pendulum system

It is supposed that the friction of the pendulum is very low, so that it can be neglected. The dynamic of the electrical circuit is neglected because of its high dynamic compared to the mechanical part.

5.2. Simulation of the system

The XCos simulation file (“simPositionServo_RodPendulum.zcos”) and the COSELICA simulation file (“simPositionServo_RodPendulum_Coselica.zcos”) can be found in the folder “PositionServo_RodPendulum\Simulation”. The screenshots in Fig. 5.2 and Fig. 5.3 show the XCos diagram and the COSELICA diagram of the position servo with rod pendulum.

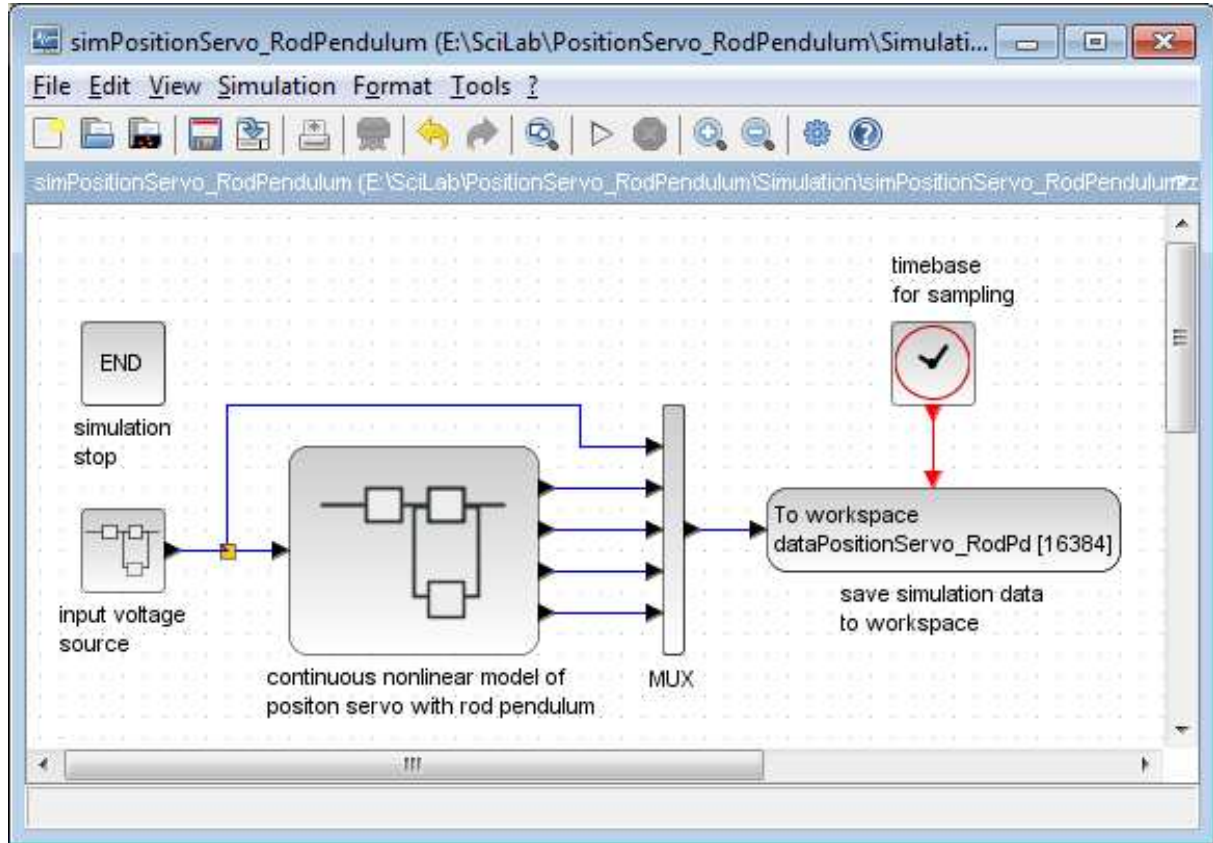


Figure 5.2: Simulation of the position servo with pendulum with standard XCos blocks

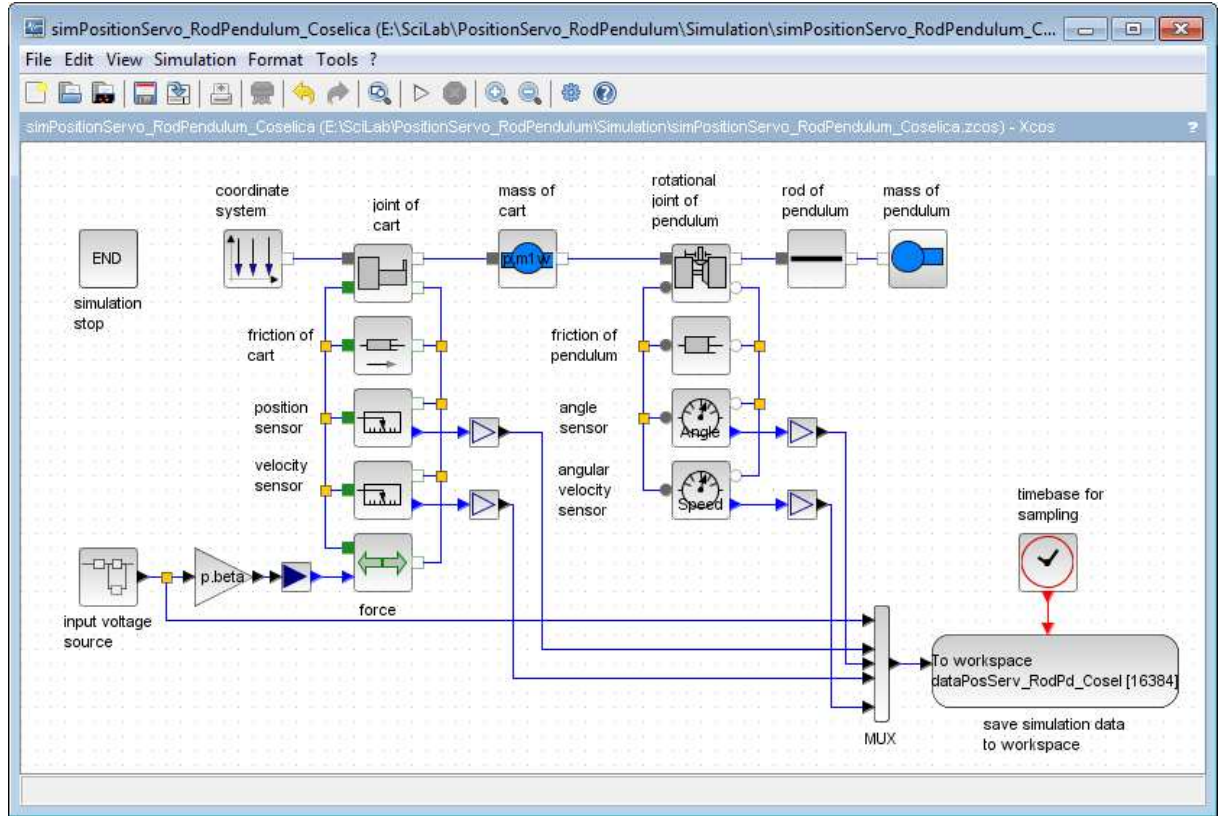


Figure 5.3: Simulation of the position servo with pendulum with COSELICA blocks

The XCos simulation of the system in Fig. 5.2 is implemented by a so-called “SCILAB function block”, very similar to the MATLAB-function block in SIMULINK. The COSELICA simulation consists of basic mechanical components, as described in Fig. 3.2.1.

To start the simulation, execute the script file “initPositionServo_RodPendulum”. This script initializes all necessary parameters and executes all simulations related to the position servo with rod pendulum. To visualize the output of the simulation, execute the file “docPostionServo_RodPendulum”.

5.3. Identification of system

Because the parameters of a mechanical system are often not known, a parameter identification, based on real measurements of the system, is necessary. One possible way of identifying a system is the RLS identification (recursive least square identification), which is used for this example.

5.3.1. Recursive Least Square Algorithm

This method assumes that the mathematical model of the system is known, furthermore the input signal and the output signal of a measurement are required.

The file “initPositionServo_RodPd_ParIdentRLS.sce” in the folder \PositionServo_RodPendulum\Identification initializes and executes the identification that is carried out by the identification program

“measPositionServo_RodPd_ParIdentRLS.zcos”. The screenshot in Fig. 5.4 shows the identification program. The RLS-subsystem is provided with the measurement data of the position servo with rod pendulum and calculates the parameters of the system. Afterwards, the identification output is stored to the workspace.

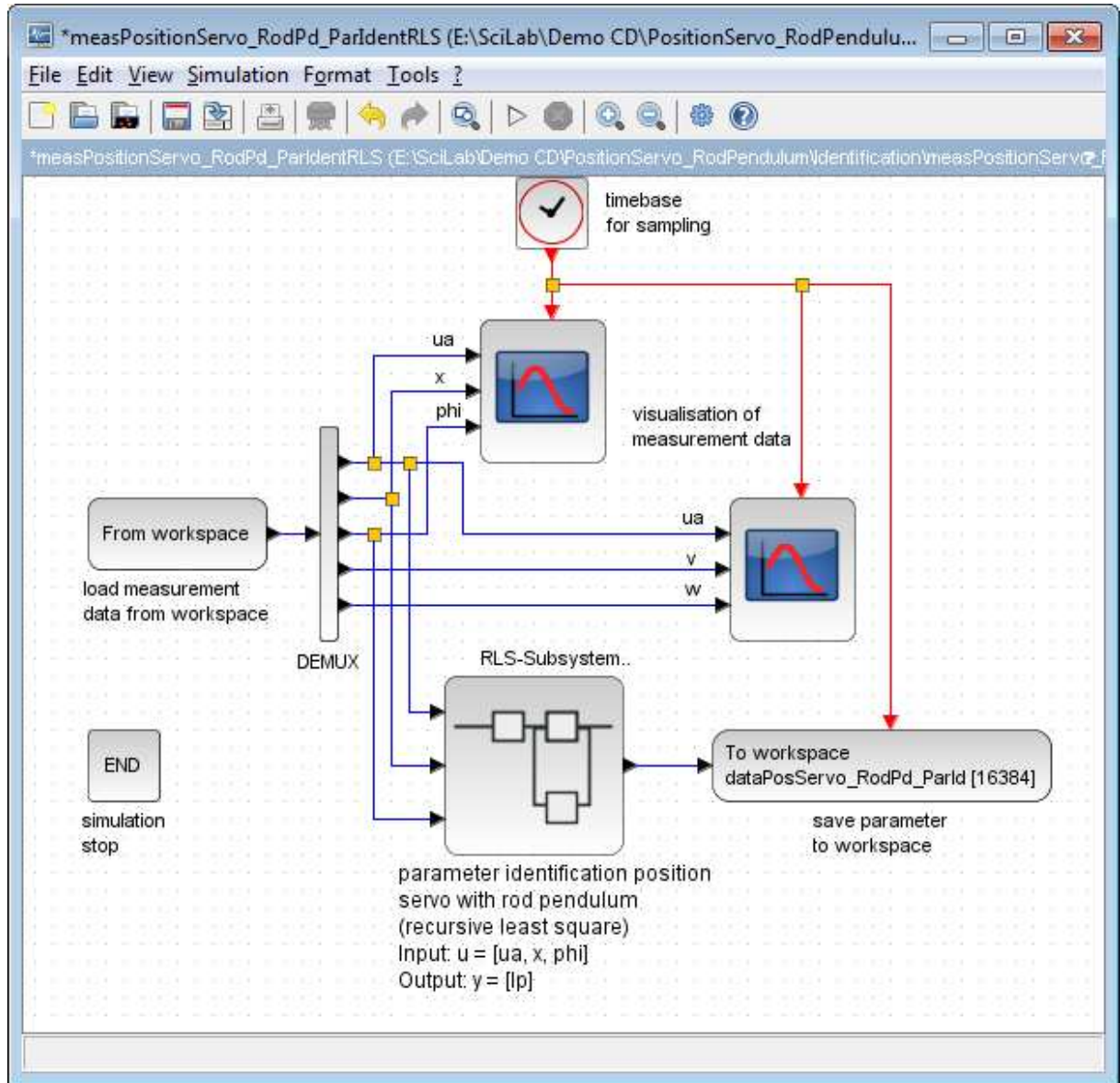


Figure 5.4: Identification program of position servo with rod pendulum

The prepared measurement data of a real position servo is stored in `dataIdent_PosSrv_RdPd_MAT.mat`. By executing the file `docPositionServo_RodPendulum_ParIdentRLS.sce` the identification will be visualized. The graph in Fig. 5.5 shows the input signals (u_a , x and ϕ) and the parameter output signal (length of pendulum l) of the RLS-block. The valid identification parameter is the stationary value of the parameter signal, it is stored in the structure `pIdent` at the workspace. The value of the mass of the pendulum is not identified because it has been measured. The

accumulated mass and friction of the cart is the same as those of the position servo, see Fig. 4.3.

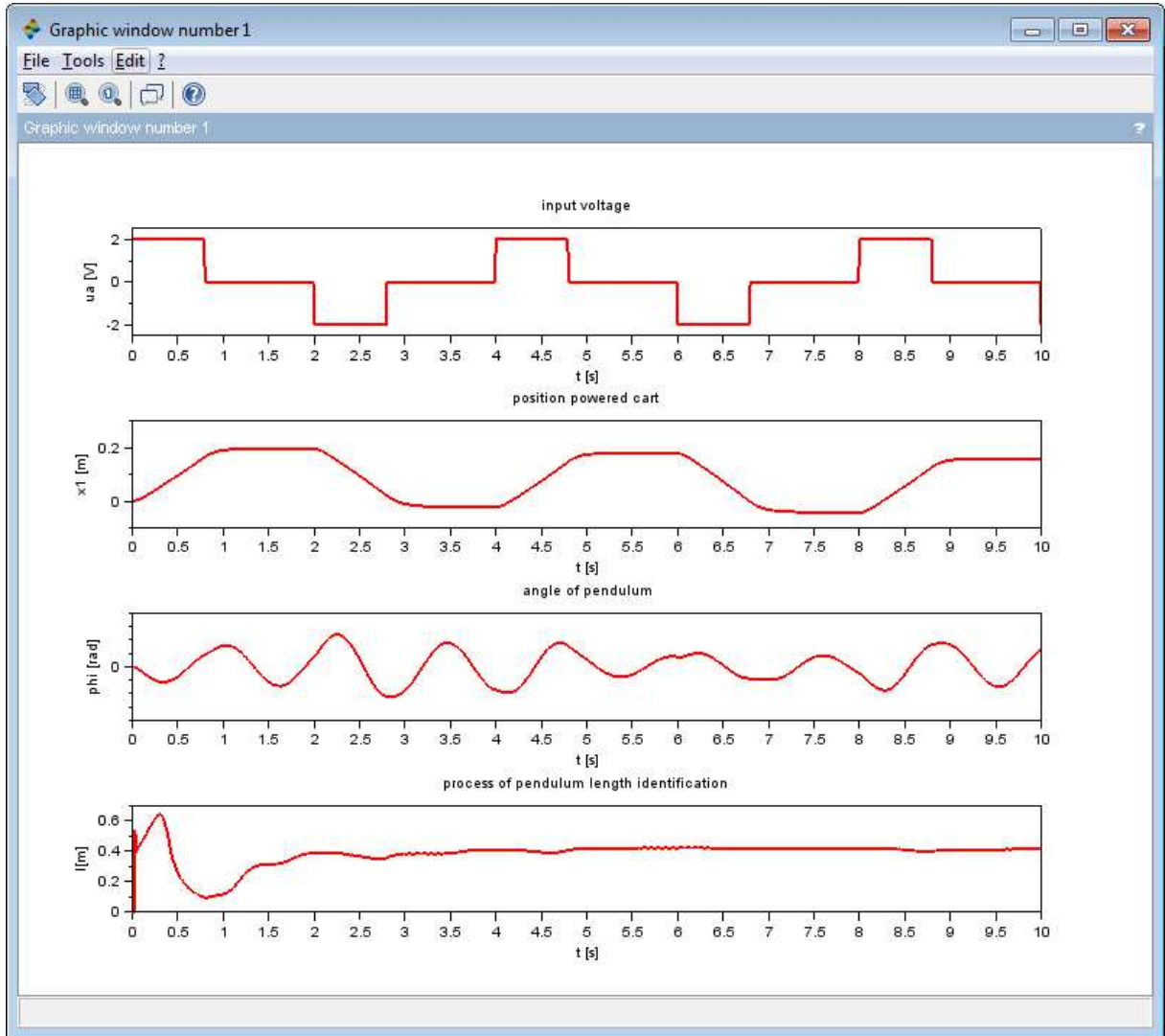


Figure 5.5: Visualisation of the parameter identification of the position servo with rod pendulum

After executing the file “docPositionServo_RodPendulum_ParIdentRLS.sce” it is possible to visualize a comparison between the measurement and the simulation with the identified parameters. To visualize the comparison, the file “initCompareMeasSim.sce” has to be executed.

5.3.2. HeuristicLab (HL)

The file “ParIdent_HL_PositionServo_RodPendulum.hl” has to be executed with HeuristicLab. For detailed information on the operation of HeuristicLab, see chapter 4.3.2. In the tab “Results”, the results of the last identification runs are displayed. As you can see, the identified parameter (l) in field “Best Solution” matches perfectly with the real value. The deviation between measurement and simulation is in the order of 10^{-6} . Duration time of the identification process is in the range of 9 minutes, see Fig. 5.6.

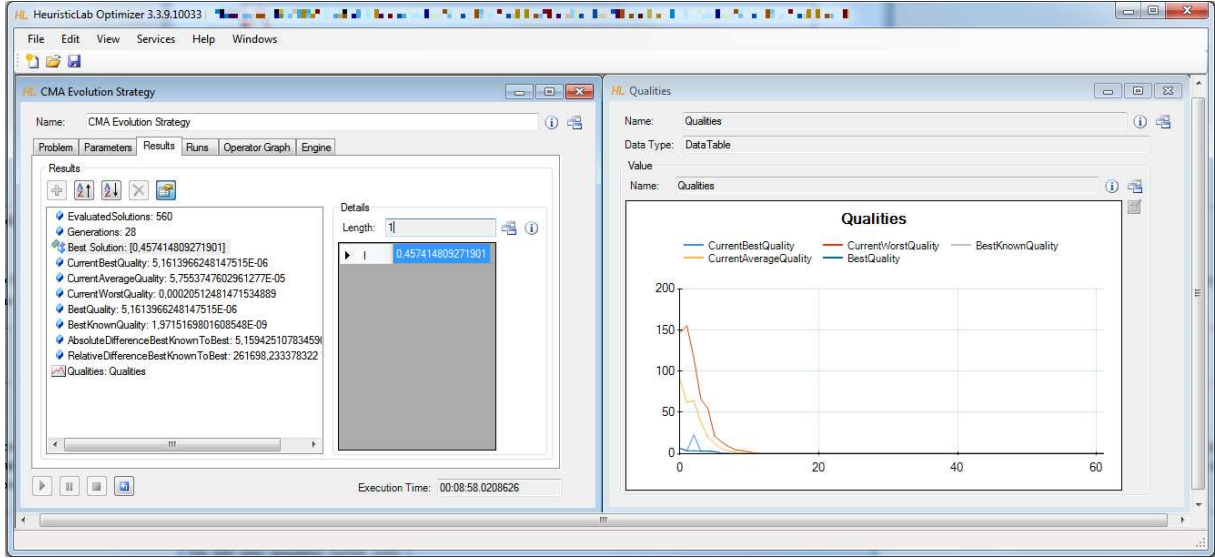


Figure 5.6: Results of an CMA-ES identification run using the example of the position servo with rod pendulum

5.4. Regulation of system

The regulation of the model is shown by the example of a state controller. The state controller uses the inner states of the system to control the model by feedbacking them. This enables you to force the system dynamic range. The state controller is developed for a linearized mathematical model of the system.

The regulation simulation file “simCtrPositionServo_RodPendulum.zcos” is shown in Fig. 5.7, it can be found in the folder \PositionServo_RodPendulum\Regulation. The model of the system is built out of standard XCos blocks, but it would also be possible to regulate a model based on COSELICA blocks. The feedback loop with the feedback vector “kt” is shown. The setpoint for the position of the cart can be set in the superblock “setpoint presetting”. The controller regulates the angle of the pendulum to zero.

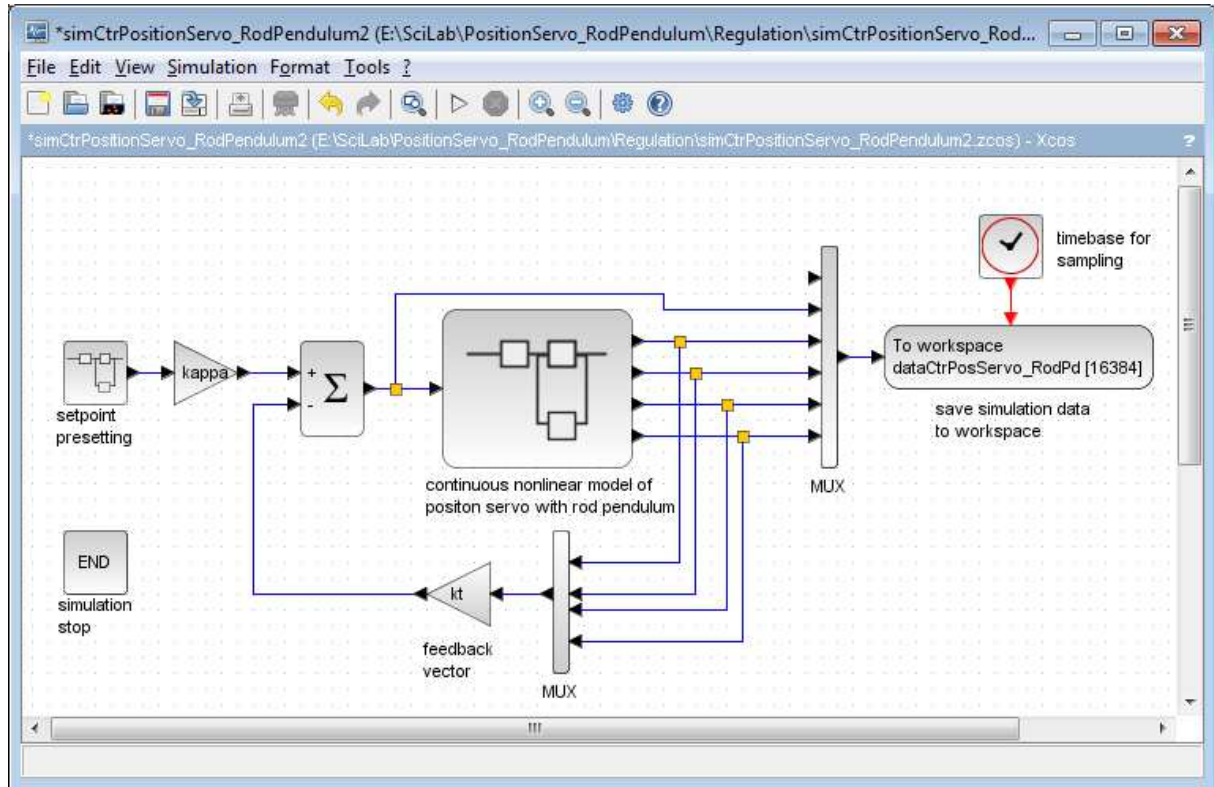


Figure 5.7: regulation of position servo with rod pendulum

The file “initCtrPositionServo_RodPendulum.sce” computes the feedback vector, initializes and executes the simulation . The outcome of the regulation simulation will be visualized by executing the file “docCtrPositionServo_RodPendulum.sce”.

6. Two mass oscillator

6.1. Model description

The two mass oscillator consists of two carts with one degree of freedom that are connected with a spring. One cart is driven by a DC motor. The drawing in Fig. 6.1 shows the schematic structure of the model.

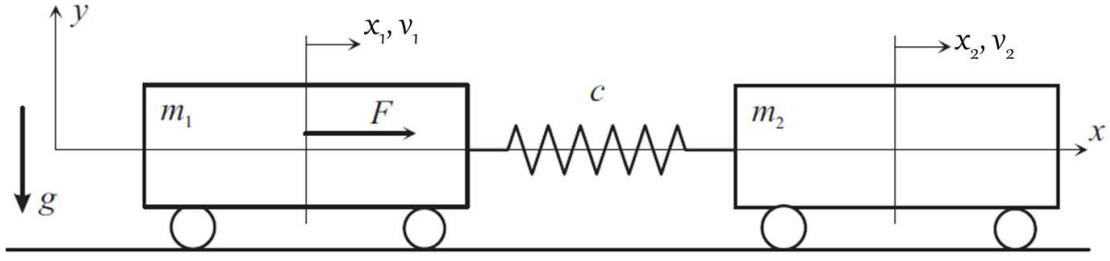


Figure 6.1: schematic of the two mass oscillator

The mathematical equations to describe this mechanical system are:

$$\begin{aligned}
 \dot{x}_1 &= v_1 \\
 \dot{x}_2 &= v_2 \\
 \dot{v}_1 &= -\frac{\tilde{d}_1}{\tilde{m}_1}v_1 + \frac{c}{\tilde{m}_1}(x_2 - x_1) + \frac{d_{12}}{\tilde{m}_1}(v_2 - v_1) + \frac{\beta}{\tilde{m}_1}u_a \\
 \dot{v}_2 &= -\frac{d_2}{m_2}v_2 - \frac{c}{m_2}(x_2 - x_1) - \frac{d_{12}}{m_2}(v_2 - v_1)
 \end{aligned}$$

Name	Description
x_1	position of forced cart
x_2	position of unforced cart
v_1	velocity of forced cart
v_2	velocity of unforced cart
\tilde{m}_1	accumulated mass of forced cart
m_2	mass of unforced cart
\tilde{d}_1	accumulated friction of forced cart
d_2	friction of unforced cart
c	spring constant
d_{12}	friction of spring
β	force factor of drive
g	acceleration of gravity

Table 6.1: Abbreviations Two mass oscillator

The dynamic of the electrical circuit is neglected because of its high dynamic compared to the mechanical part.

6.2. Simulation of the system

The XCos simulation file (“simTwoMassOsc.zcos”) and the COSELICA simulation file (“simTwoMassOsc_Coselica.zcos”) can be found in the folder “TwoMassOscillator\Simulation”.

The screenshots in Fig. 6.2 and Fig. 6.3 show the XCos diagram and the COSELICA diagram of the two mass oscillator.

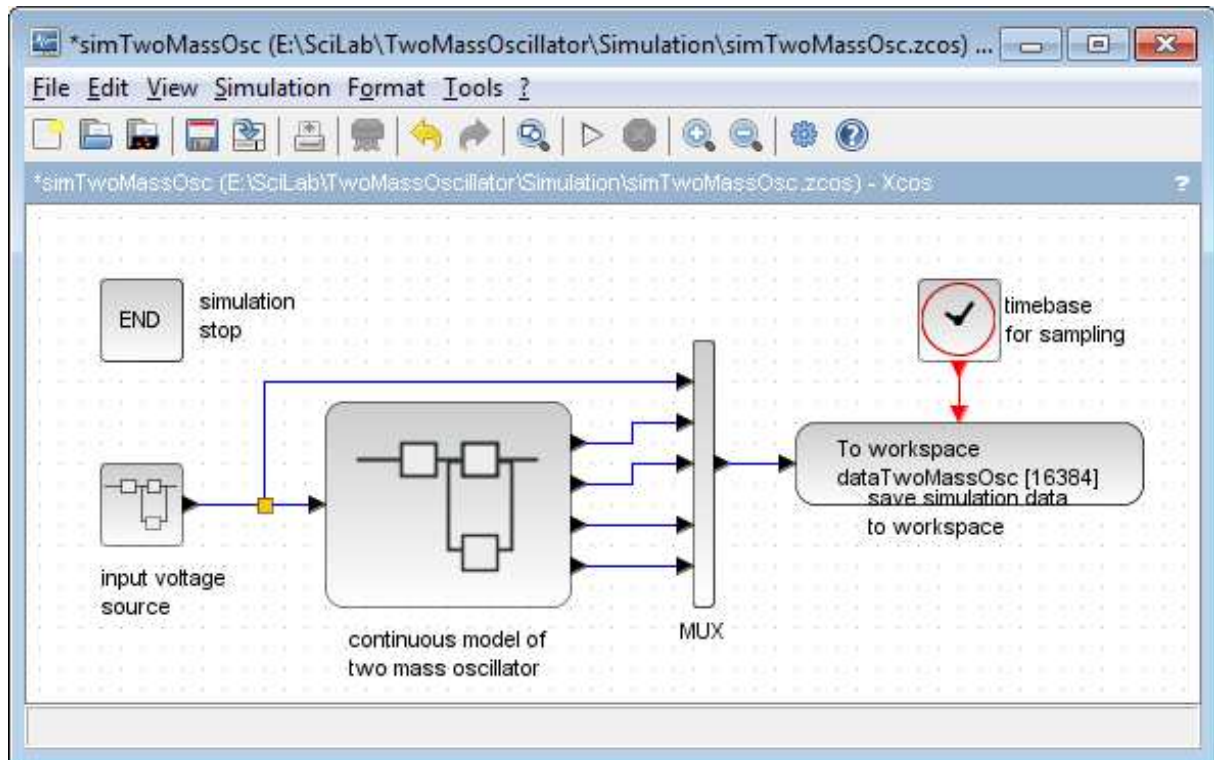


Figure 6.2: Simulation of the two mass oscillator with standard XCos blocks

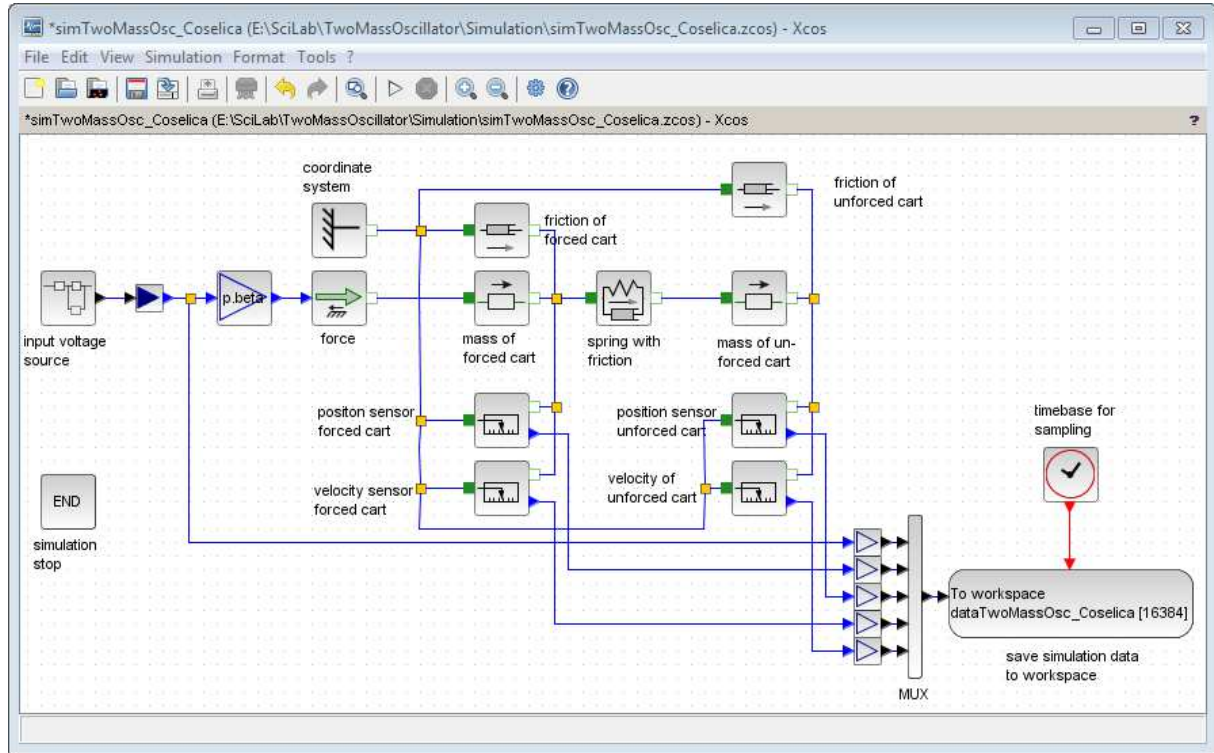


Figure 6.3: Simulation of the two mass oscillator with COSELICA blocks

The Xcos simulation of the system in Fig. 6.2 is implemented by a so-called “SCILAB function block”, very similar to the MATLAB-function block in SIMULINK. In opposite to the COSELICA example in Fig. 3.2.1 with blocks from the planar library, the simulation in Fig. 6.3 is built with blocks from the translational library. How the blocks are used is very similar to Fig. 3.2.1. The only difference is that there are no joints in the translational library because there is only one degree of freedom. For this reason, the forces are connected to masses, not to joints. To start the simulation, execute the script file “initTwoMassOsc”. This script initializes all necessary parameters and executes all simulations related to the two mass oscillator. To visualize the output of the simulation, execute the file “docTwoMassOsc”.

6.3. Identification of system

Because the parameters of a mechanical system are often not known, a parameter identification, based on real measurements of the system, is necessary. One possible way of identifying a system is the RLS identification (recursive least square identification), which is used for this example.

6.3.1. Recursive Least Square Algorithm

This method assumes that the mathematical model of the system is known, furthermore the input signal and the output signal of a measurement are required.

The file “initTwoMassOsc_ParIdentRLS.sce” in the folder \TwoMassOscillator\Identification initializes and executes the identification that is carried out by the identification program “measTwoMassOsc_ParIdentRLS.zcos”. The screenshot in Fig. 6.4 shows the identification program. The RLS-subsystem is provided with the measurement data of the two mass oscillator and calculates the parameters of the system. Afterwards, the identification output is stored to the workspace.

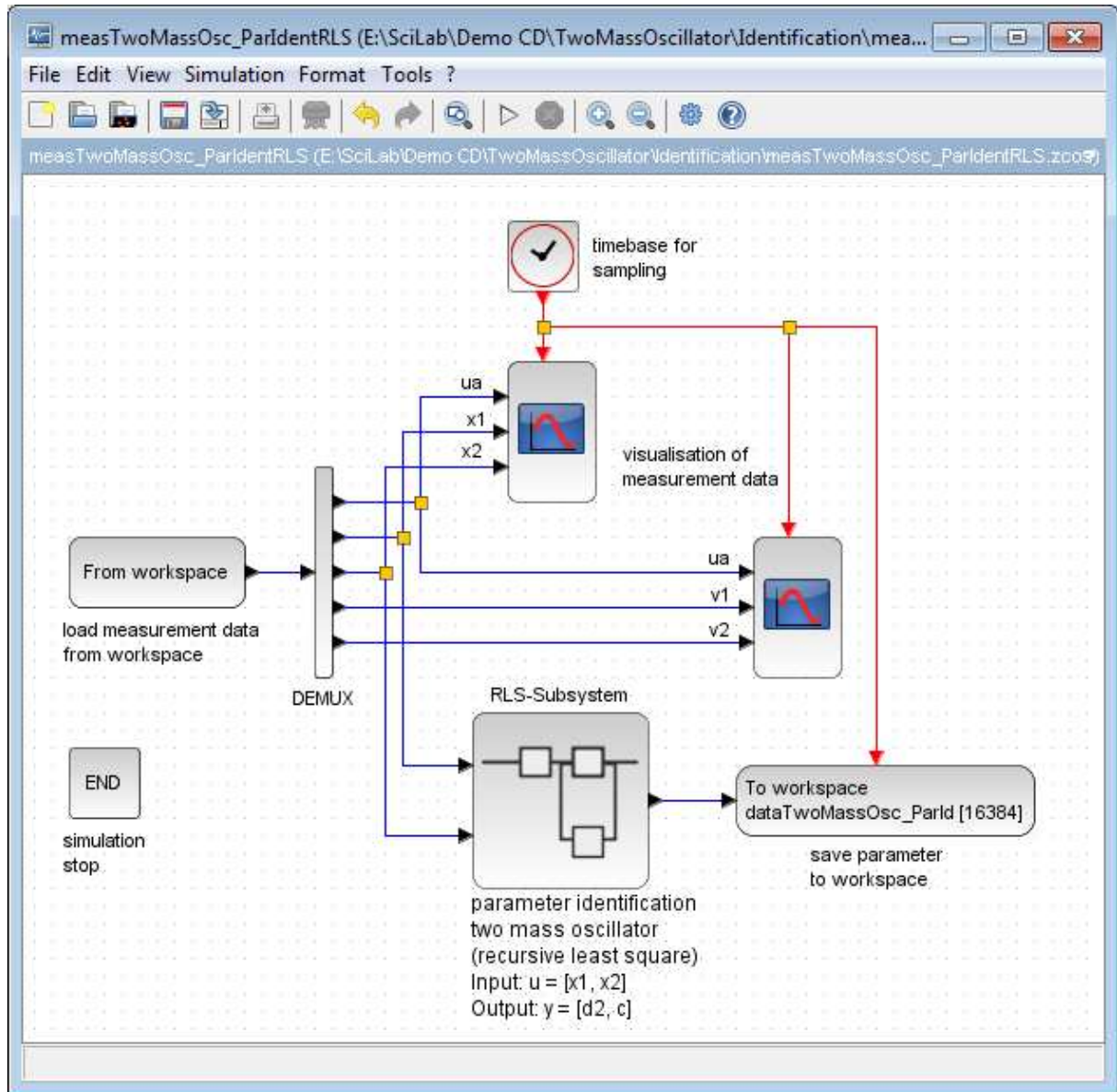


Figure 6.4: Identification program of two mass oscillator

The prepared measurement data of a real two mass oscillator is stored in “dataIdent_TwoMassOsc_MATLAB.mat”. The identification will be visualized by executing the file “docTwoMassOsc_ParIdentRLS.sce”. The graph in Fig. 6.5 shows the input signals (x_1 and x_2) and the parameter output signals (d_2 and c) of the RLS-block. The valid identification

parameters are the stationary values of the parameter signals, they are stored in the structure `pIdent` at the workspace. The value of the spring friction was neglected because it is very low. The accumulated mass and friction of the forced cart is the same as those of the position servo (identified in Chapter 4.3). The mass of the unforced cart is known because it can be weighted.

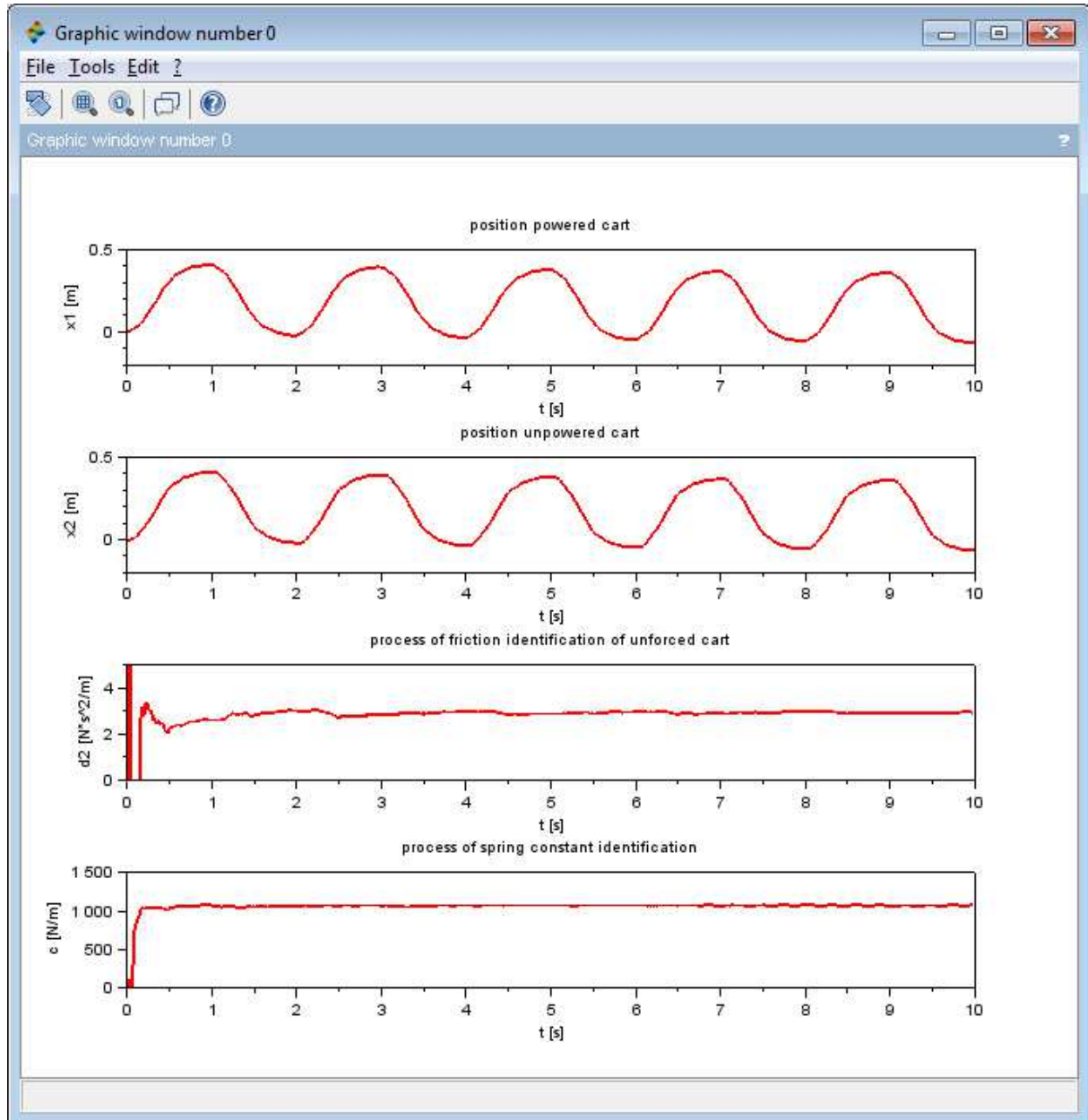


Figure 6.5: Visualisation of the parameter identification of the two mass oscillator

After executing the file "docTwoMassOsc_ParIdentRLS.sce" it is possible to visualize a comparison between the measurement and the simulation. To visualize the comparison, the file "initCompareMeasSim.sce" has to be executed.

6.3.2. HeuristicLab (HL)

The file “ParIdent_HL_TwoMassOsc.hl” has to be executed with HeuristicLab. For detailed information on the operation of HeuristicLab, see Chapter 4.3.2. In the tab “Results”, the results of the last identification run are displayed. As you can see, the identified parameters (d_2, c) in field “Best Solution” match perfectly with the real values. The deviation between measurement and simulation with the identified parameters is in the order of 10^{-6} . Duration time of the identification process is in the range of 23 minutes, see Fig. 6.6.

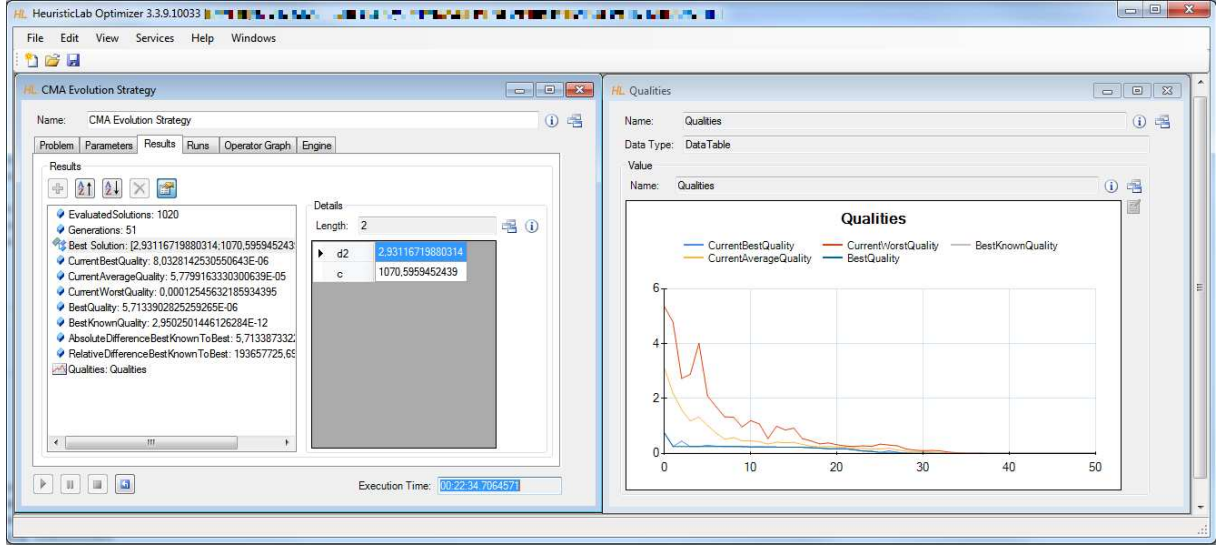


Figure 6.6: Results of an CMA-ES identification run using the example of the Two-Mass-Oscillator

6.4. Regulation of system

The regulation of the model is shown by the example of a state controller. The state controller uses the inner states of the system to control the model by feedbacking them. This enables you to force the system dynamic range.

The regulation simulation file “simCtrTwoMassOsc.zcos” is shown in Fig. 6.7, and can be found in the folder \TwoMassOscillator\Regulation. The model of the system is built out of standard XCos blocks, but it would also be possible to regulate a model based on COSELICA blocks. The feedback loop with the feedback vector “kt” is shown. The setpoint for the position of the unforced cart can be set in the superblock “setpoint presetting”.

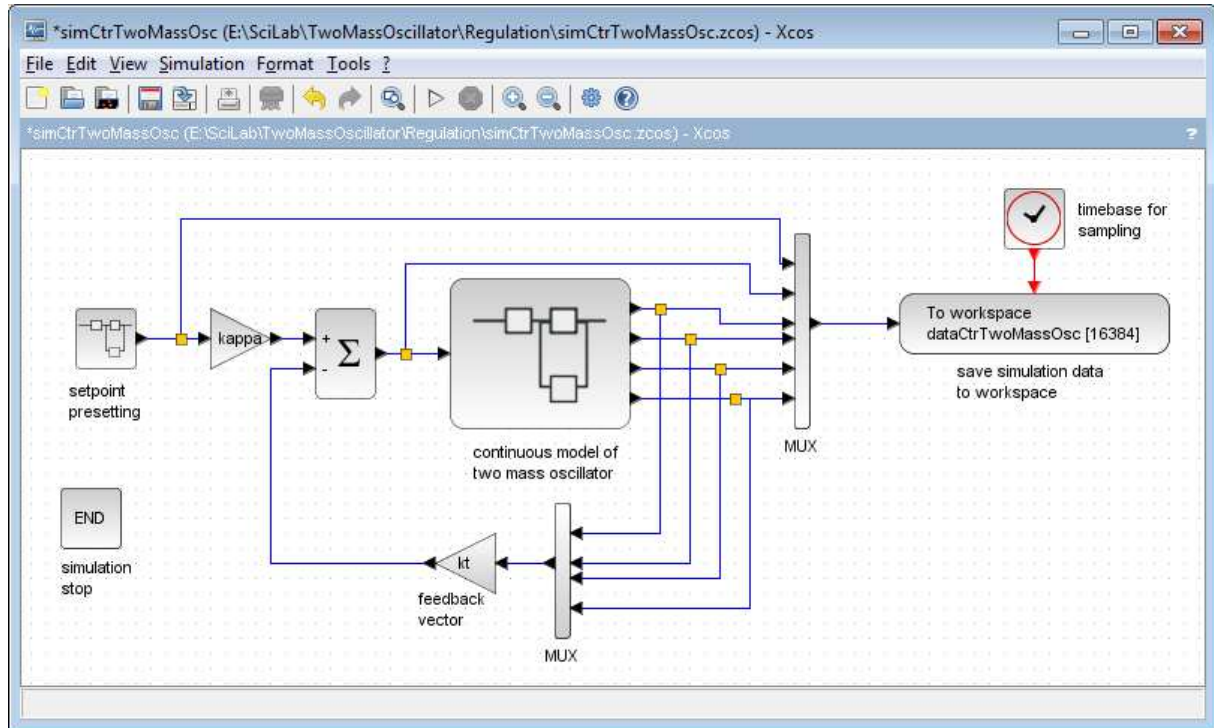


Figure 6.7: regulation of two mass oscillator

The file “initCtrTwoMassOsc.sce” computes the feedback vector, initializes and executes the simulation. The outcome of the regulation simulation will be visualized by executing the file “docCtrTwoMassOsc.sce”.

7. Two mass oscillator with rod pendulum

7.1. Model description

The two mass oscillator with rod pendulum consists of two carts with one degree of freedom that are connected with a spring. One cart is driven by a DC motor. On the second cart, there is a rod pendulum installed. The drawing in Fig. 7.1 shows the schematic structure of the model.

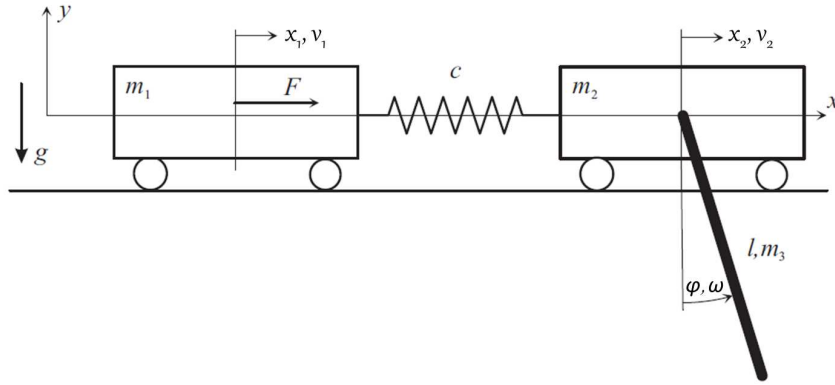


Figure 7.1: schematic of the two mass oscillator with rod pendulum

The nonlinear mathematical equations to describe this mechanical system are:

$$\dot{x}_1 = v_1$$

$$\dot{x}_2 = v_2$$

$$\dot{\varphi} = \omega$$

$$\dot{v}_1 = -\frac{\tilde{d}_1}{\tilde{m}_1}v_1 + \frac{c}{\tilde{m}_1}(x_2 - x_1) + \frac{d_{12}}{\tilde{m}_1}(v_2 - v_1) + \frac{\beta}{\tilde{m}_1}u_a$$

$$\dot{v}_2 = \frac{8cx_1 - 8cx_2 + 8d_{12}v_1 - 8(d_{12} + d_2)v_2 + 2m_3\sin(\varphi)(3g\cos(\varphi) + 2l\omega^2)}{8m_2 + 5m_3 - 3m_3\cos(2\varphi)}$$

$$\dot{\omega} = \frac{-6g(m_2 + m_3)\sin(\varphi) - 3\cos(\varphi)(2cx_1 - 2cx_2 + 2d_{12}v_1 - 2(d_{12} + d_2)v_2 + lm_3\sin(\varphi)\omega^2)}{2l}$$

Name	Description
x_1	position of forced cart
x_2	position of unforced cart
φ	angle of pendulum
v_1	velocity of forced cart
v_2	velocity of unforced cart
ω	angular velocity of pendulum
\tilde{m}_1	accumulated mass of forced cart
m_2	mass of unforced cart
m_3	mass of pendulum
\tilde{d}_1	accumulated friction of forced cart
d_2	friction of unforced cart
c	spring constant
d_{12}	friction of spring
β	force factor of drive
g	acceleration of gravity

Table 7.1: Abbreviations Two mass oscillator with rod pendulum

It is supposed that the friction of the pendulum is very low, so that it can be neglected. The dynamic of the electrical circuit is neglected because of its high dynamic compared to the mechanical part.

7.2. Simulation of the system

The simulation file (“simTwoMassOscillator_RodPendulum.zcos”) can be found in the folder “TwoMassOscillator_RodPendulum\Simulation”. The screenshot in Fig. 7.2 show the XCos diagram of the two mass oscillator with rod pendulum.

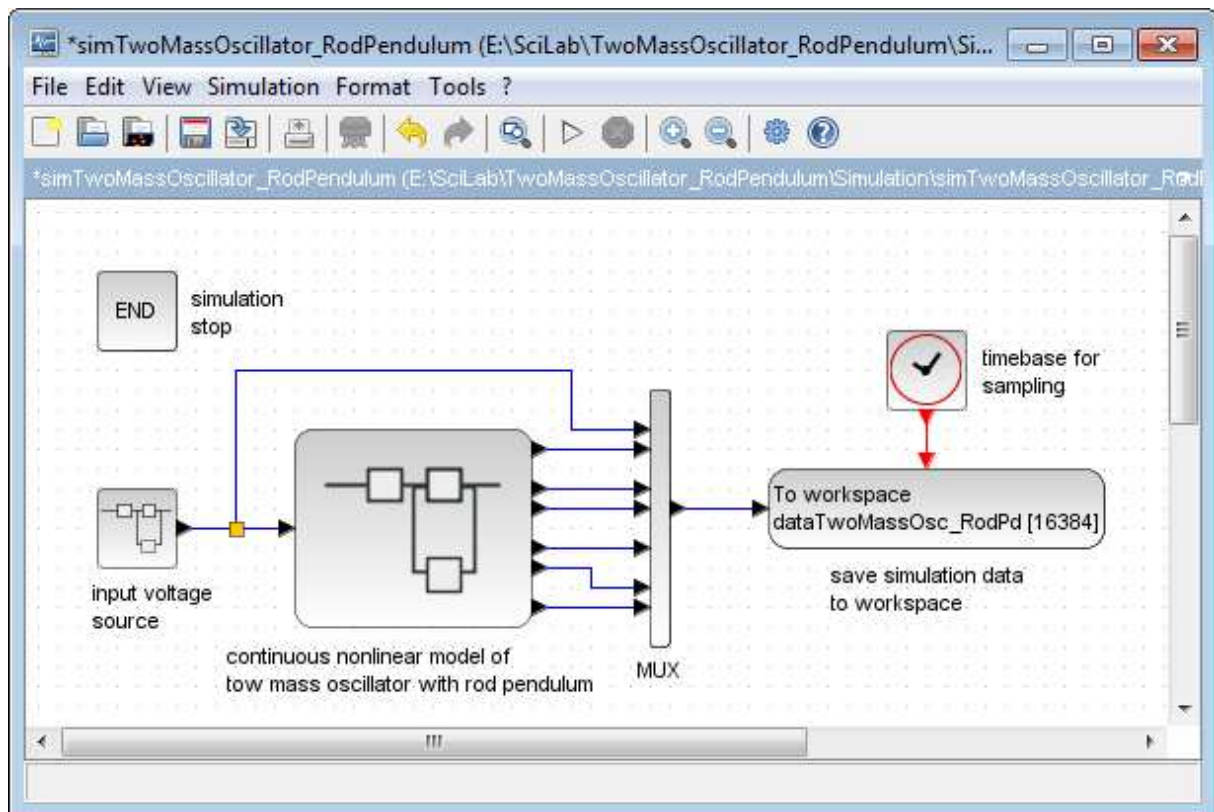


Figure 7.2: Simulation of the two mass oscillator with pendulum

The XCos simulation of the system in Fig. 7.2 is implemented by a so-called “SCILAB function block”, very similar to the MATLAB function block in SIMULINK. To start the simulation, execute the script file “initTwoMassOscillator_RodPendulum.sce”. This script initializes all necessary parameters and executes the simulation of the two mass oscillator with rod pendulum. To visualize the output of the simulation, execute the file “docTwoMassOscillator_RodPendulum.sce”.

7.3. Regulation of system

The regulation of the model is shown by the example of a state controller. The state controller uses the inner states of the system to control the model by feedbacking them. This enables you to

force the system dynamic range. The state controller is developed for a linearized mathematical model of the system.

The XCos regulation simulation file “simCtrTwoMassOscillator_RodPendulum.zcos” is shown in Fig. 7.3 and can be found in the folder \TwoMassOscillator_RodPendulum\Regulation. The feedback loop with the feedback vector “kt” is shown. The setpoint for the position of the unforced car can be set in the superblock “setpoint presetting”. The controller regulates the angle of the pendulum on the unforced cart to zero.

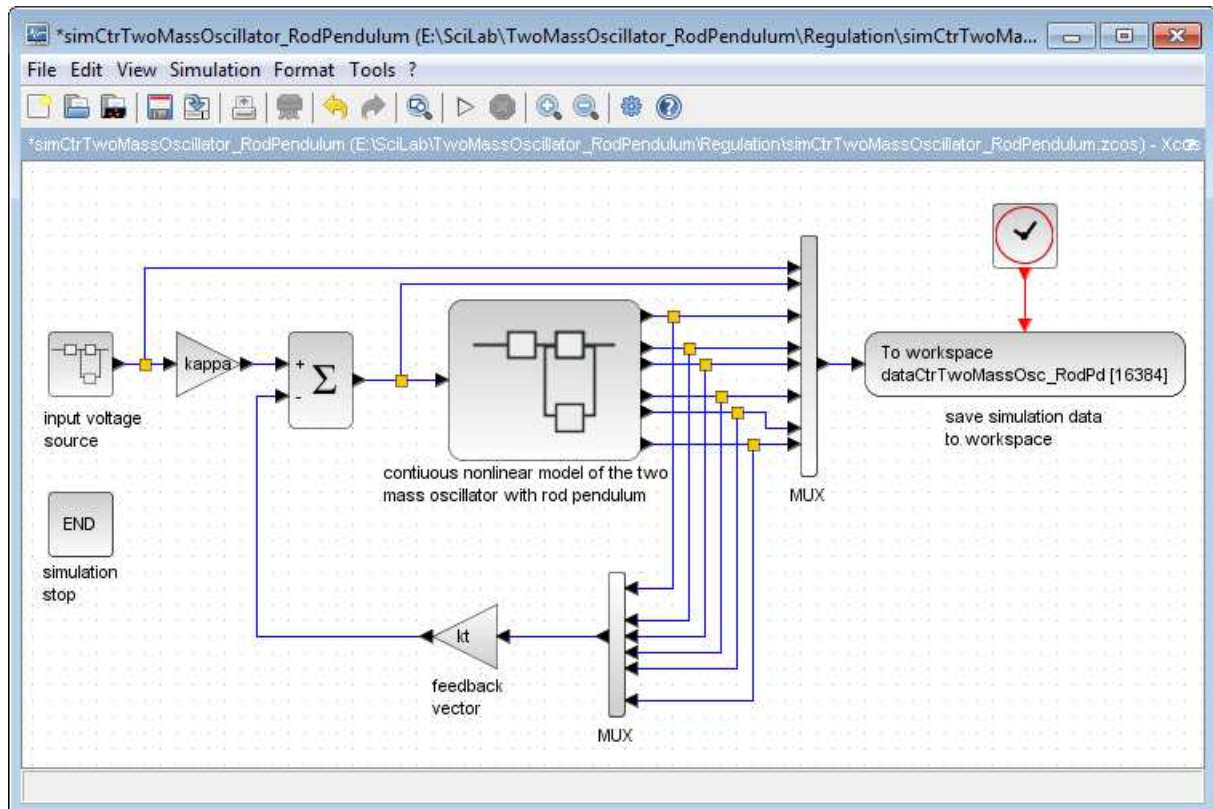


Figure 7.3: regulation of two mass oscillator with rod pendulum

The file “initCtrTwoMassOscillator_RodPendulum.sce” computes the feedback vector, initializes and executes the simulation. The outcome of the regulation simulation will be visualized by executing the file “docTwoMassOscillator_RodPendulum.sce”.

References

- [1] Wagner, S. et al. Architecture and Design of the HeuristicLab Optimization Environment. In Advanced Methods and Applications in Computational Intelligence, Topics in Intelligent Engineering and Informatics Series, pp. 197-261. Springer (2014)