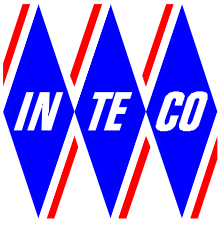
Magnetic Levitation System 2EM

(MLS2EM)

USB Version

User's manual



www.inteco.com.pl

Table of contents

Introduction

The *Magnetic Levitation System with 2 Electromagnets* (*MLS2EM*) is a complete (after assembling and software installation) control laboratory system ready to experiments. The is an ideal tool for demonstration of magnetic levitation phenomena. This is a classic control problem used in many practical applications such as transportation - magnetic levitated trains, using both analogue and digital solutions to maintain a metallic ball in an electromagnetic field. *MLS2EM* consists of two electromagnets, the suspended hollow steel sphere, the sphere position sensors, computer interface board and drivers, a signal conditioning unit, connecting cables, real time control toolbox and a laboratory manual. This is a single degree of freedom system for teaching of control systems; signal analysis, real-time control applications such as MATLAB. MLS is a nonlinear, open-loop unstable and time varying dynamical system. The basic principle of MLS operation is to apply the voltage to an upper electromagnet to keep a ferromagnetic object levitated. The object position is determined through a sensor. Additionally the coil current is measured to explore identification and multi loop or nonlinear control strategies. To levitate the sphere a real-time controller is required. The equilibrium stage of two forces (the gravitational and electromagnetic) has to be maintained by this controller to keep the sphere in a desired distance from the magnet. When two electromagnets are used the lower one can be used for external excitation or as contraction unit. This feature extends the MLS application and is useful in robust controllers design. The position of the sphere may be adjusted using the set-point control and the stability may be varied using the gain control. Two different diameter spheres are provided. The band-width of lead compensation may be changed and the stability and response time investigated. User-defined controllers may be tested.

1.1 Laboratory set-up

A schematic diagram of the laboratory set-up is shown in Fig. 1.

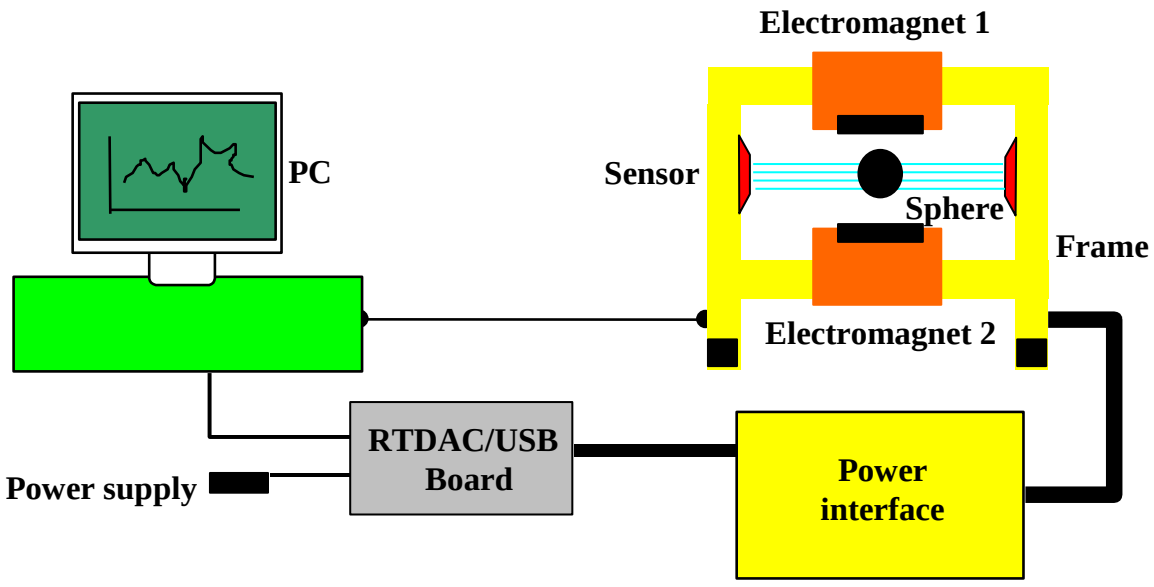


Fig. 1. MLS2EM laboratory set-up

One obtains the mechanical unit with power supply and interface to a PC and the dedicated RTDAC/USB2 I/O board configured in the Xilinx® technology. The software operates in real time under MS Windows® using MATLAB®/Simulink and Simulink Coder toolboxes .

Control experiments are programmed and executed in real-time in the MATLAB/Simulink environment. Thus it is strongly recommended to a user to be familiar with the Simulink Coder Toolbox. One has to know how to use the attached models and how to create his own models.

The control software for the MLS2EM is included in the *MLS2EM toolbox*. This toolbox uses the built in RT-CON from Inteco and Simulink Coder toolbox from Mathworks.

MLS2EM Toolbox is a collection of M-functions, MDL-models and C-code DLL-files that extends the MATLAB environment in order to solve MLS modelling, design and control problems. The integrated software supports all phases of a control system development:

- on-line process identification,
- control system modelling, design and simulation,
- real-time implementation of control algorithms.

MLS2EM Toolbox is intended to provide a user with a variety of software tools enabling:

- on-line information flow between the process and the MATLAB environment,
- real-time control experiments using demo algorithms,
- development, simulation and application of user-defined control algorithms.

MLS2EM Toolbox is distributed on a CD-ROM. It contains software and the *MLS2EM User's Manual*. The *Installation Manual* is distributed in a printed form.

1.2 Hardware and software requirements.

MLS Toolbox is distributed on a CD-ROM or usb stick. It contains the software and *MLS User's Manual*. The *Installation Manual* is distributed in a printed form.

Hardware

Hardware installation is described in the *Installation Manual*. It consists of:

- Electromagnet
- Ferromagnetic spheres
- Position sensor
- Current sensor
- Power interface
- RTDAC/USB2 I/O board. The board contains FPGA equipped with dedicated logic,
- Pentium or AMD based personal computer.

Software

- Microsoft W7/W10x64 and MATLAB 64 bit with Simulink, and Simulink Coder (RTW) toolboxes (not included),
- Third party compiler MS Visual C++ depending on Matlab's version
- Details at: https://www.mathworks.com/support/sysreq/previous_releases.html
- The TCP/IP protocol must be installed in the computer system,
- CD-ROM or USB stick with MLS software and e-manuals (*User's Manual* and *Installation Manual*)

Details of the required software are available at: http://www.inteco.com.pl/support/Software_requirements.htm

Real-time is supported by the RT-CON toolbox from INTECO (included in MLS Toolbox and transparent for a user).

1.3 Features of MLS

One can highlight a number of features of MLS, among them the following deserve our attention

- The aluminum construction
- Three ferromagnetic objects (spheres) with different weights
- An optical detector to sense the object position
- A coils current sensors
- A highly nonlinear system ideal for illustrating complex control algorithms
- A frictionless system
- The full integration with MATLAB®/Simulink®
- Real-time control under W7/W10

1.4 Typical teaching applications

- System Identification
- SISO, MISO, BIBO controllers design
- Intelligent/Adaptive Control
- Frequency analysis
- Nonlinear control
- Hardware-in-the-Loop
- Real-Time control
- Closed Loop PID Control

1.5 Software installation

Insert the installation CD and proceed step by step following displayed commands.

2 ML Main Window

The user has a rapid access to all basic functions of the MLS System from the *MLS Control Window*.

> **If the MATLAB R2018 or newer is used run the *rehash toolbox* command, close Matlab and open again.**

Then type:

mls2em_usb2_main

and then the *Magnetic Levitation Main* window opens (see Fig. 2).

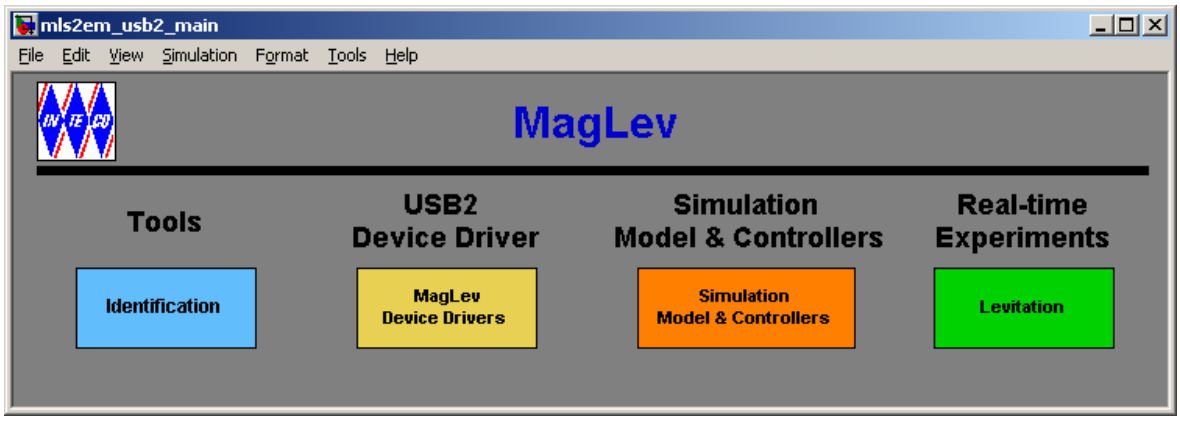


Fig. 2. The Magnetic Levitation Main window

In the mls2em_usb2_main window one can find: testing tools, drivers, models and demo applications. You can see a number of pushbuttons ready to use.

The mls2em_usb2_main window shown in Fig. 2 contains four groups of the menu items:

- Tools – identification
- USB2 Device Driver – MagLev device drivers
- Simulation model and controllers
- Real-time experiments – levitation

Section 2 is divided into four subsections. Under each button in the mls2em_usb2_main window one can find the respective portion of software

corresponding to the problem announced by the button name. These problems are described below in four consecutive subsections.

2.1 Identification

If we click the identification button the following window (see Fig. 3) opens. There are the default values of all parameters defined by the manufacturer. Nevertheless, a user is equipped with a number of identification tools. He can perform the identification procedures to verify and if necessary modify static and dynamic characteristics of MLS2EM.

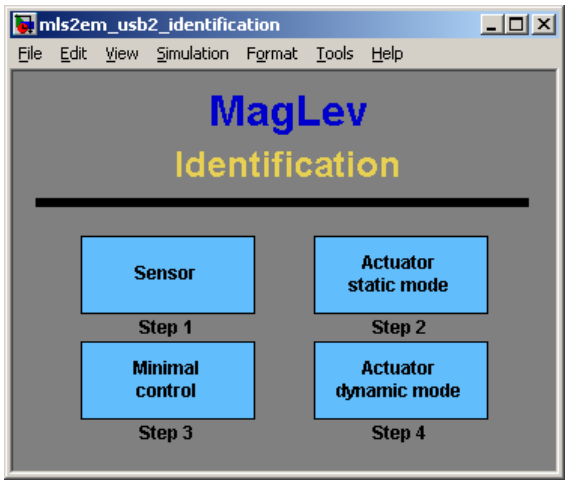


Fig. 3. The identification window

Four identification steps have been preprogrammed. They are described below.

2.1.1 Sensor

In this subsection the position sensor characteristics is identified.

If you click the Sensor button the following window opens (see Fig. 4)

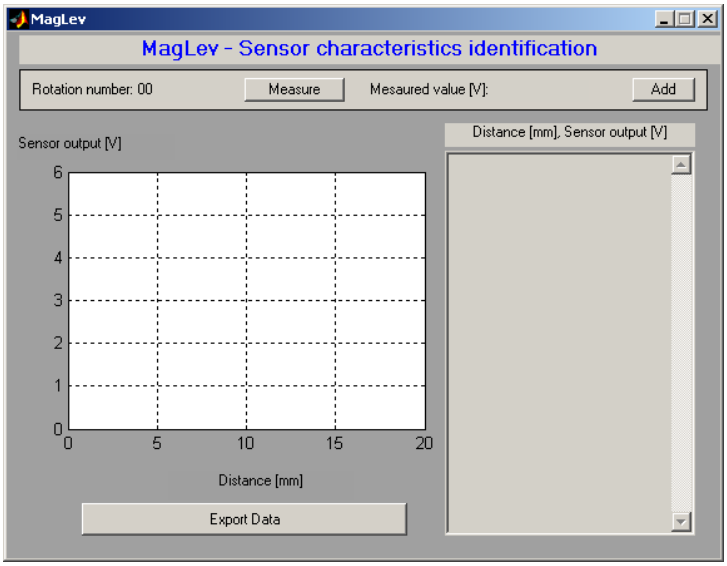


Fig. 4. Sensor signal in [V] vs. the sphere distance from the electromagnet in [mm]

The following procedure is required to identify the characteristics.

1. Screw in the screw bolt into the bottom electromagnet.
2. Screw in the black sphere and lock it by the nut. Notice, that **the sphere is fixed to the bottom electromagnet and frame recpectively!**
3. Turn round the screw so the sphere is in touch with the bottom of the top electromagnet.
4. Make sure that the power is on.
5. Start the measuring and registration procedure. It consists of the following steps:
6. Push the *Measure* button – the voltage from the position sensor is stored and displayed as *Measured value [V]*. One can correct this value by measuring it again.
    - Push the *Add* button – the measured value is added to the list. A rotation number value is automatically enlarged by one (see Fig. 5).

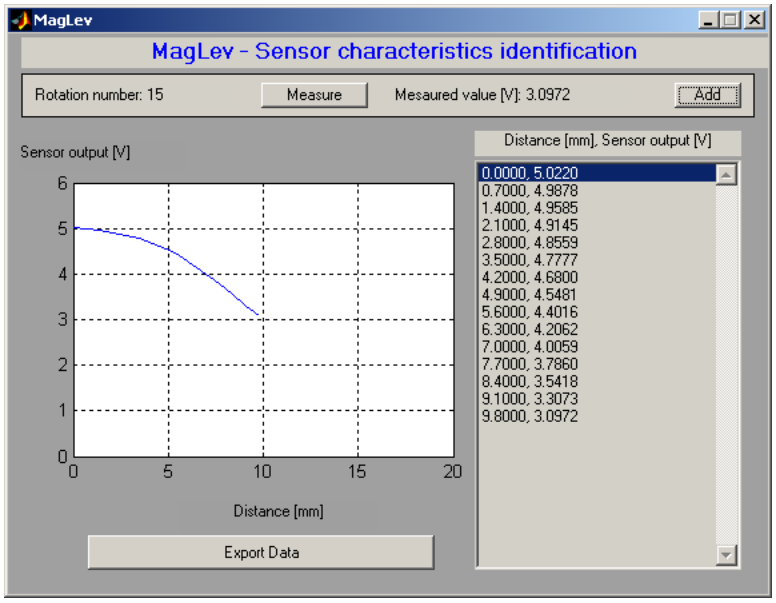


Fig. 5. Characteristics of the sphere position sensor

- Manually make one full rotation of the screw.
- Repeat three last steps so many times as none change in the voltage vs. position characteristics is observed.

7. Push the *Export Data* button – the data are written to the disc (see Fig. 6). Data are stored in the *mls2em_usb2_Sensor.mat* file as the *SensorData* structure with the following signals: *Distance_mm*, *Distance_m* and *Sensor_V*.

In the Simulink real-time models the above characteristics is used as a Look-Up-Table model. The block named *Position scaling* is located inside the device driver block of MLS2EM (see Fig. 7). Notice, that the characteristics shows meters vs. Volts. In Fig. 6 there were shown Volts vs. meters. It is obvious that we require the inverse characteristics because we need to define the output as the position in meters.

Notice that the characteristics can be different due to manufacturing process and light conditions.

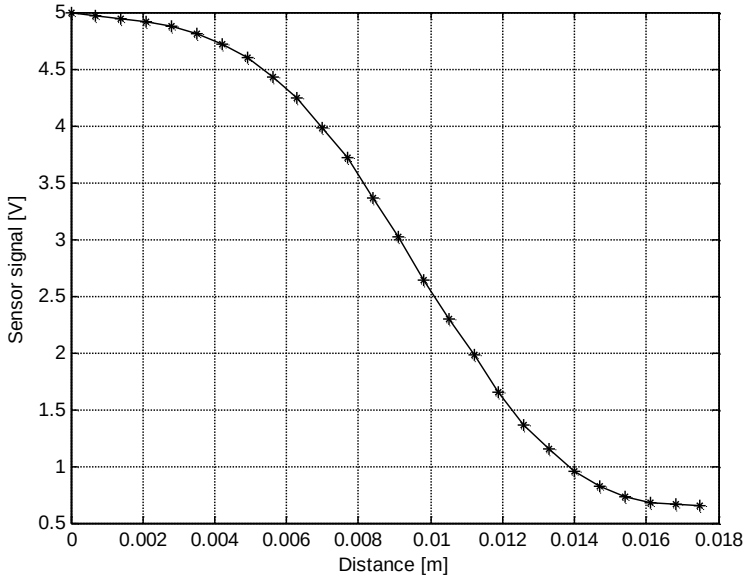


Fig. 6. The sensor characteristics after being measured and exported to the disc

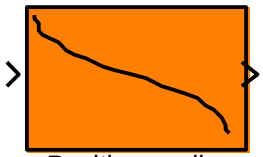


Fig. 7. The Simulink Look Up Table model representing the position sensor characteristics

If we click this block the window shown in Fig. 8 opens. Any time you like to modify the sensor characteristics you can introduce new data related to the voltage measured by the sensor. The voltage corresponds to the distance of the sphere set by a user while the identification procedure is performed. The sensor characteristics is loaded from the *mls2em_usb2_sensor.mat* file which has been created during the identification procedure. If the curve of the *Position scaling* block is not visible please load the file with data.

The sensor characteristics can be approximated by a polynomial of a given order. For example, we can use a fifth order polynomial.

$$P(x) = p_5 x^5 + \ldots + p_0$$

$p_5 = -25697073504.59$, $p_4 = 1245050011.25$, $p_3 = -18773635.92$, $p_2 = 79330.24$, $p_1 = -150.21$ and $p_0 = 5.015$.

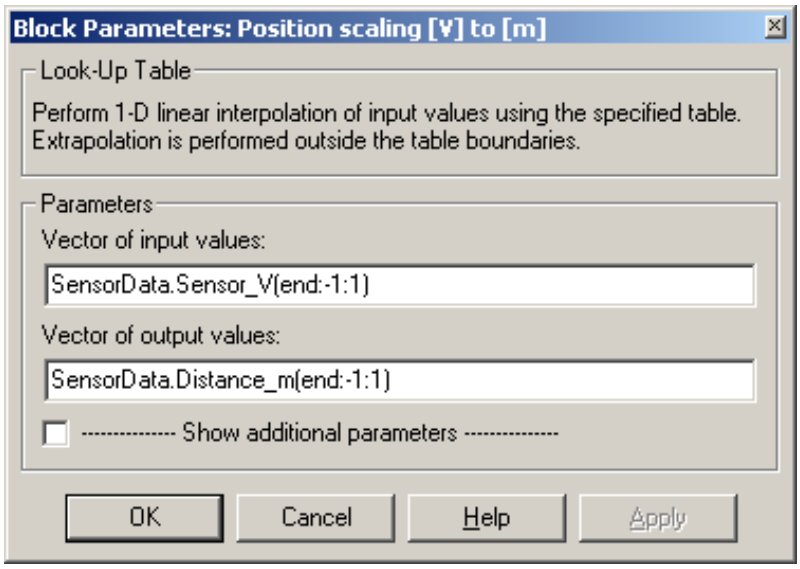


Fig. 8. Look-Up Table to be fulfilled with vectors of input and output values

The approximated polynomial (the red line) is shown in Fig. 9. The polynomial approximation will be not used in this manual due to the fact that the entire model is built in Simulink. Therefore we recommend to model the characteristics as a Look-Up Table block (see Fig. 7 and Fig. 8).

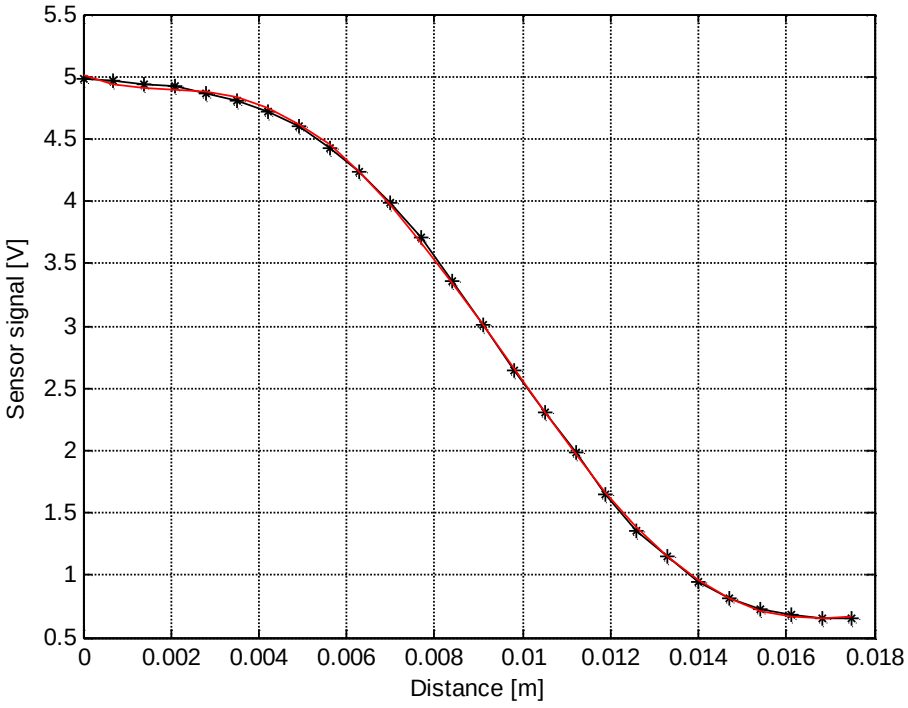


Fig. 9. The sensor characteristics approximated by the fifth order polynomial

2.1.2 Actuator static mode

In this subsection we examine static features of the actuator i.e. the electromagnet. Notice, that **the sphere is not present**!

Click the *Actuator static mode* button and the window shown in Fig. 10 opens.

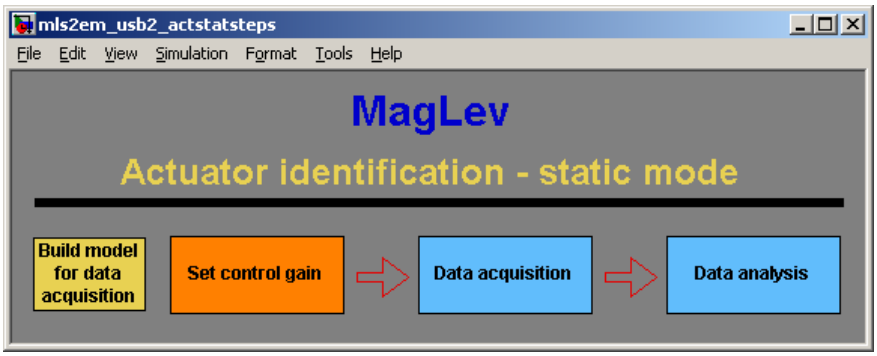


Fig. 10. Identification window of a static current/voltage characteristics

Now, we can perform button by button the operations depicted in Fig. 10. We begin from the *Build model for data acquisition* button. The window of the real-time task shown in Fig. 11 opens and the RTW build command is executed (the executable code is created).

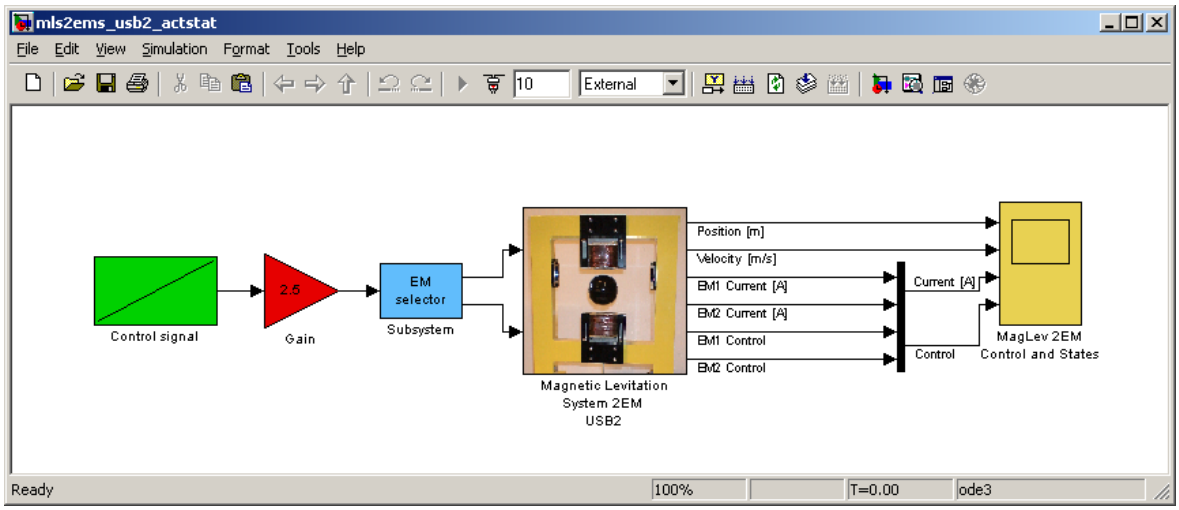


Fig. 11. Real-time model built to examine the current in the electromagnetic coil

Click the *Set control gain* button. It results in activation of the model window and the following message is displayed (see Fig. 12):

Fig. 12. Message – Set the “Control Gain”

In Fig. 11 one can notice the *Control signal* block. In fact the control signal increases linearly. We can modify the slope of this signal changing the *Control Gain* value.

Click the *Data acquisition* button. Within 10 seconds data are acquired and stored in the workspace.

Click the *Data analysis* button. The collected values of the coil current are displayed in Fig. 13.

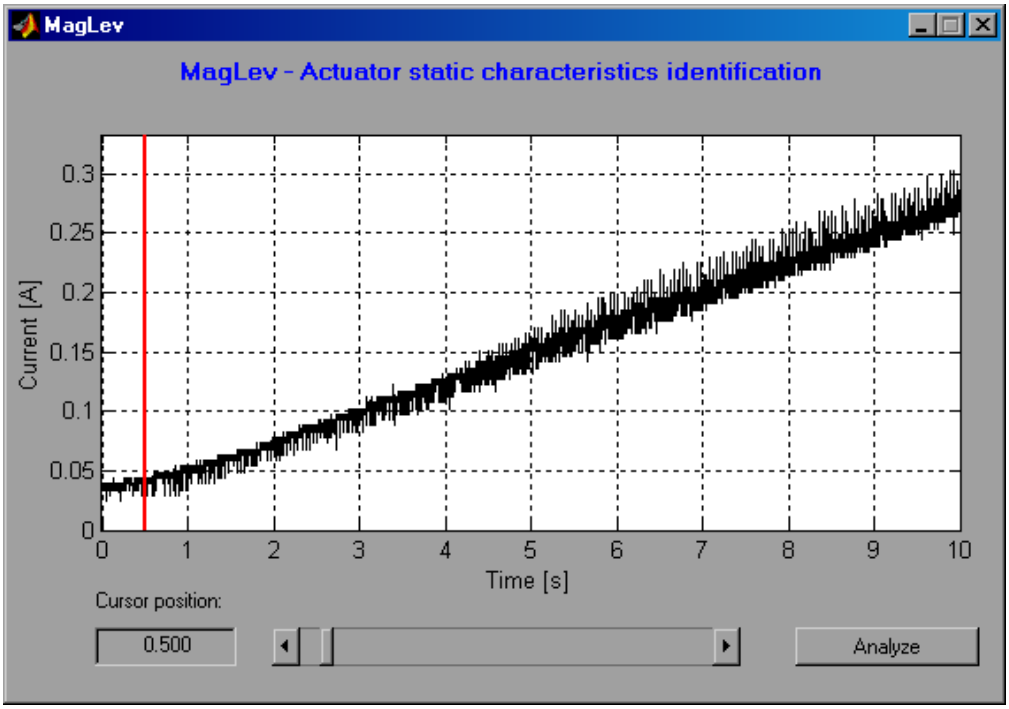


Fig. 13. Current in the electromagnetic coil

The characteristics is linear except a small interval at the beginning. We can locate the cursor at the point where a new line slope starts (see the red line in the picture). We can move the cursor in two ways: by writing down a value into the edition window or by drugging the slider. In this way the current characteristics is prepared to be analyzed in the next step. The line is divided into two intervals: the first – from the beginning of measurements to the cursor and the second – from the cursor to the end of measurements.

After setting the cursor position, consequently, click the *Analyze* button. The following message (see Fig. 14) appears. We obtain the dead zone values corresponding to the control and current. The constants *a* and *b* of the linear part are the parameters of the line equation: $i(u) = a\,u + b$.

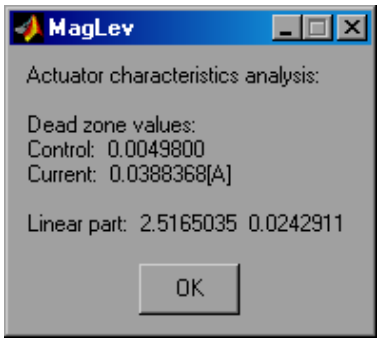


Fig. 14. Coefficients of the actuator characteristics

These parameters, namely: $u_{MIN} = 0.00498$, $x_{3MIN} = 0.03884$, $k_i = 2.5165$ and $c_i = 0.0243$ are going to be used in the simulation model in section 2.3.1 (see the differential equations parameters).

To obtain a family of static characteristics for linear controls with different slopes we repeat the following experiment. We apply a PWM voltage signal in the time interval from 0 to 10 s. The PWM duty cycles for the subsequent ten experiments are varying linearly in the ranges: [0, 0.1], [0, 0.2], ..., [0, 1.0] (see Fig. 16).

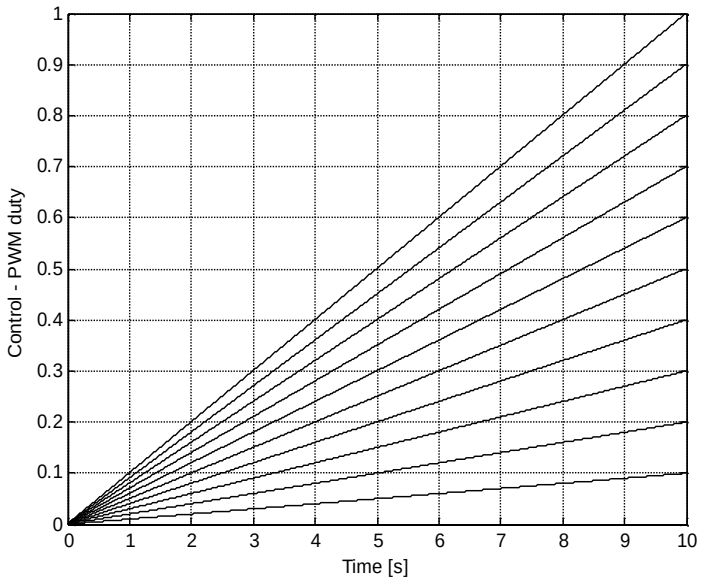


Fig. 15. Family of the input (PWM) characteristics

Consequently, we obtain diagrams of the currents corresponding to ten experiment (see Fig. 16). Each characteristics is approximated by a polynomial of the first order. Finally the entire current vs. PWM duty cycle relation is depicted (black points) in Fig. 17. The red line represents the linear approximation of measurements. We obtain the following numerical values of linear characteristics:

$i(u) = k_i u + c$; a = 2.60798876298869, b = –0.01077522109792.

The constant *c* is obtained for $u = 0$. The family of linear characteristics is used to obtain the coefficients k_i vs. control *u*.

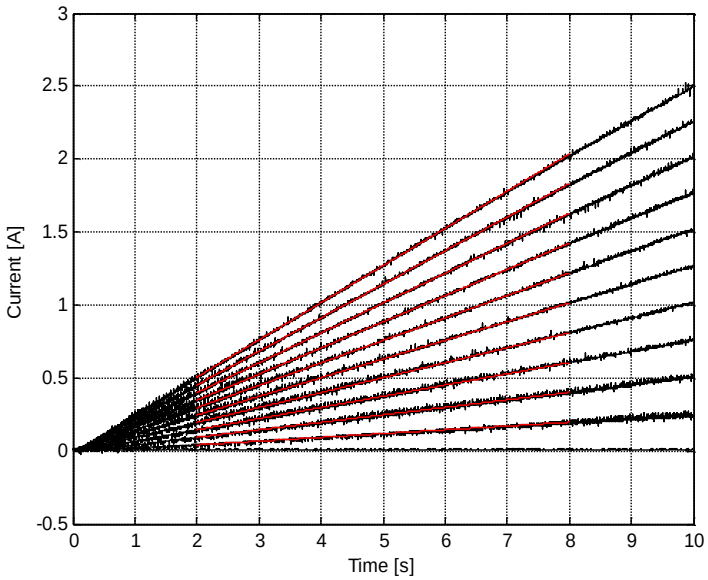


Fig. 16. Family of the output (current) characteristics

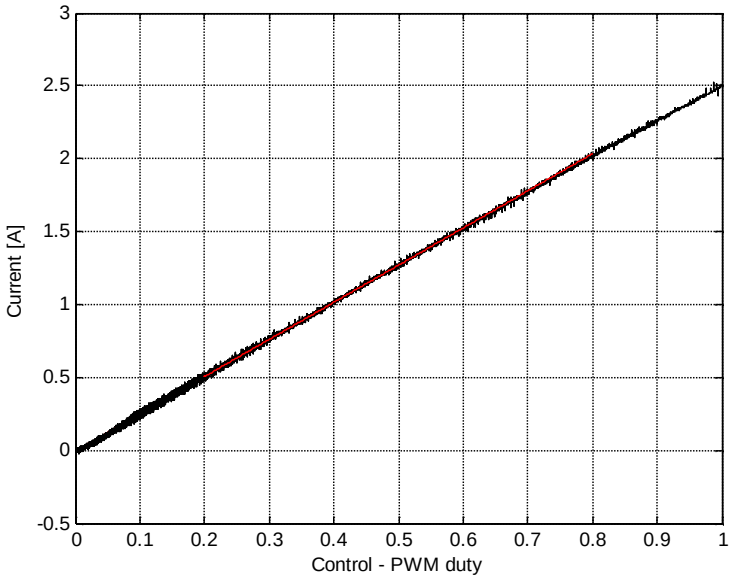


Fig. 17. Current vs. PWM duty cycle

Using the *EM Selector* block select the electromagnet to be controlled and repeat the whole procedure for both electromagnets.

2.1.3 Minimal control

In this subsection we examine the minimal control to cause a forced motion of the sphere from the bottom electromagnet toward the upper electromagnet

against the gravity force. Notice, that in this experiment **the sphere is not levitating**! It is kept nearby the electromagnet in the operating range.

Click the *Minimal control* button and the window shown in Fig. 18 opens.

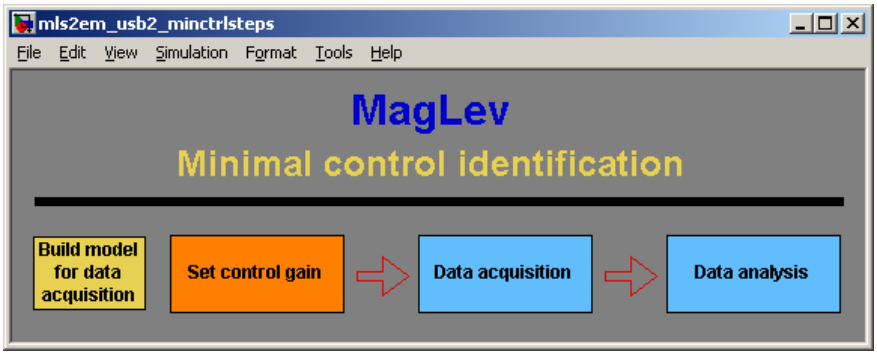


Fig. 18. Window to identify the minimal control vs. distance (between the sphere and electromagnet)

Now, we proceed button by button the operations depicted in Fig. 18 similarly to the procedure described in the previous subsection. We begin from the *Build model for data acquisition* button. The window of the real-time task shown in Fig. 19 opens.

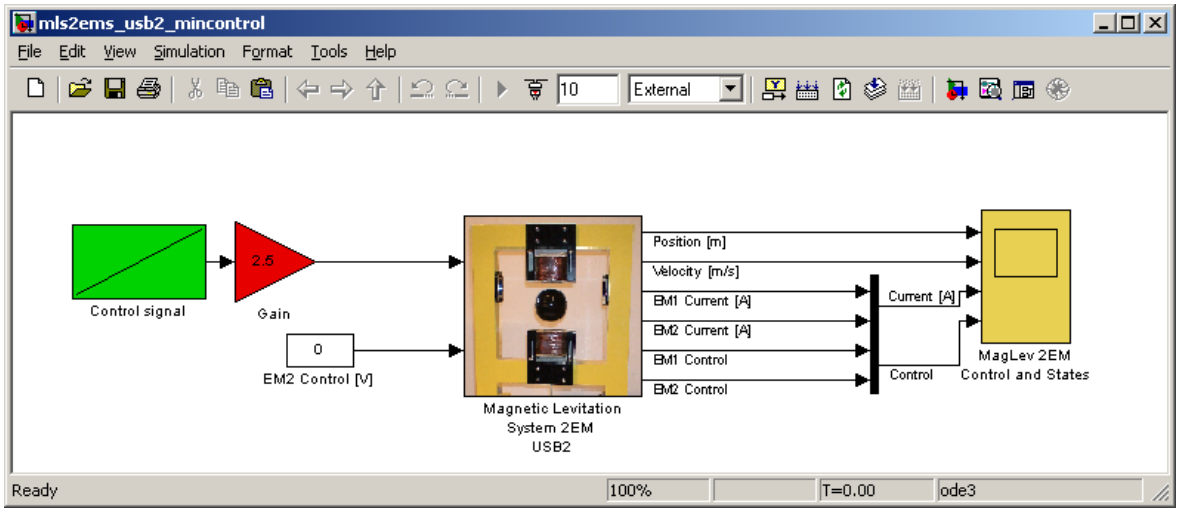


Fig. 19. Real-time model built to examine the minimal electromagnetic force

Click the *Set control gain* button. It results in activation of the model window and the following message is displayed (see Fig. 20).

Fig. 20. Message – set the “Control Gain”

It means that we can set a duty cycle of the control PWM signal. The sphere is located on the support and the experiment starts. Click the *Data acquisition* button. A forced motion of the ball toward the electromagnet begins.

Click the *Data analysis* button. The collected values of the ball position are displayed in Fig. 21.

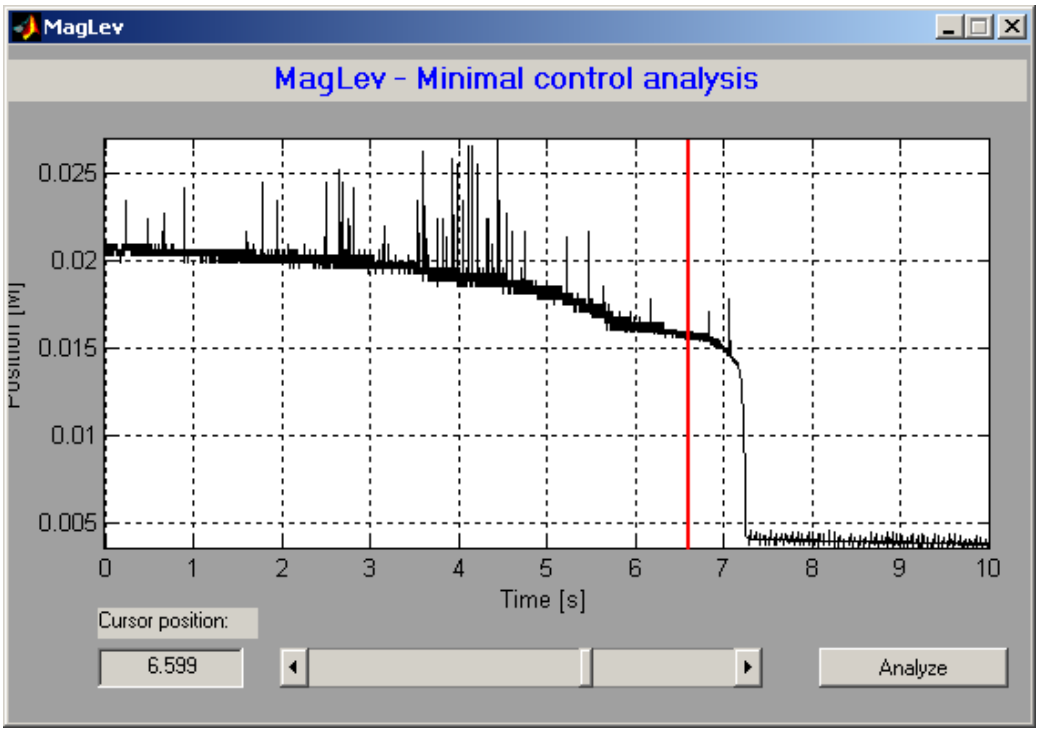


Fig. 21. The sphere motion

The sphere motion is visible. We can locate the cursor at the point slightly before a position jump occurs (takes place) (see the red line in the picture). We can move the cursor in two ways: by writing down a value into the edition window or by drugging the slider. In this way the acquired data are prepared to be analyzed in the next step.

After setting the cursor position, consequently, click the *Analyze* button. The following message (see Fig. 22) appears. This information means that the sphere located 15.82 mm from the electromagnet begins to move toward it when the PWM control over-crosses the 0.49485 duty cycle value.

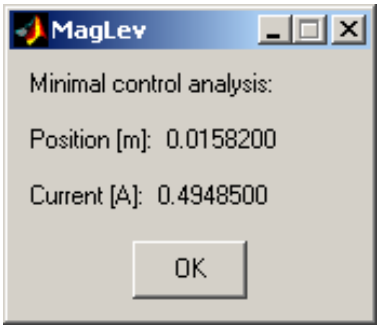


Fig. 22. Message of the experiment results

2.1.4 Actuator dynamic mode

In this subsection we examine dynamic features of the actuator i.e. the electromagnet. It means that the moving sphere generates an electromotive force (EMF). EMF diminishes the current in the electromagnet coil. Click the *Actuator static mode* button and the window shown in Fig. 23 opens.

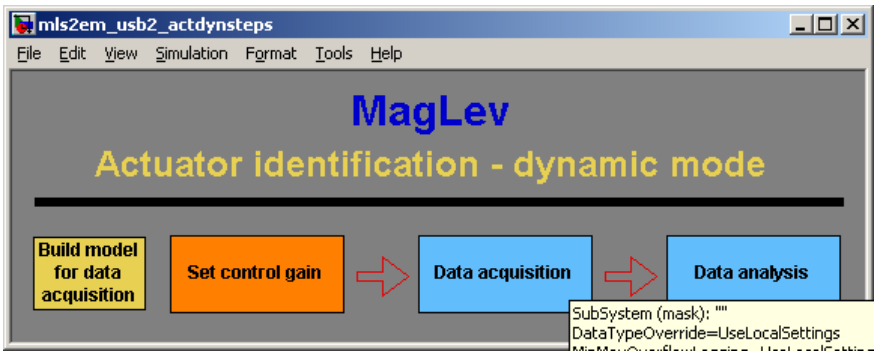


Fig. 23. Identification window of a dynamic current/voltage characteristics

A user should perform three experiments: without the sphere (*Without ball*), with the sphere located on the bottom electromagnet (*Ball on EM*) and with the sphere fixed to the rigid screw (*Ball fixed*).

We begin from the *Build model for data acquisition* button. The window of the real-time task shown in Fig. 24 opens. We have to set the control gain. If we are going to modify the control magnitude then we set the default gain to 1 and the subsequent duty cycles to: 0.25, 0.5, 0.75 and 1. Click the *Data acquisition* button and save data under a given file name.

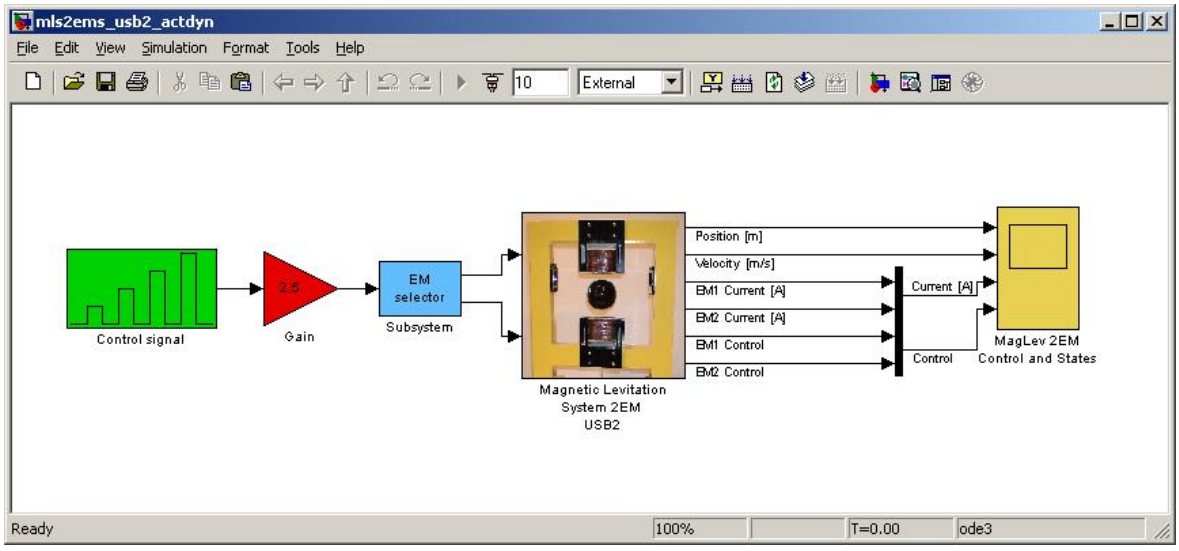


Fig. 24. Real-time model built to examine EMF influence on the coil current

Click the *Data analysis* button. It calls the *mls2em_find_curr_dyn.m* file. The following window opens (see Fig. 25). The parameters optimization procedure starts. The optimization routine is based on the *mls2em_current_m.mdl* model.

When *mls2em_find_curr_dyn.m* runs the optimization function *fminsearch* is executed. *Fminsearch* uses the *mls2em_opt_current.m* file.

The k_i and f_i parameters are iteratively changed during the optimization procedure. The current curve is fitted four times. This is due to the control signal form.

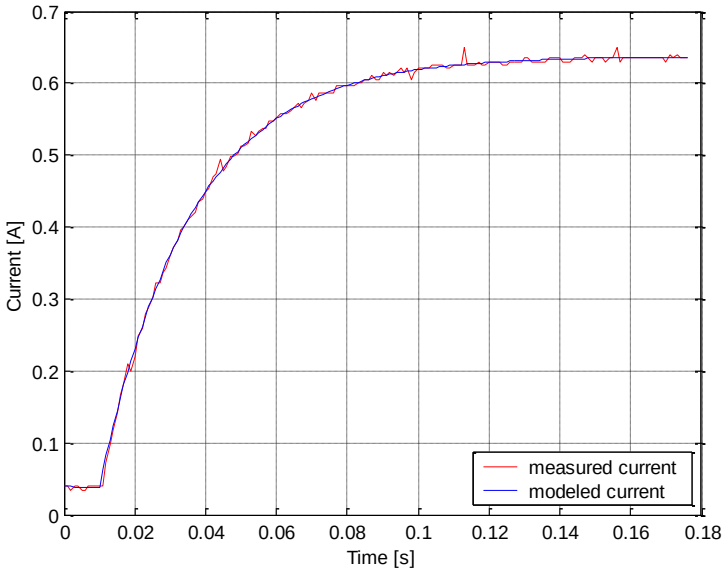


Fig. 25. Current curve – the fitting result of the optimization procedure

Finally the information about the mean values is displayed (see Fig. 26). The advanced user can use the functions code to perform a detailed analysis.

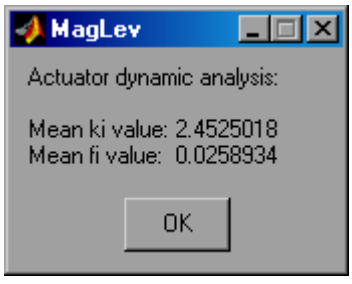


Fig. 26. Optimization results

Using the *EM Selector* block select the electromagnet to be controlled and repeat the whole procedure for both electromagnets.

2.2 MagLev device drivers

The driver is a software go-between for the real-time MATLAB environment and the RT-DAC4/USB acquisition board. The control and measurements are driven. Click the *Device Drivers* button in the *Magnetic Levitation Main* window. The following window opens (see Fig. 27).

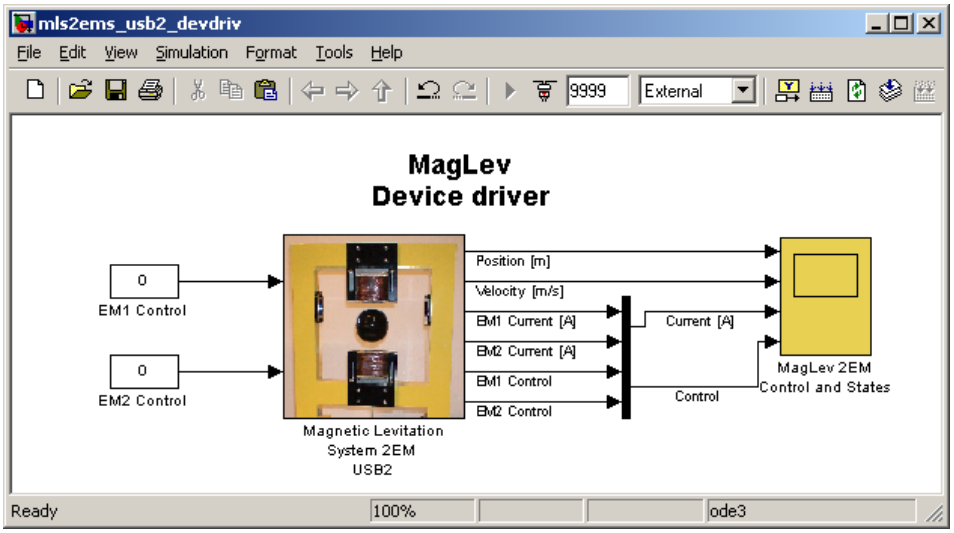


Fig. 27. USB MagLev device driver window

Notice that the scope block writes data to the *MLS2EMExpData* variable defined as a structure with time. The structure consists of the following signals: Position [m], Velocity [m/s], EM1 Current [A] and EM2 Current [A], EM1 Control [PWM duty 0÷1] and EM2 Control [PWM duty 0÷1]. The interior of the *Magnetic Levitation System 2EM USB2* block (it means the interior of the driver block) is shown in Fig. 28.

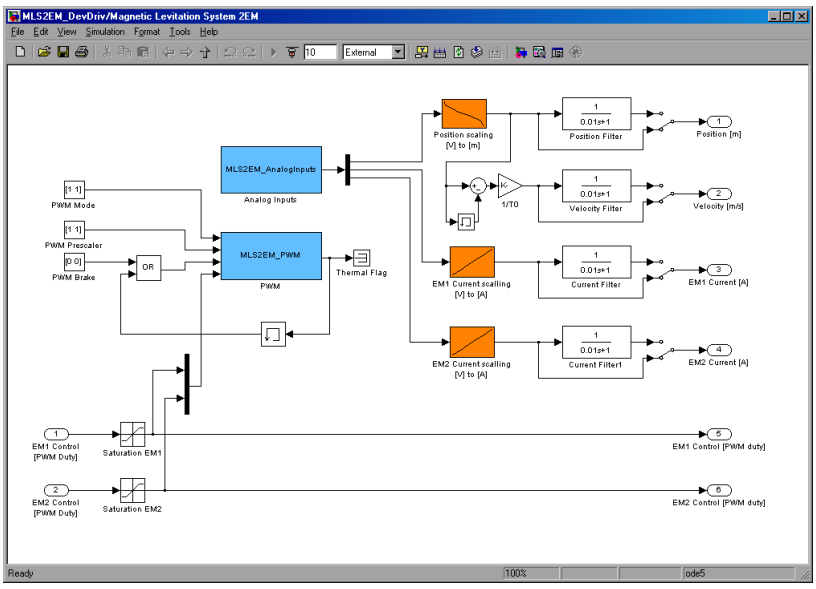


Fig. 28. Interior of the driver block

In fact there are two drivers: *MLS2EM USB2 Read* and *MLS2EM USB2 Write*. There are also two characteristics: the ball position [m] vs. the position sensor voltage [V] and the coil current vs. the current sensor voltage [V]. The driver uses functions which communicate directly with a logic stored at the RT-DAC/USB2 module. When one wants to build his own application one can copy this driver to a new model. The interior of the device drivers block is presented in Fig. 29 and Fig. 30. The information between the MATLAB/Simulink and RT-DAC/USB2 board is exchanged by the driver files *mls2em_usb2_pwm_read* and *mls2em_usb2_pwm_write* devoted to read and write operations via USB.

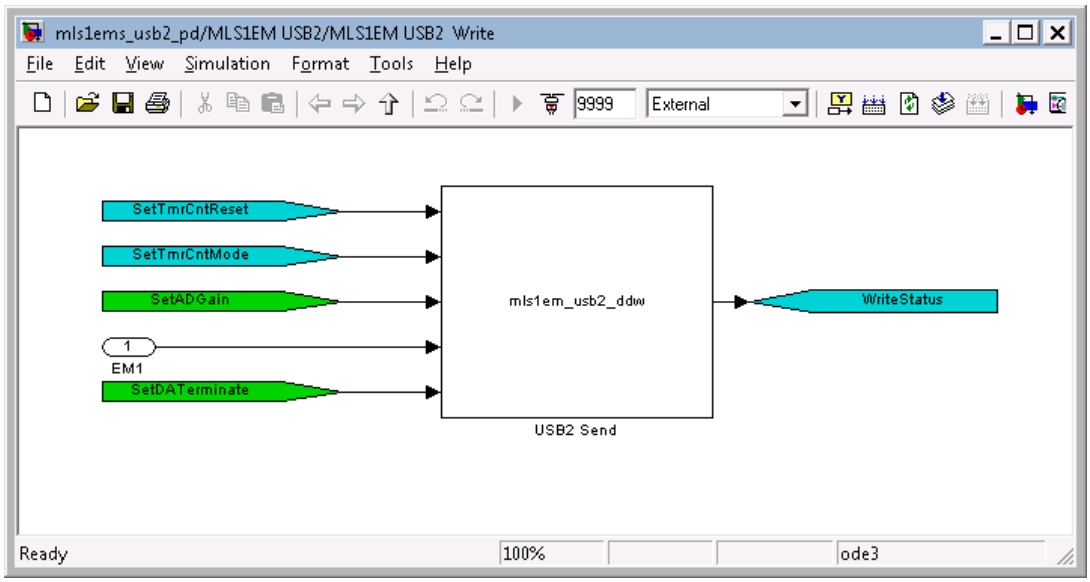


Fig. 29. USB Send device driver file connections

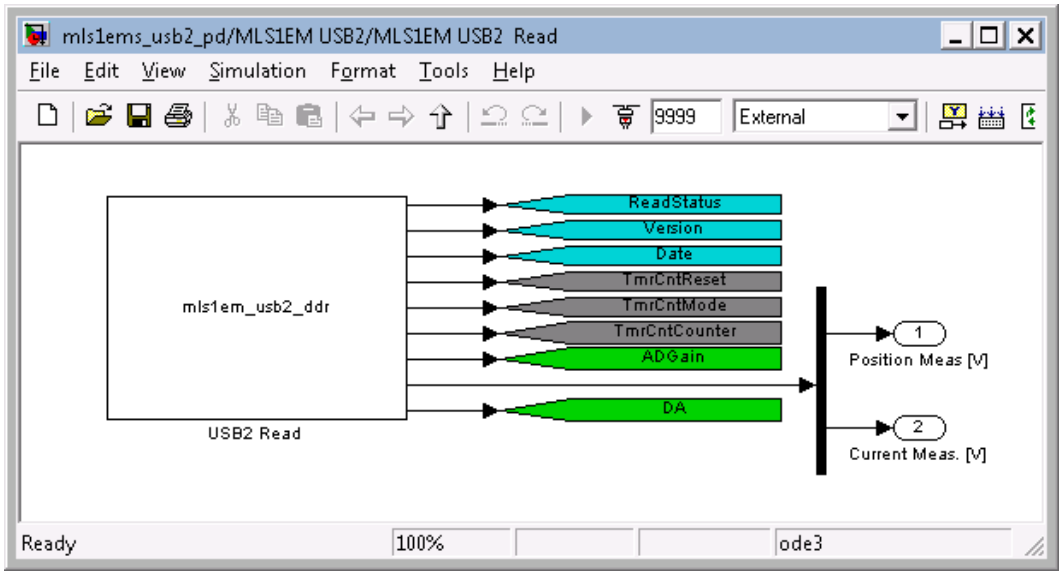


Fig. 30. USB Read device driver file connections

Do not introduce any changes inside the original driver. They can be introduced only inside its copy!!! Make a copy of the installation CD.

The Simulink Look-Up-Table model named *Position scaling* (see Fig. 7) representing the position sensor characteristics has been already described. Now let us present the second Simulink Look-Up-Table model named *Current scaling* (see Fig. 31).

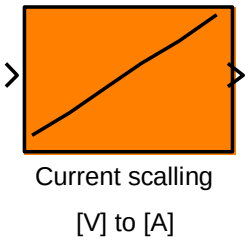


Fig. 31. The Simulink Look-Up-Table model representing the current sensor characteristics

To build the above characteristics it is necessary to measure the current of the electromagnet coil. The algorithm in the computer is the source of the desired value of the control in the form of the voltage PWM signal. This PWM is the input voltage signal transferred to the hardware driver chip of the power interface. Due to a high frequency of the PWM signal the measured current values correspond to the average current value in the coil. This characteristics has been built by the manufacturer. It is not recommended to repeat measurements by a user because to do so one must unsolder the input wires of the electromagnet. On the basis of the data given in the table below one can generate his own characteristics. For a fixed PWM frequency and a variable duty cycle the coil amperage is measured. The measured data are given below in the table.

PWM duty cycle	amperage [A]	voltage [V]
0	0	0.374811
0.1	0.25	0.262899
0.2	0.51	0.510896
0.3	0.77	0.752465
0.4	1.02	0.993620
0.5	1.28	1.229133
0.6	1.52	1.459294

0.7	1.74	1.651424
0.8	1.99	1.875539
0.9	2.21	2.076814
1	2.43	2.269865

The current [A] vs. voltage [V] characteristics is shown in Fig. 32.

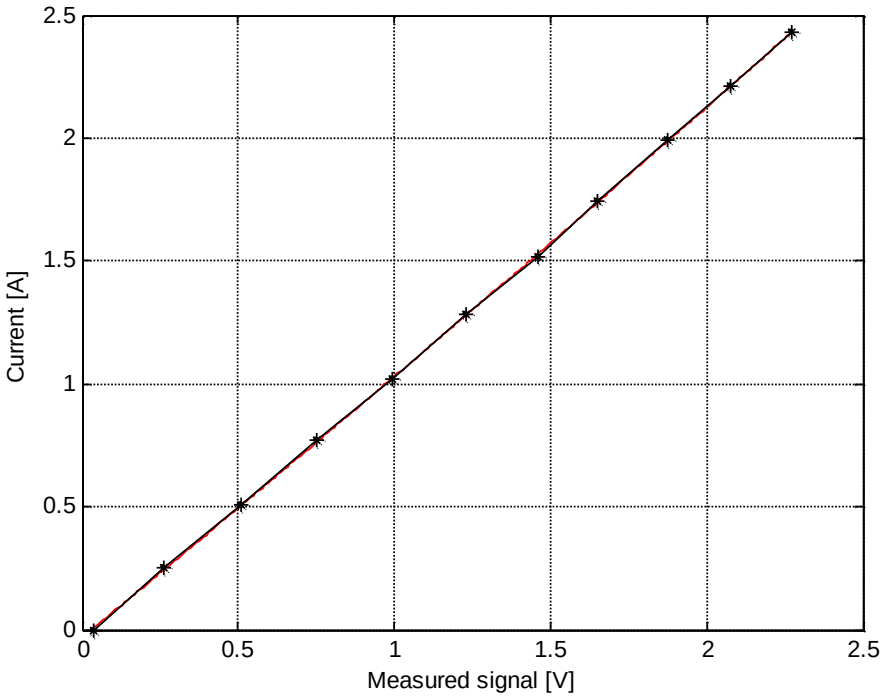


Fig. 32. Current vs. voltage characteristics approximated by the red curve

The characteristic can be approximated by a polynomial of the second order:

$$I(U) = a_2 U^2 + a_1 U + a_0$$

where:

I – current,

U – voltage from the A/D converter

a_0, a_1, a_2 - identified parameters of the polynomial

$a_2 =$ **0.0168**

$a_1 =$ **1.0451**

$a_0 =$ **–0.0317**

2.3 Simulation Model & Controllers

Click the *Simulation Model & Controllers* button in the *Magnetic Levitation Main* window. The following window opens (see Fig. 33).

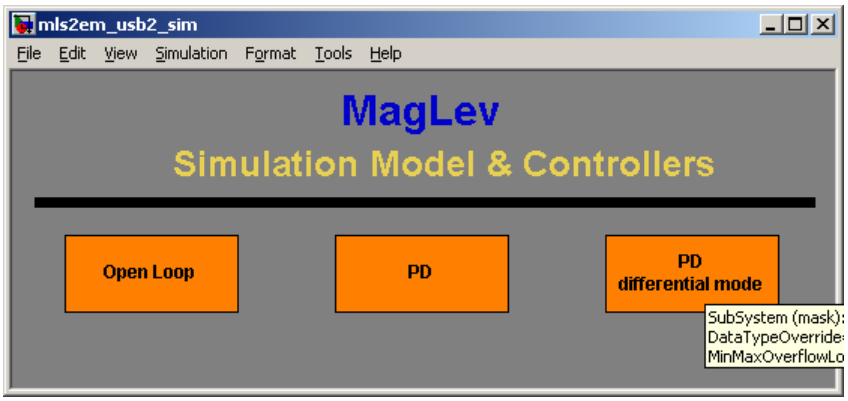


Fig. 33. Simulation Model & Controllers window

2.3.1 Open Loop

- **Simulink model**

Next, you can click the first *Open Loop* button. The following window opens (see Fig. 34). Notice that the scope block writes data to the *MLS2EMSimData* variable defined as a structure with time. The structure consists of the following signals: Position [m], Velocity [m/s], Currents [A], Controls [PWM duty 0÷1].

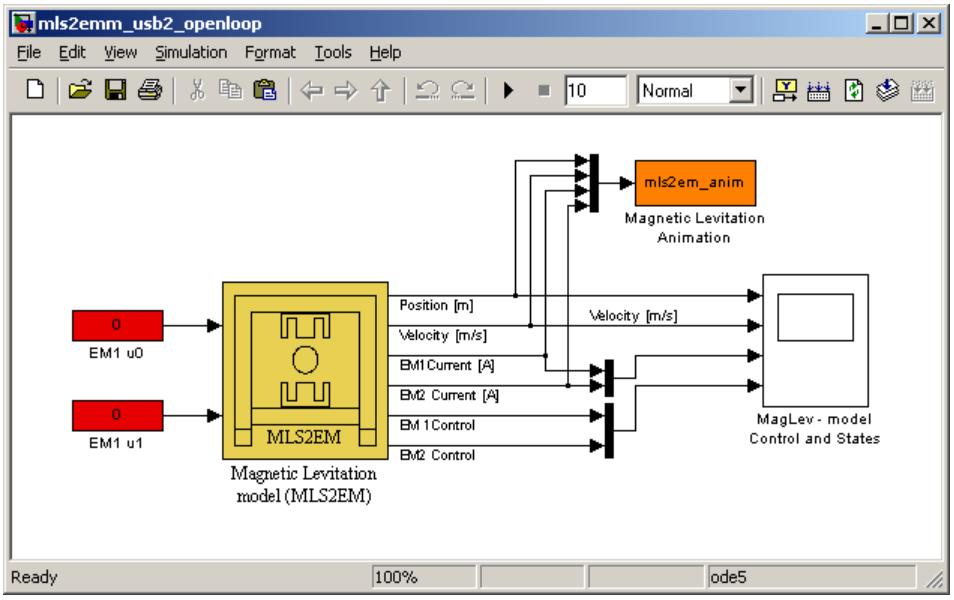


Fig. 34. Open-loop simulation

If you click the *Magnetic Levitation model* block the following mask opens (see Fig. 35).

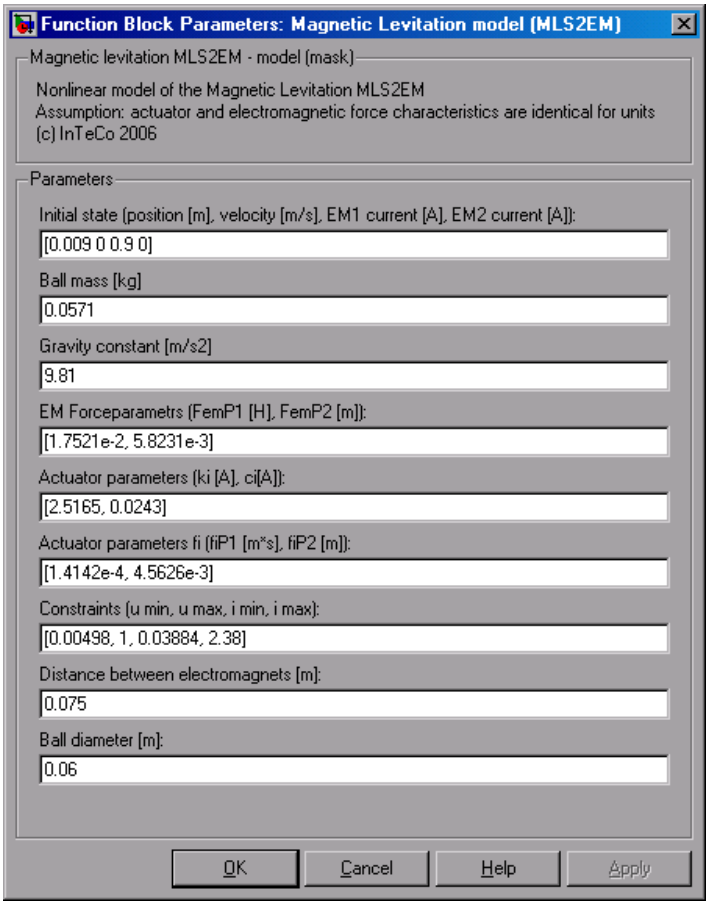


Fig. 35. Mask of the Magnetic Levitation model MLS2EM

In Fig. 34 enter into the *File* option and choose *Look under mask*. The interior of the *Magnetic Levitation model (MLS2EM)* block shown in Fig. 36 opens.

Please note that we assume that the actuator and electromagnetic force characteristics are the same for both units.

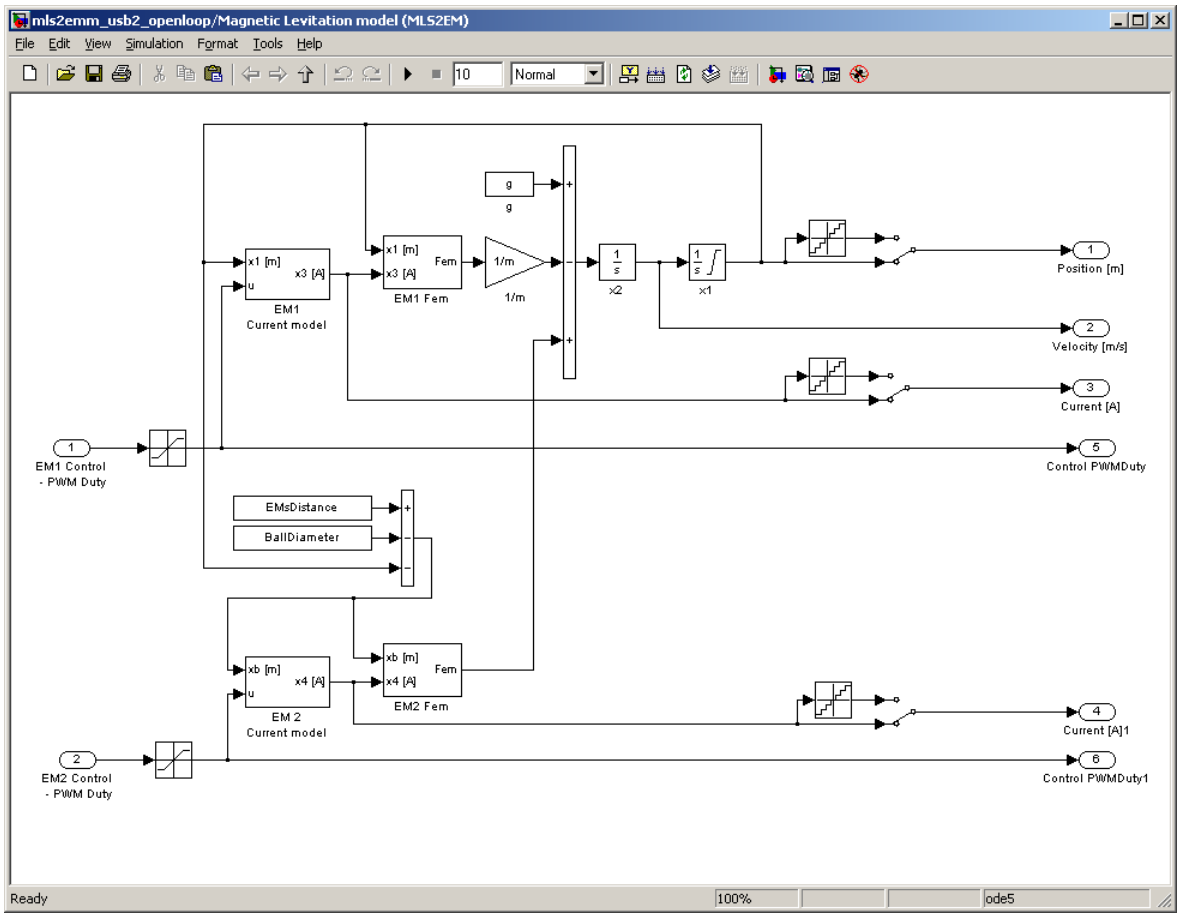


Fig. 36. Interior of the MLS2EM model

Notice two integrator blocks in Fig. 36. In fact we deal with third order dynamical system. The third integrator related to the coil current is visible in Fig. 37.

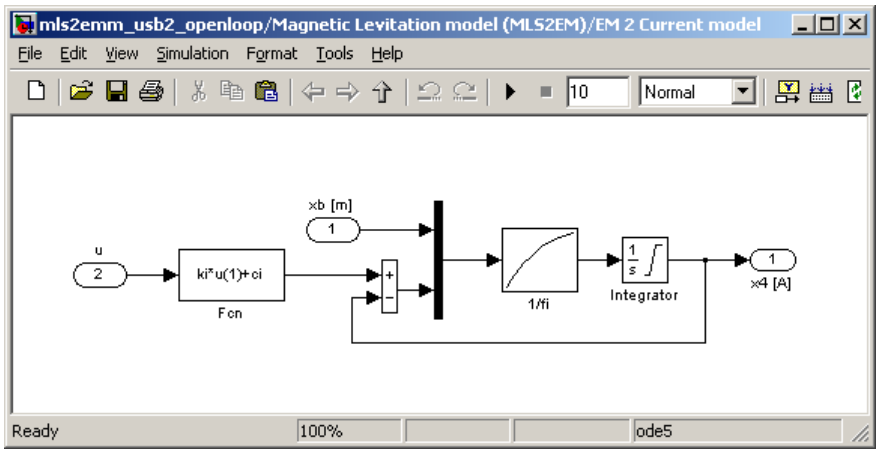


Fig. 37. Interior of the *Current model* block

The Simulink model is also equipped with the animation block. When a simulation starts the following window opens (see Fig. 38). The animation screen is updated in every sample time. All state variables: the ball position and velocity, and also the coil currents are animated.

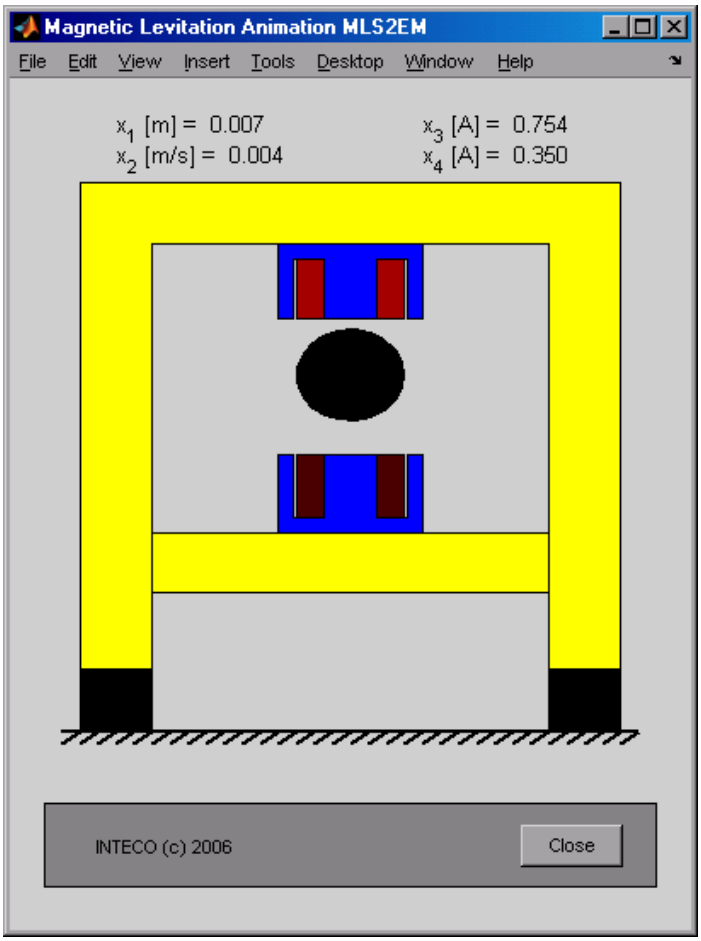


Fig. 38. MLS2EM animation

3 Real-time model in MATLAB version R2019b or newer

This section describes how to compile (build) and run a real-time model for new versions of Matlab _ R2019b or newer. The *Simulink Coder* and *RT-CON* toolboxes are used.

To build the system that operates in the real-time mode the user has to:

- create a Simulink model of the control system which consists of *Device Driver* and other blocks chosen from the Simulink library,
- build the executable file compiling model,
- start the real-time code.

The description in this section contains only the differences from the previous versions of MATLAB. So reading the previous chapter carefully is useful to understand the process of creating and running real time.

3.1 Creating a model

The simplest way to create a Simulink model of the control system is to use one of the models included in the INTECO's software. as a template. For example, click *Maglev Device Driver* model in the *Magnetic Levitation Main* window (Fig.2) The *Maglev Device Driver* Simulink model is shown in Fig. 3.1.

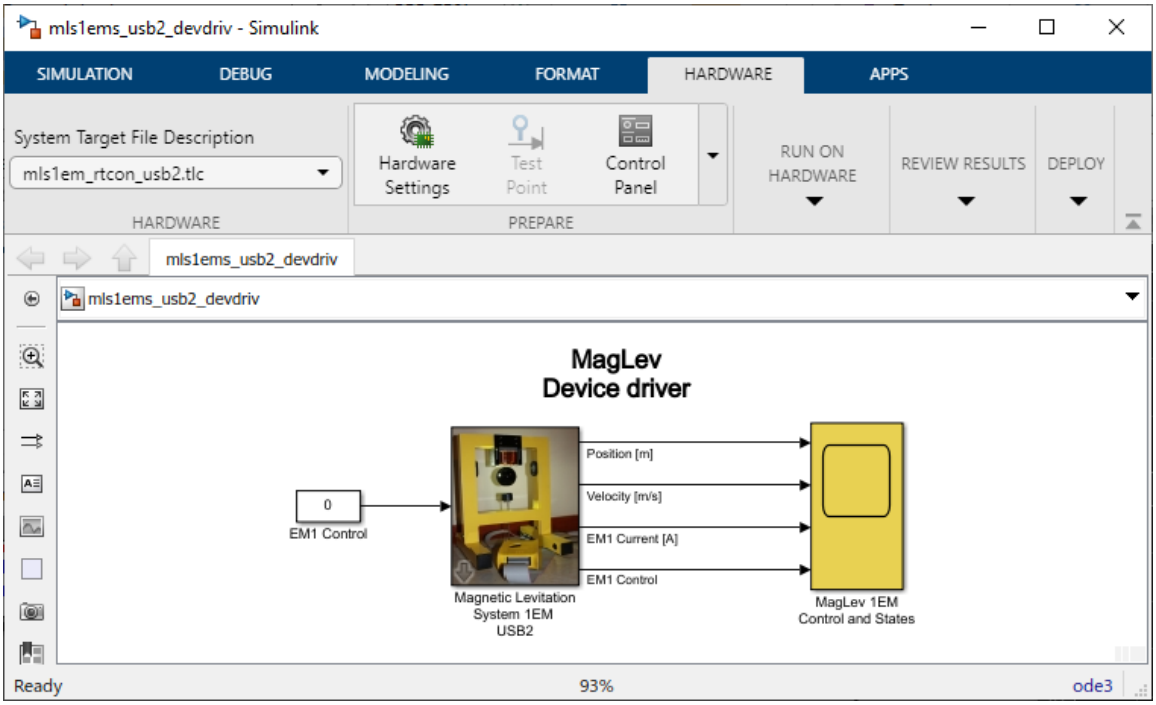


Fig. 3.1. *Maglev Device Driver* real-time model

Now, you can modify the model. You get absolute freedom to develop your own controller. Remember to leave the *Magnetic Levitation System 1EM USB2* block in the window. This is necessary to work in the real-time environment.

Though it is not obligatory, we recommend you add at least one scope. You need a scope to watch how the system runs.

Creating your own model on the basis of an old example ensures that all-internal options of the model are set properly. These options are required to

compiling and linking in a proper way. See at Fig. 3.2 and Fig. 3.3. To build real-time code a special files shown in *System target file* and *Template file* sections are required. These files are included to the INTECO's software.

You can apply most of the blocks from the Simulink library. However, some of them cannot be used (see Simulink Coder reference manuals).

When the Simulink model is ready, click the *Hardware Settings* option and next click the *Code Generation* button. The window presented in Fig. 3.2 opens. Select *Interface* button to see the external mode options at Fig. 3.3. The system target file name is *rtcon_tras_USB2.tlc*. It manages the code generation process.

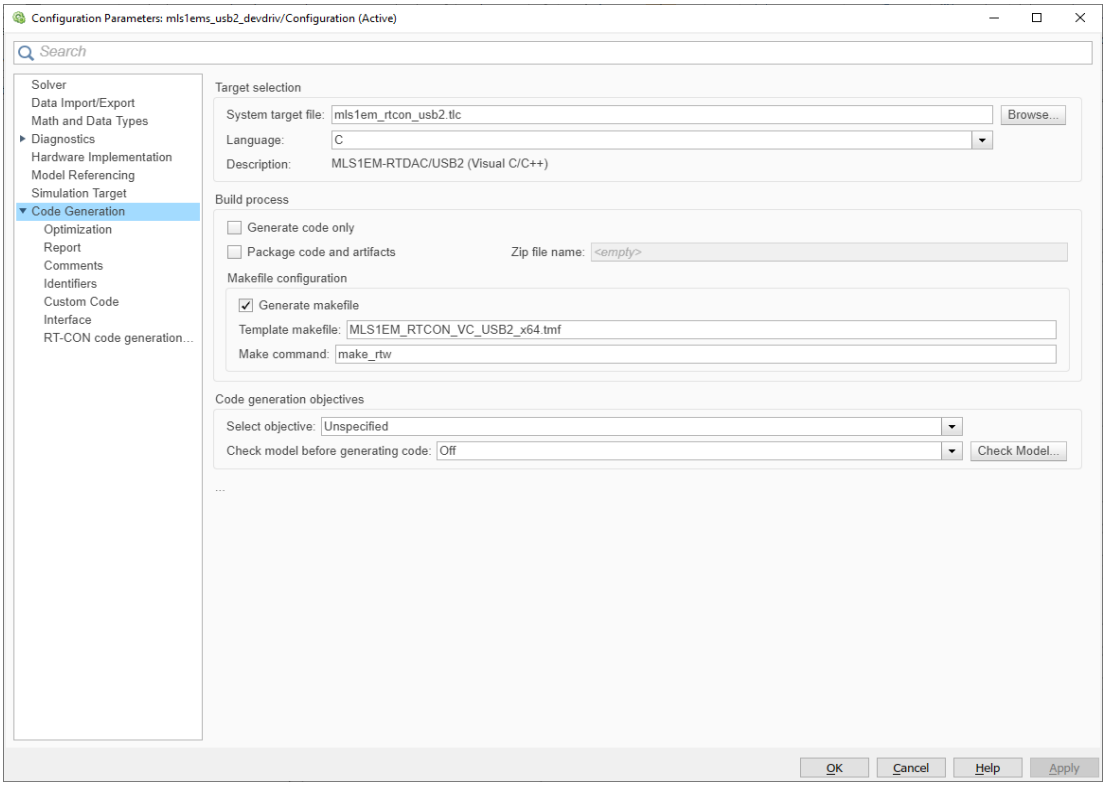


Fig. 3.2 Internal code generation options

The *MLS2EM_RTCON_vc_us2bx64.tmf* template make-file is devoted to C code generation using the Visual C++ compiler.

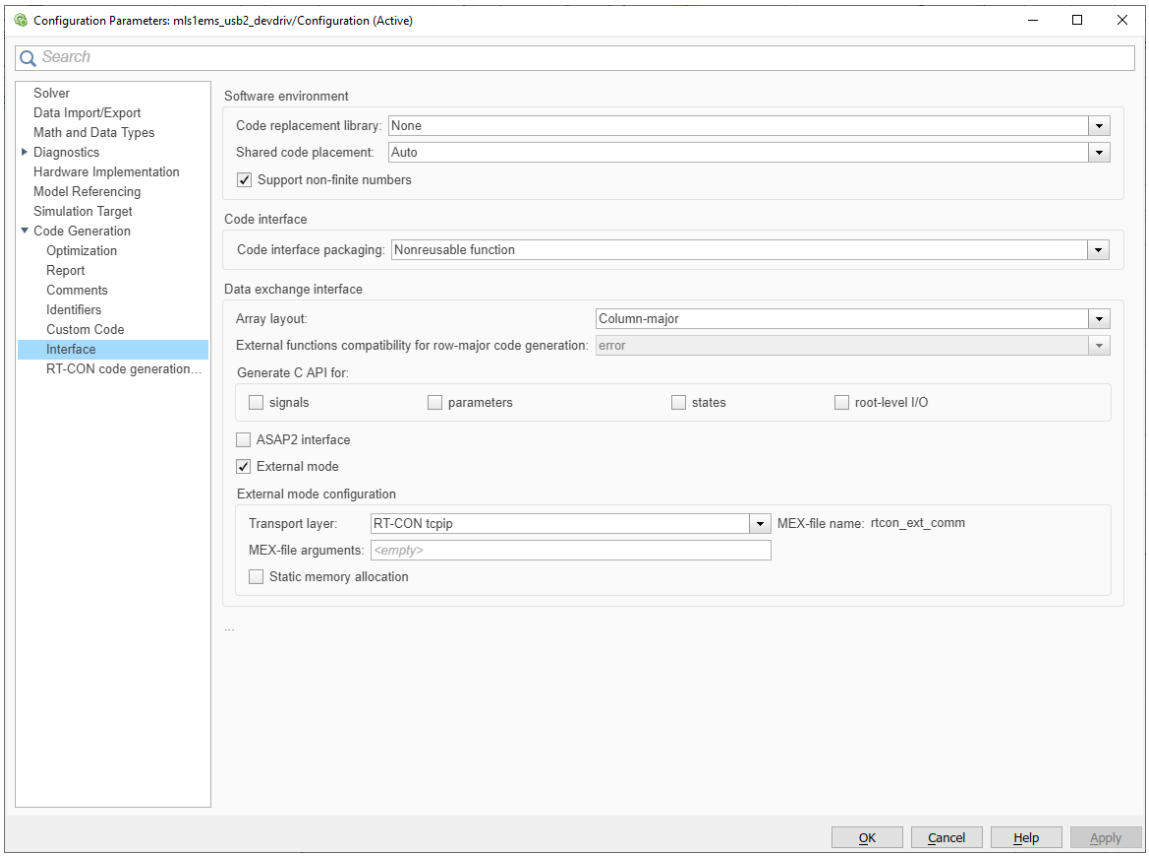


Fig. 3.3. Interface of the external mode options

3.2 Code generation and the build process

Once a model of the system has been designed the code for real-time mode can be generated, compiled, linked and downloaded into the processor.

To compile (build) model click *Monitor&Tune* button and next *Build for Monitoring* option (see in Fig. 3.4). You can also use the keyboard by clicking CTRL+b keys. You will get the same effect in this way.

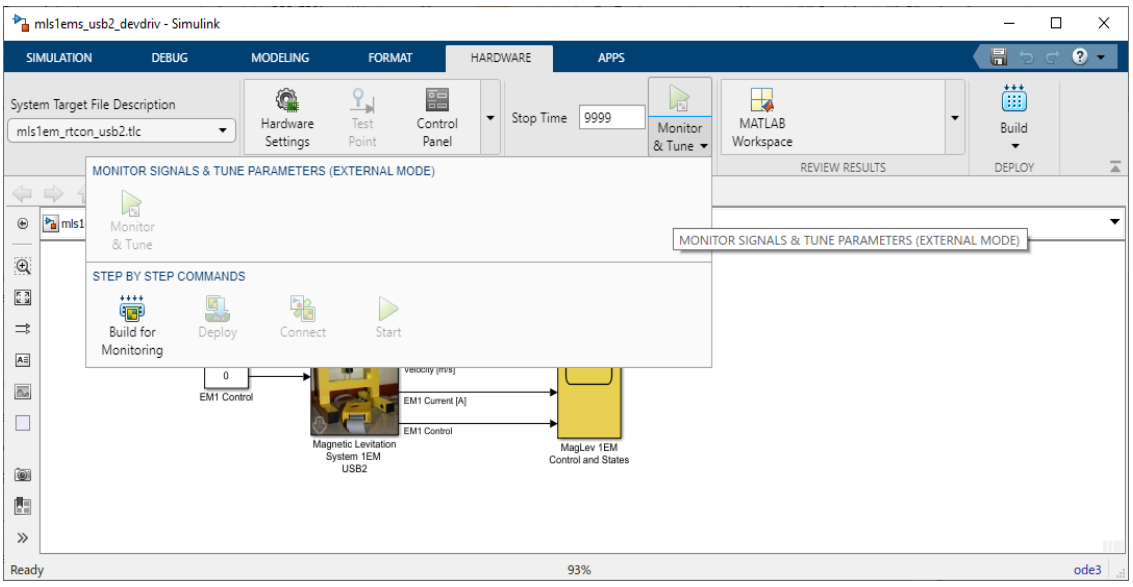


Fig. 3.4. Building model

Successful compilation and linking processes generate the following message:

Successful completion of build procedure for model: mls1ems_usb2_devdriv
Simulink cache artifacts for ' mls1ems_usb2_devdriv' were created in '..mls1ems_usb2_devdriv.slxc '. The build process completed successfully

Otherwise, an error massage is displayed in the *Diagnostic Viewer* window.

To run the real-time model click the *Control Panel* option and window shown in Fig. 3.5 will appear. Next click *Connect* button and real-time starts.

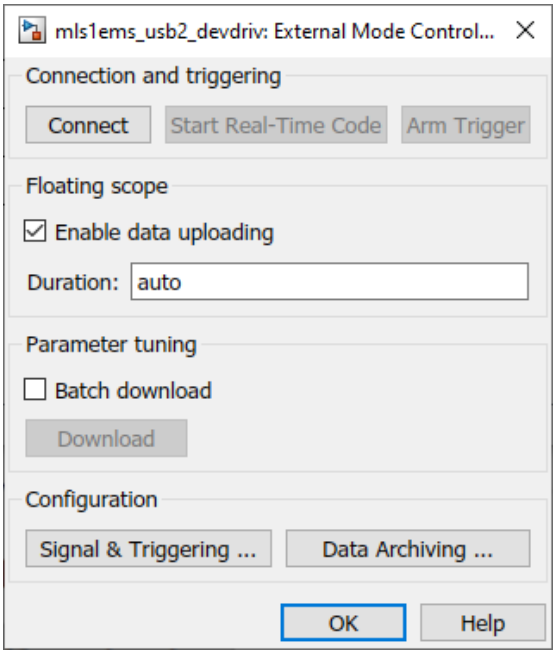


Fig. 3.5 Connect with target

Do not use option *Build Stand-Alone* shown in Fig. 3.6 **for compilation .**

.

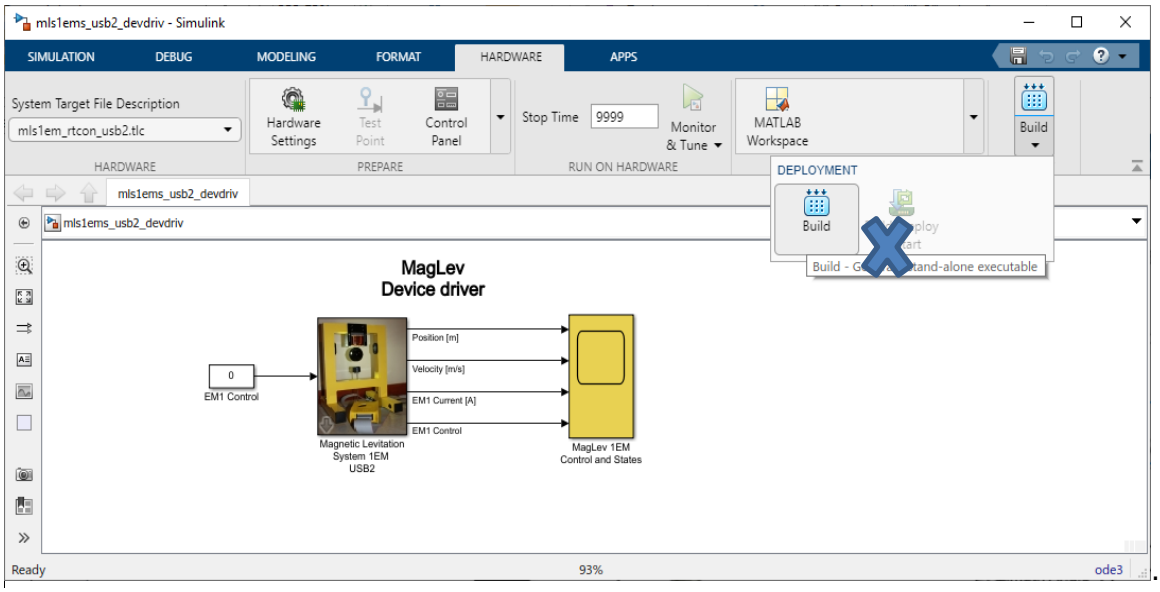


Fig. 3.6 Do not use this option !!

4 References

One can find a number of references related to the Active Magnetic Levitation Systems. Here is the list of selected references when INTECO MLS systems were used.

Dragoş, C.A., Preitl, S., Precup, R.E., Petriu, E.M., *Points of View on Magnetic Levitation System Laboratory-Based Control Education*, Springer Berlin Heidelberg, Human – Computer Systems Interaction: Backgrounds and Applications 2, 2012, 978-3-642-23171-1, http://dx.doi.org/10.1007/978-3-642-23172-8_18

Piłat A. (2005). Programmable analog hardware for control systems exampled by magnetic suspension, Computer Methods and Systems, Cracow, Poland 14-16 November

Pilat A., Turnau A., Neural adapted controller learned on-line in real-time. 14 International Conference on Methods and Models in Automation and Robotics, 19-21 August, Miedzyzdroje, Poland. 2009,
http://www.ifac-papersonline.net/Detailed/41053.html

Pilat A., Testing performance and reliability of magnetic suspension controllers, 14 International Conference on Methods and Models in Automation and Robotics, 19-21 August, Miedzyzdroje, Poland. 2009,
http://www.ifac-papersonline.net/Detailed/41071.html

Pilat A., Stiffness and damping analysis for pole placement method applied to active magnetic suspension. Automatyka, ISSN 1429-3447. 2009 vol. 13 no. 1, pp. 43-54.
http://journals.bg.agh.edu.pl/AUTOMATYKA/2009-01/Auto04.pdf

Pilat A., Investigation of discrete PID controller for active magnetic suspension,
Automatyka, ISSN 1429-3447. 2010 vol 14 no. 2, pp. 181–196.
http://journals.bg.agh.edu.pl/AUTOMATYKA/2010-02/Auto02.pdf

Panuncio Cruz Francisco, Control de un sistema de levitación magnética con compensación de redes neuronales, CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL, DEPARTAMENTO DE CONTROL AUTOMÁTICO, Octubre, 2009
http://www.ctrl.cinvestav.mx/~yuw/pdf/MaTesPCF.pdf

Pilat A., Active magnetic suspension and bearing, Modelling and simulation, eds. Giuseppe Petrone, Giuliano Cammarata. — [Vienna] : InTech Education and Publishing, 2008. ISBN 978-3-902613-25-7. pp. 453–470.
http://www.intechopen.com/books/modelling_and_simulation/active_magnetic_suspension_and_bearing

Pilat A., The programmable analog controller : static and dynamic configuration, as exemplified for active magnetic levitation. Przegląd Elektrotechniczny, Stowarzyszenie Elektryków Polskich, ISSN 0033-2097. 2012 vol. 88 no 4b, pp. 282–287

pe.org.pl/articles/2012/4b/51.pdf