

INTECO Krakow

Magnetic Levitation System

(MLS)

User's Manual

version 1.6 for MATLAB 6.5

Kraków, March 2005

Table of contents

INTRODUCTION	3
1.1 LABORATORY SET-UP	3
1.2 HARDWARE AND SOFTWARE REQUIREMENTS	4
1.3 FEATURES OF MLS	5
1.4 TYPICAL TEACHING APPLICATIONS	5
1.5 SOFTWARE INSTALLATION	5
2 ML MAIN WINDOW	6
2.1 IDENTIFICATION	7
2.1.1 <i>Sensor</i>	7
2.1.2 <i>Actuator static mode</i>	11
2.1.3 <i>Minimal control</i>	14
2.1.4 <i>Actuator dynamic mode</i>	17
2.2 MAGLEV DEVICE DRIVERS	19
2.3 SIMULATION MODEL & CONTROLLERS	22
2.3.1 <i>Open Loop</i>	22
2.3.2 <i>PID</i>	28
2.3.3 <i>LQ</i>	31
2.3.4 <i>LQ tracking</i>	34
2.4 LEVITATION	37
2.4.1 <i>PID</i>	37
2.4.2 <i>LQ</i>	39
2.4.3 <i>LQ tracking</i>	41
3 DESCRIPTION OF THE MAGNETIC LEVITATION CLASS PROPERTIES	44
3.1 BASEADDRESS	45
3.2 BITSTREAMVERSION	45
3.3 PWM	45
3.4 PWMPRESCALER	46
3.5 STOP	46
3.6 VOLTAGE	46
3.7 THERMSTATUS	47
3.8 TIME	47
3.9 QUICK REFERENCE TABLE	47

Introduction

The *Magnetic Levitation System MLS* 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. *MLS* consists of the electro-magnet, 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 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 electro-magnetic) 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 analogue controllers may be tested.

1.1 Laboratory set-up

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

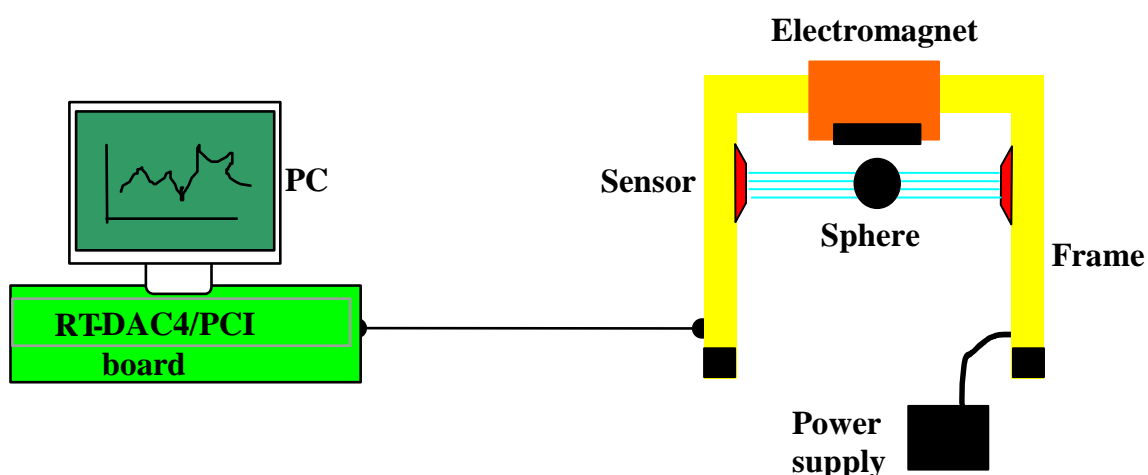


Fig. 1. MLS laboratory set-up

One obtains the mechanical unit with power supply and interface to a PC and the dedicated RTDAC4/PCI I/O board configured in the Xilinx[®] technology. The software operates in real time under MS Windows[®] 98/NT/ 2000/XP using MATLAB[®] 6.5 , RTW and RTWT 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 RTW and RTWT toolboxes. One has to know how to use the attached models and how to create his own models.

The control software for the MLS is included in the *MLS toolbox*. This toolbox uses the RTWT and RTW toolboxes from MATLAB.

MLS 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.

MLS 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.

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

1.2 Hardware and software requirements.

Hardware

Hardware installation is described in the *Assembling* manual. It consists of:

- Electromagnet
- Ferromagnetic objects
- Position sensor
- Current sensor
- Power interface
- RTDAC4/PCI measurement and control I/O board
- Pentium or AMD based personal computer.

Software

For development of the project and automatic building of the real-time program is required. The following software has to be properly installed on the PC:

- MS Windows 2000 or Windows XP. MATLAB version 6.5 with Simulink 5. Signal Processing Toolbox and Control Toolbox from MathWorks Inc. to develop the project.
- Real Time Workshop to generate the code.
- Real Time Windows Target toolbox.

- The MLS toolbox which includes specialised drivers for the MLS System, These drivers are responsible for communication between MATLAB and the RT-DAC4/PCI measuring and control board.
- MS Visual C++ to compile the generated code.

1.3 Features of MLS

- Aluminium construction
- Two ferromagnetic objects (spheres) with different weight
- Photo detector to sense the object position
- Coil current sensor
- A highly nonlinear system ideal for illustrating complex control algorithms
- None friction effects are present in the system
- The set-up is fully integrated with MATLAB[®]/Simulink[®] and operates in real-time in MS Windows[®] 98/2000/XP
- The software includes complete dynamic models.

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*. In the Matlab Command Window type:

ML_Main

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

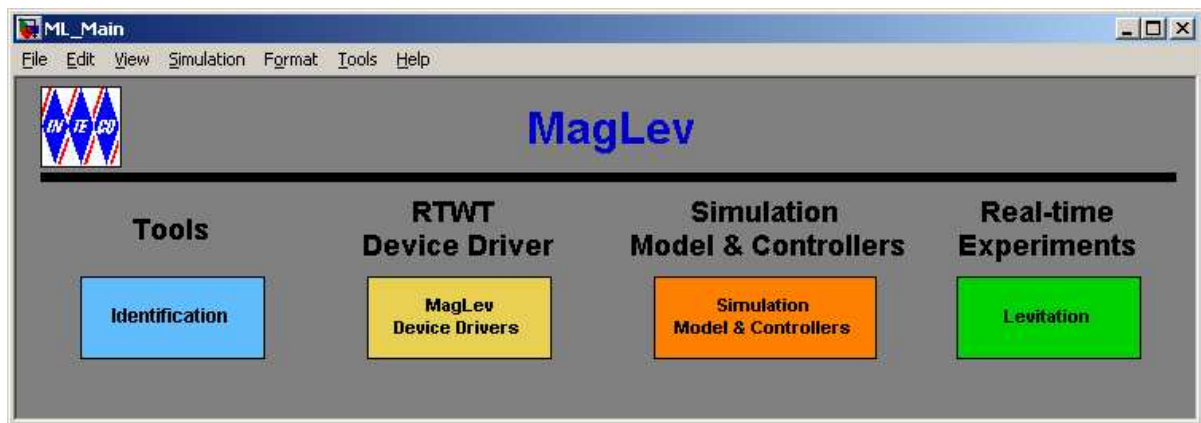


Fig. 2. The Magnetic Levitation Main window

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

The *ML Main* window shown in Fig. 2 contains four groups of the menu items:

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

Section 2 is divided into four subsections. Under each button in the ML 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 MLS.

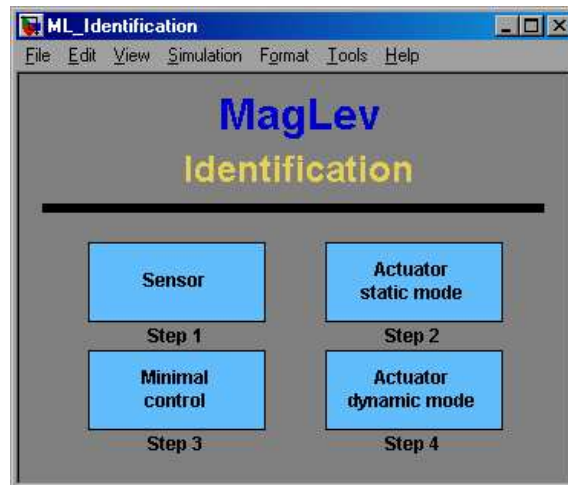


Fig. 3. The identification window

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)

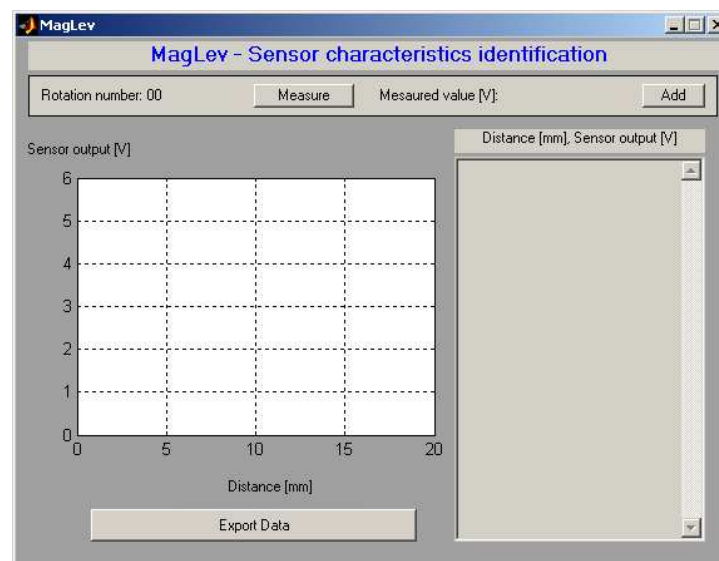


Fig. 4. Sensor signal in [V] vs. the sphere distance from the electromagnet in [mm]

The following procedure is required to identify the characteristics.

1. Screw in the screw bolt into the seat.
2. Screw in the black sphere and lock it by the butterfly nut. Notice, that **the sphere is fixed to the frame!**
3. Turn round the screw so the sphere be in touch with the bottom of the electromagnet.
4. Switch on the power supply and the light source.
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).

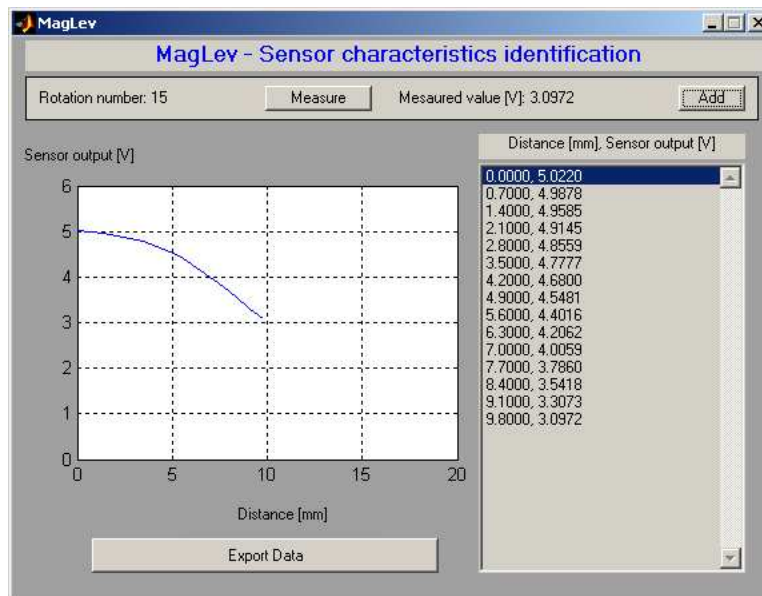


Fig. 5. Characteristics of the sphere position sensor

- 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 *ML_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 MLS (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.

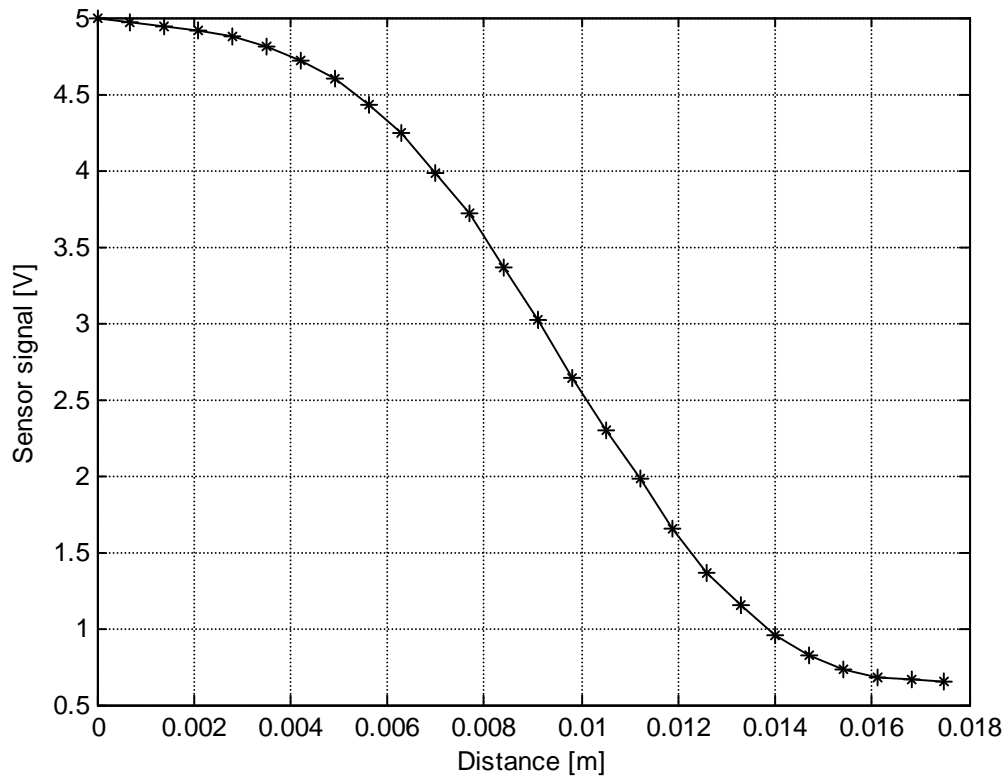


Fig. 6. The sensor characteristics after being measured and exported to the disc

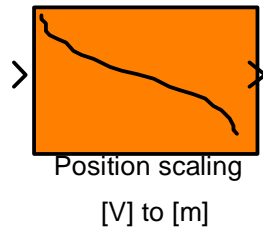


Fig. 7. The Simulink Look Up Table model representing the position sensor characteristics

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 *ML_sensor.dat* 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.

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

$$p_5 = -25697073504.59, \quad p_4 = 1245050011.25, \quad p_3 = -18773635.92, \quad p_2 = 79330.24, \\ p_1 = -150.21 \text{ and } p_0 = 5.015.$$

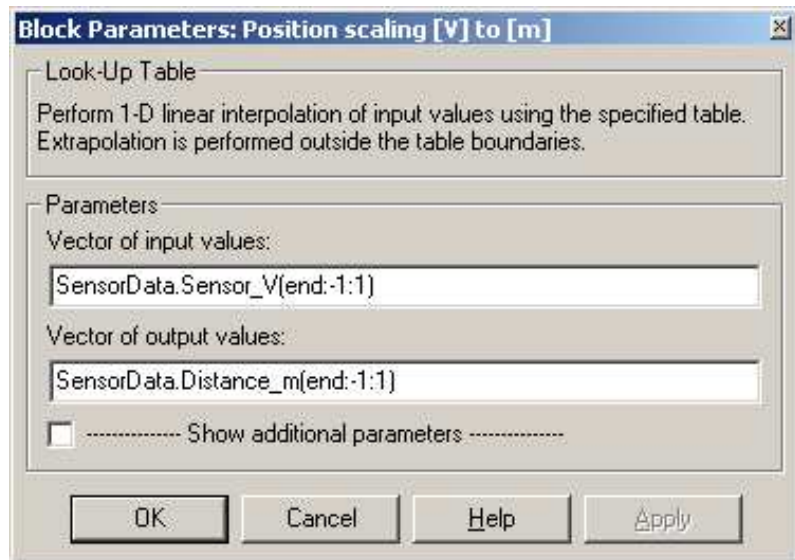


Fig. 8. Look-Up Table to be fulfilled with vectors of input and output values

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).

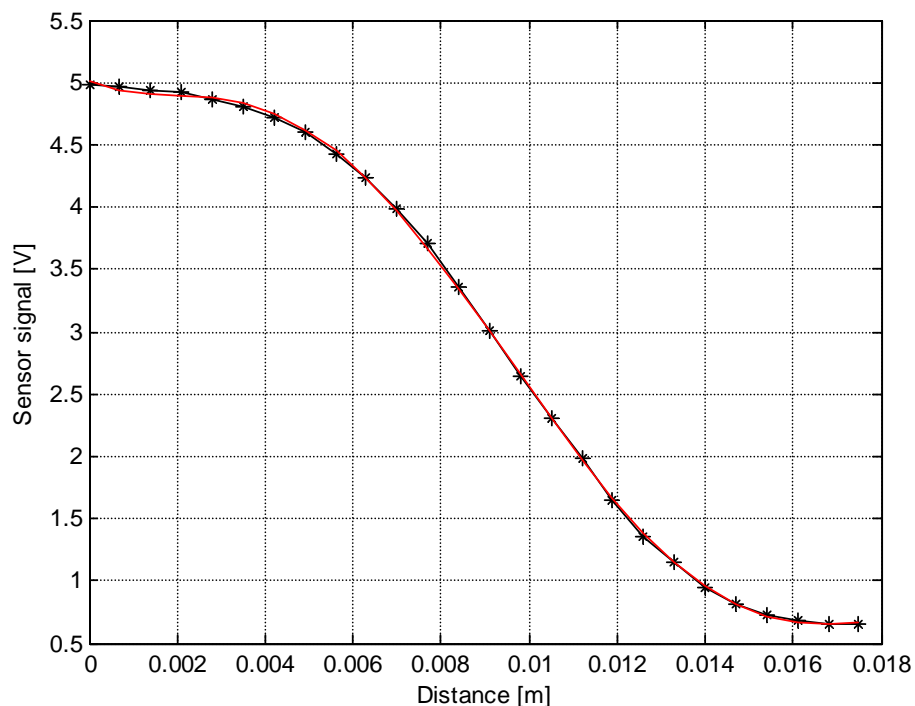


Fig. 9. The sensor characteristics approximated by the fifth order polynomial

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.

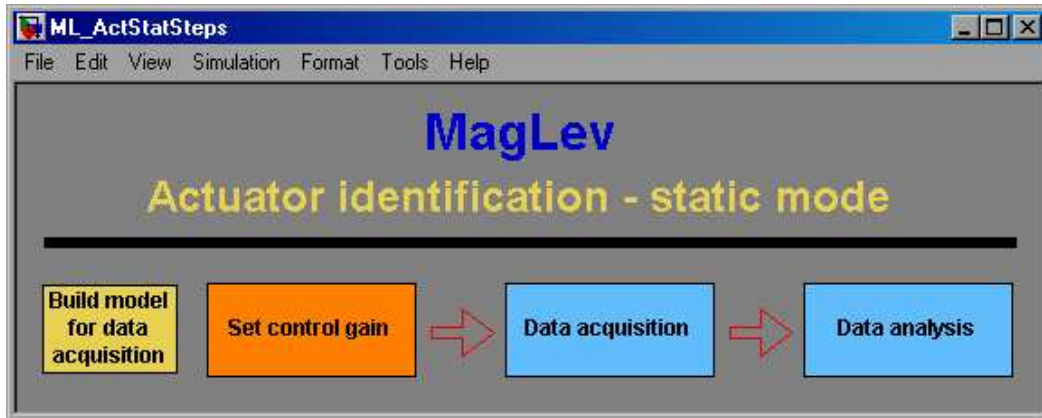


Fig. 10. Identification window of a static current/voltage characteristics

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).

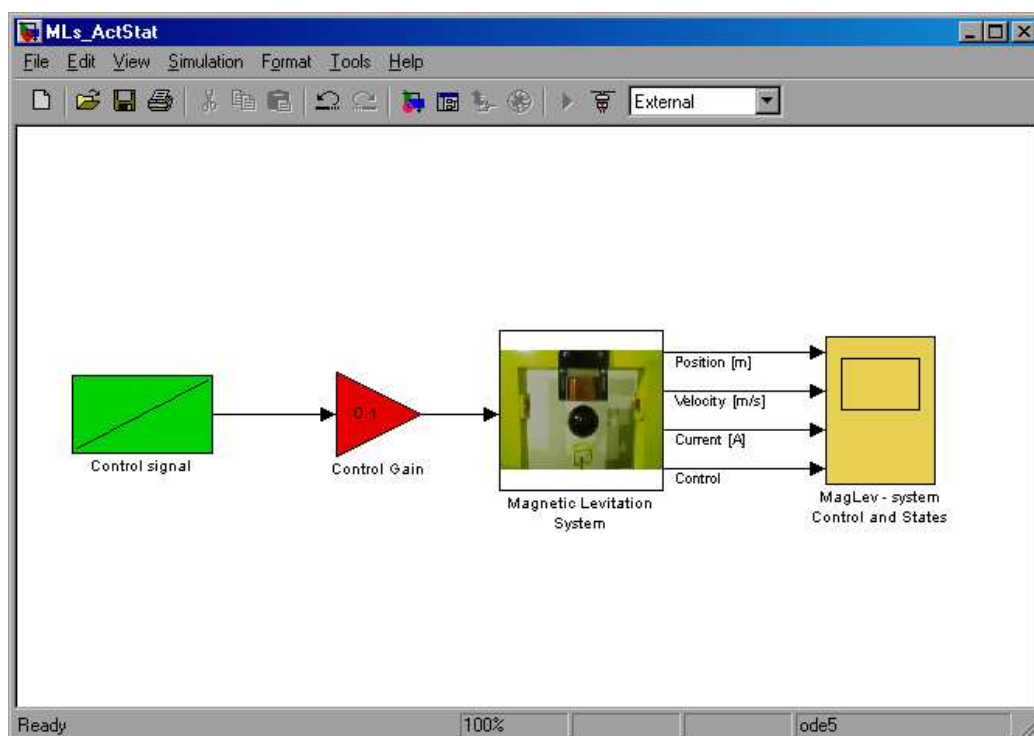


Fig. 11. Real-time model built to examine the current in the electromagnetic coil

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

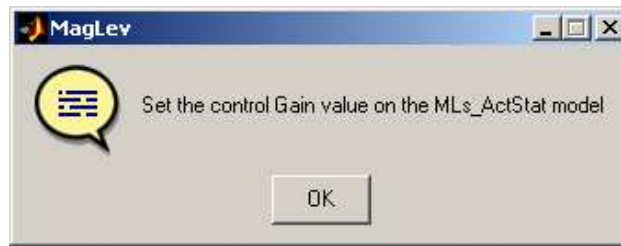


Fig. 12. Message – Set the “Control Gain”

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.

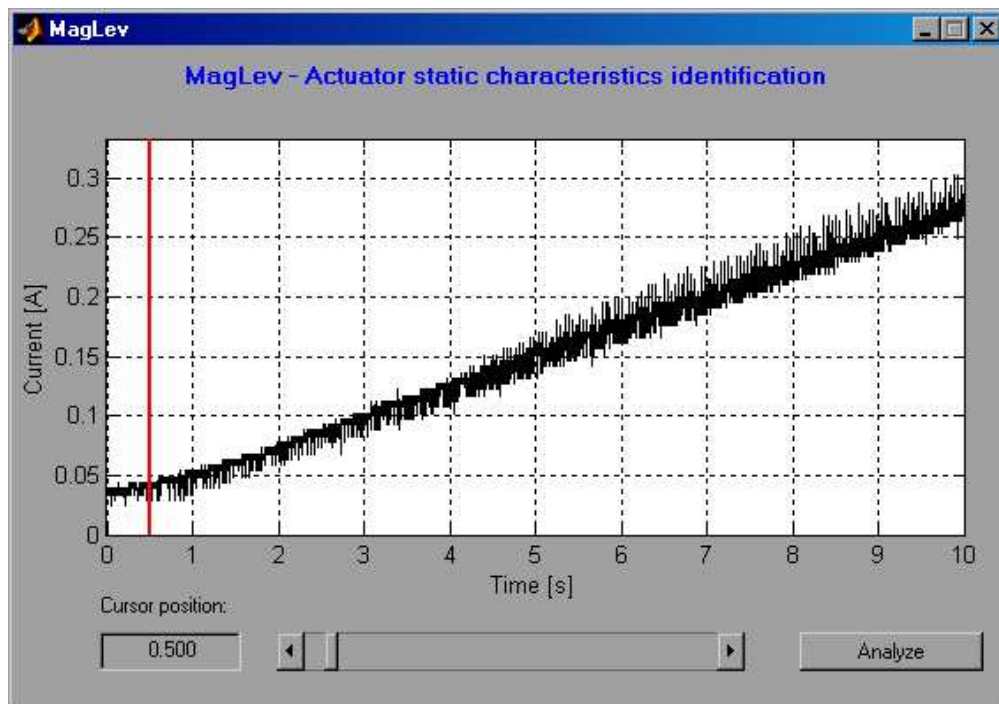


Fig. 13. Current in the electromagnetic coil

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 dragging 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: $i(u) = a u + b$.

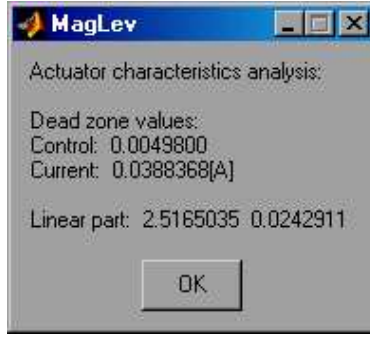


Fig. 14. Coefficients of the actuator characteristics

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).

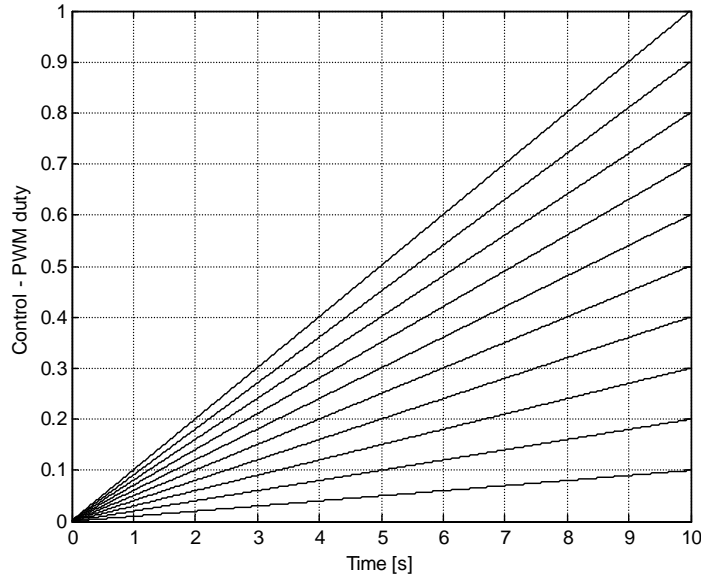


Fig. 15. Family of the input (PWM) characteristics

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:

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

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

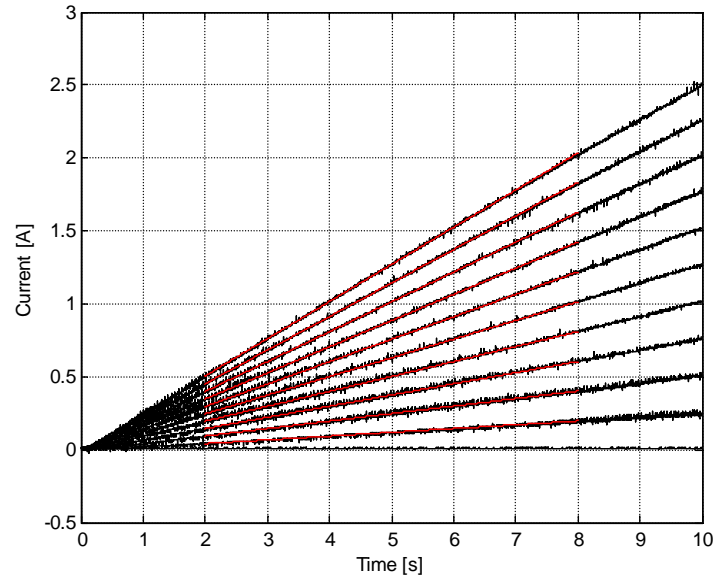


Fig. 16. Family of the output (current) characteristics

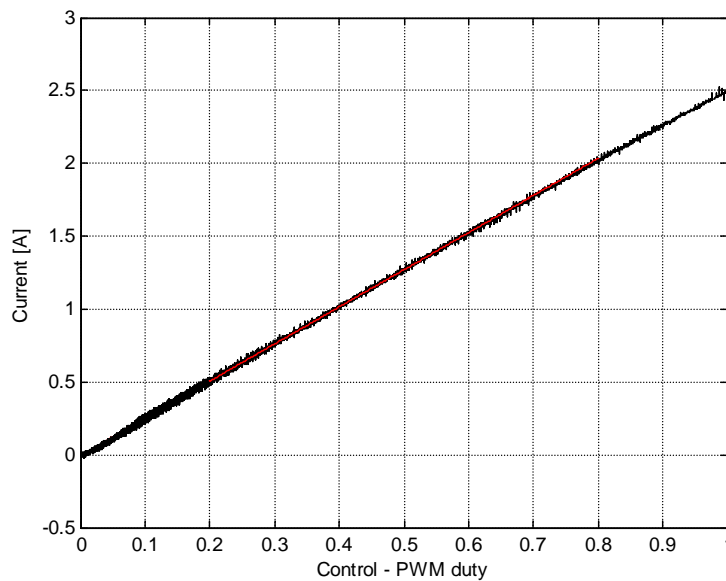


Fig. 17. Current vs. PWM duty cycle

2.1.3 Minimal control

In this subsection we examine the minimal control to cause a forced motion of the sphere from the supporting structure (tablet) toward the electromagnet against the gravity force. Notice, that in this experiment **the sphere is not levitating!** It is kept nearby the electromagnet by the supporting structure.

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

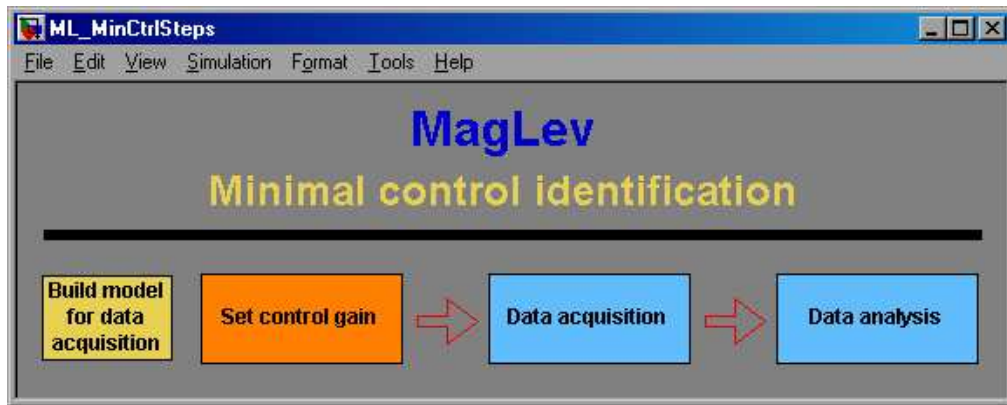


Fig. 18. Window to identify the minimal control vs. distance
(between the sphere and electromagnet)

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.

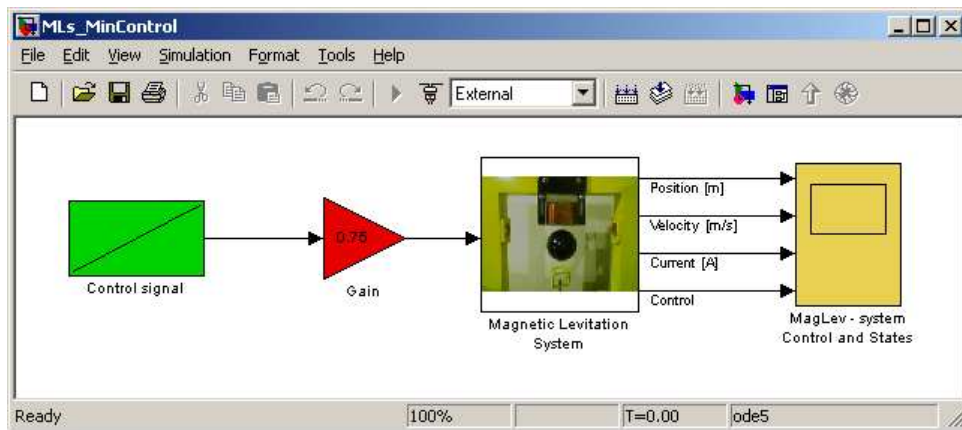


Fig. 19. Real-time model built to examine the minimal electromagnetic force

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

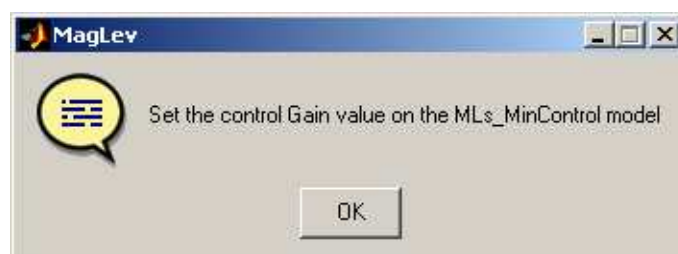


Fig. 20. Message – set the “Control Gain”

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.

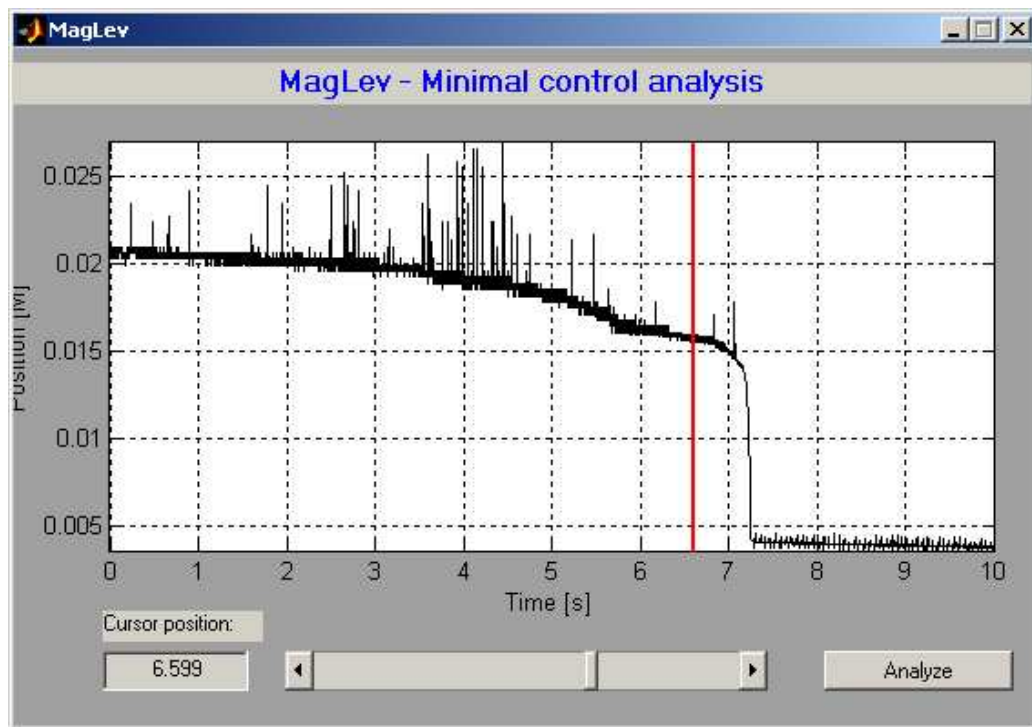


Fig. 21. The sphere motion

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 dragging 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.

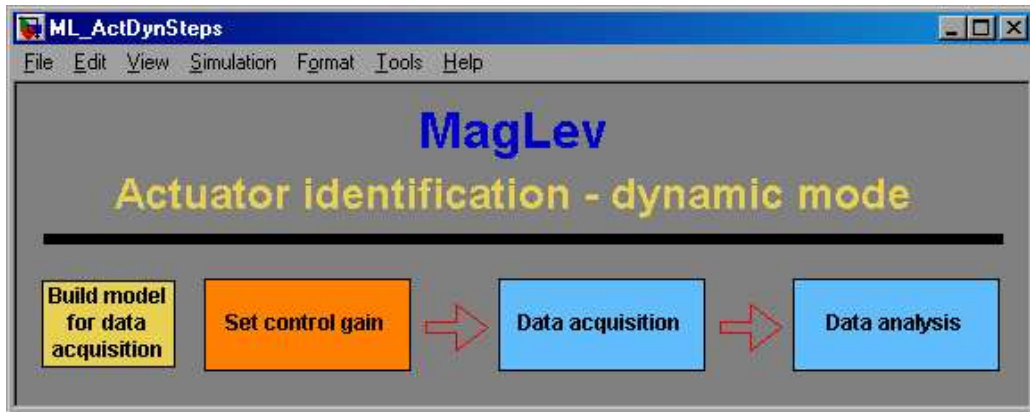


Fig. 23. Identification window of a dynamic current/voltage characteristics

A user should perform three experiments: without the sphere (*Without ball*), with the sphere on the supported structure (*Ball on the tablet*) 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.

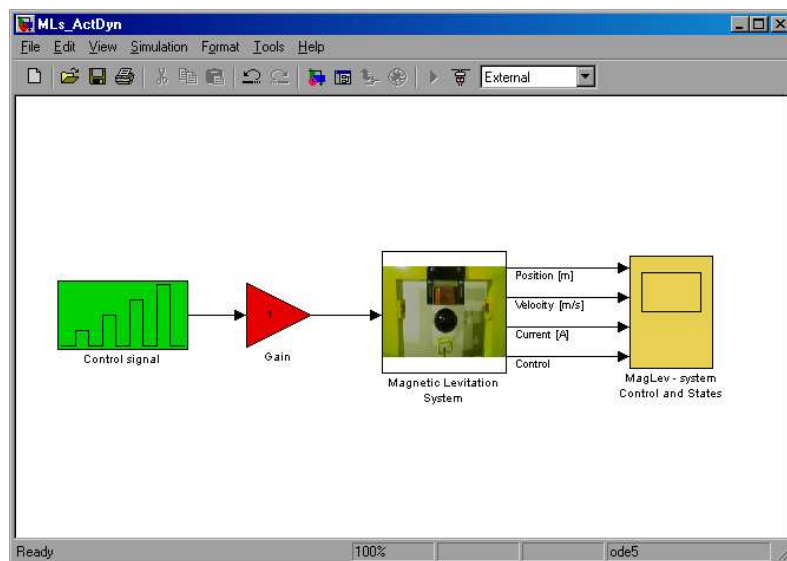


Fig. 24. Real-time model built to examine EMF influence on the coil current

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

When *ml_find_curr_dyn.m* runs the optimization function *fminsearch* is executed. *Fminsearch* uses the *ml_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.

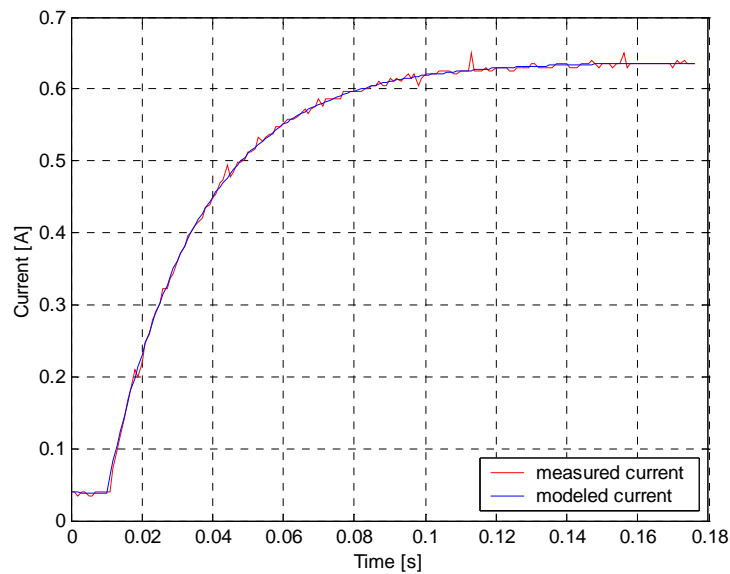


Fig. 25. Current curve – the fitting result of the optimization procedure

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.

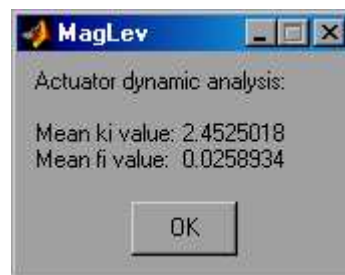


Fig. 26. Optimization results

2.2 MagLev device drivers

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

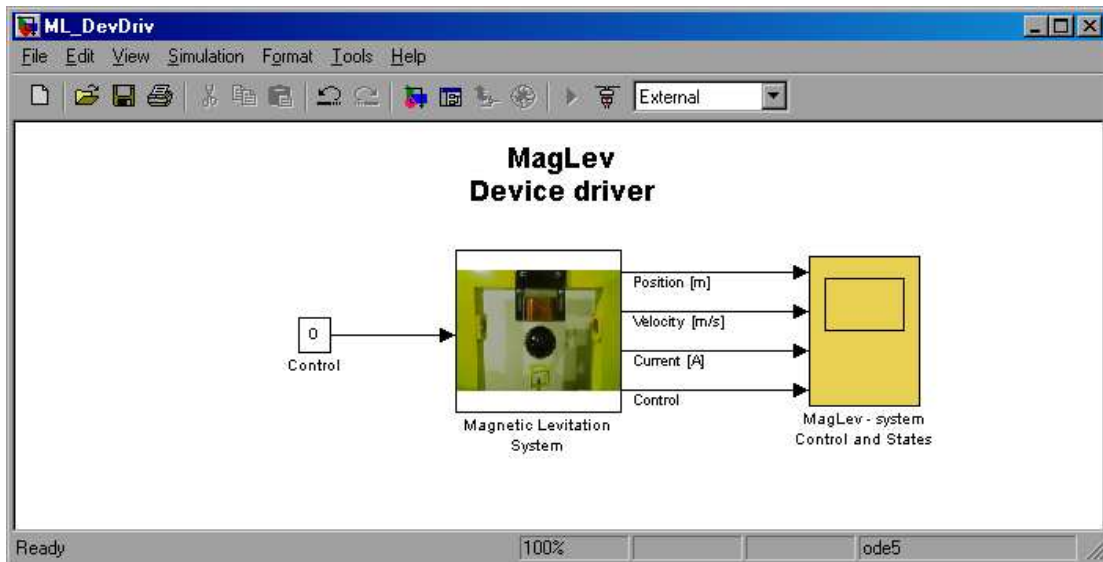


Fig. 27. RTWT MagLev device driver window

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

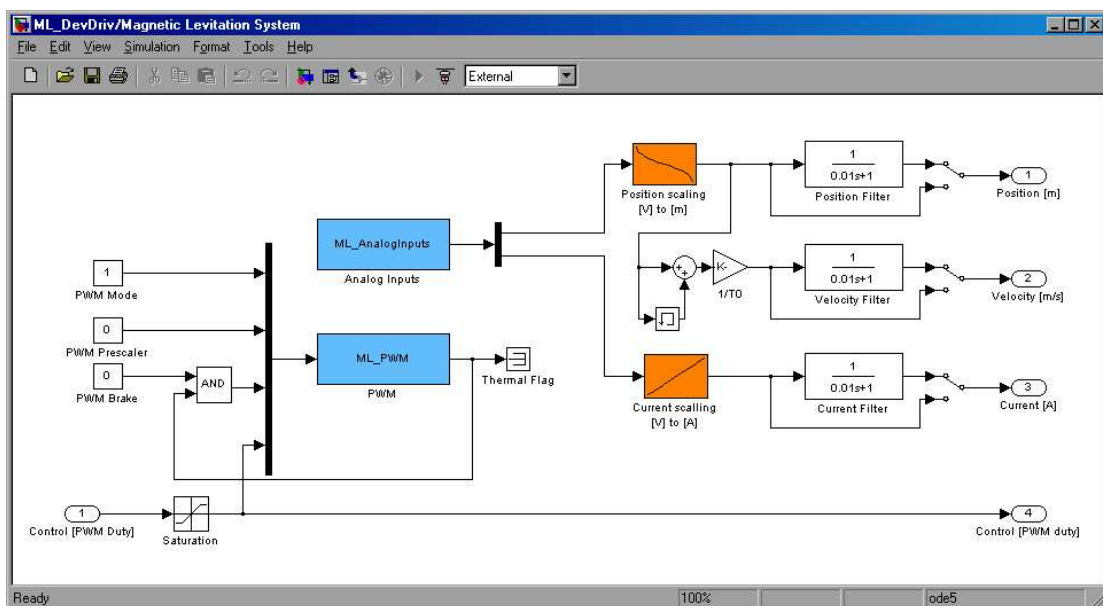


Fig. 28. Interior of the driver block

In fact there are two drivers: ML_AnalogInputs and ML_PWM. 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 communicates directly with a logic stored at the RT-DAC4/PCI board. When one wants to build his own application one can copy this driver to a new model.



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. 29).

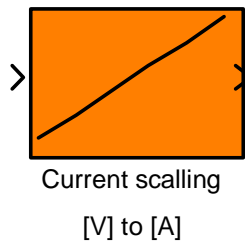


Fig. 29. The Simulink Look-Up-Table model representing the current sensor characteristics

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 LMD18200 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. 30.

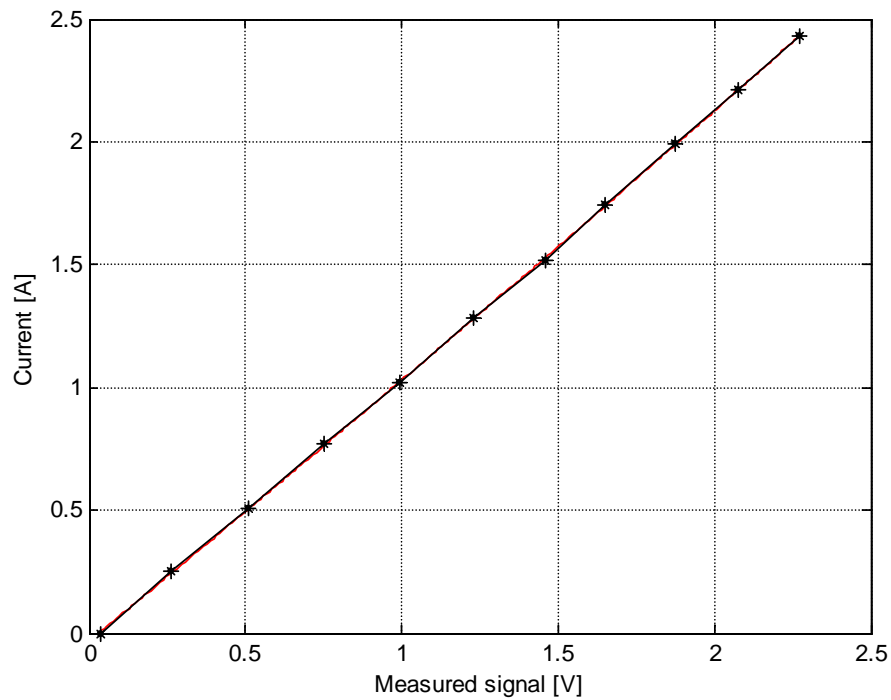


Fig. 30. Current vs. voltage characteristics approximated by the red curve

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

$$a_2 = \mathbf{0.0168}$$

$$a_1 = \mathbf{1.0451}$$

$$a_0 = \mathbf{-0.0317}$$

2.3 Simulation Model & Controllers

Click the *Simulation Model & Controllers* button in the *Magnetic Levitation Main* window. The following window opens (see Fig. 31).

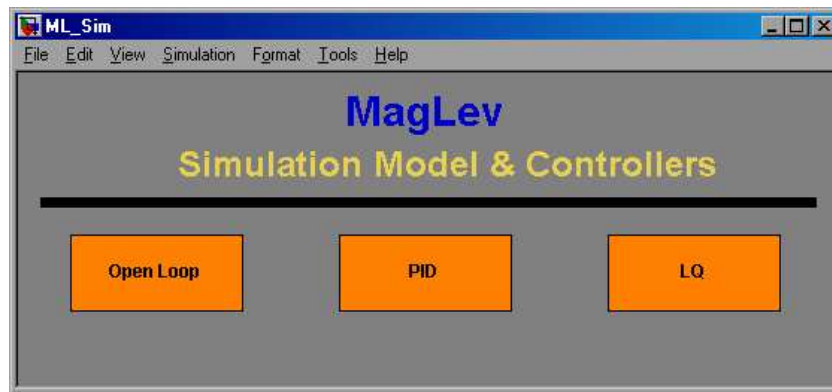


Fig. 31. Simulation Model & Controllers window

2.3.1 Open Loop

- **Simulink model**

Next, you can click the first *Open Loop* button. The following window opens (see Fig. 32). Notice that the scope block writes data to the *MLSimData* variable defined as a structure with time. The structure consists of the following signals: Position [m], Velocity [m/s], Current [A], Control [PWM duty 0÷1].

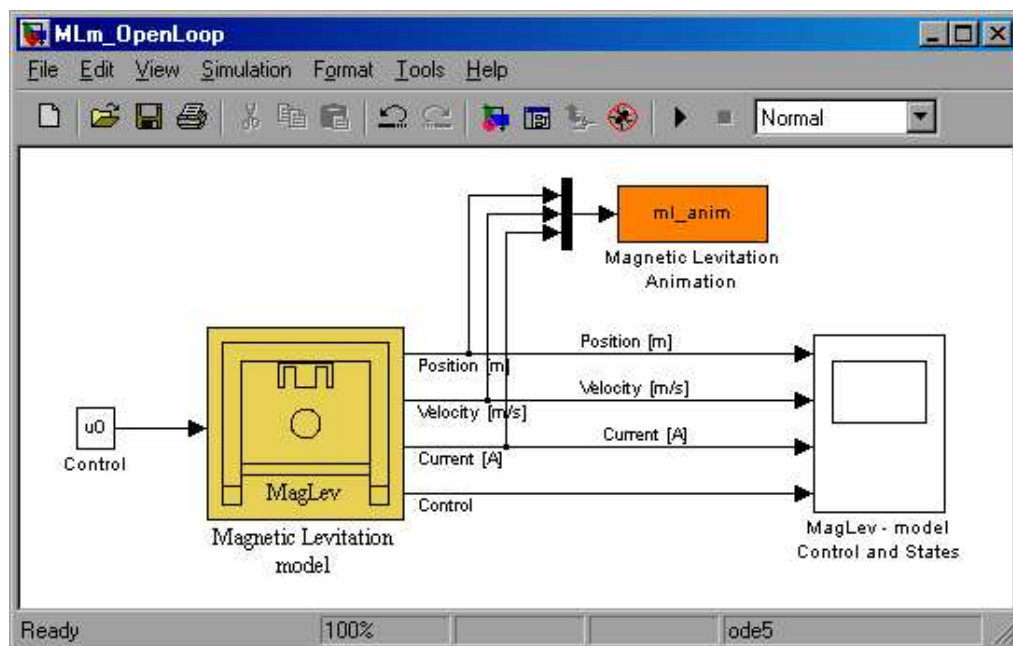
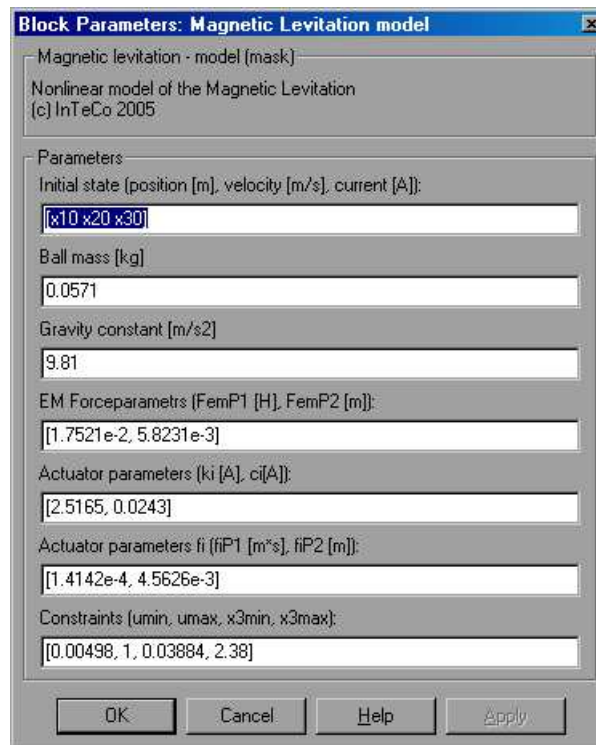


Fig. 32. Open-loop simulation

If you click the *Magnetic Levitation model* block the following mask opens (see Fig. 33).



Block Parameters: Magnetic Levitation model

Magnetic levitation - model (mask)

Nonlinear model of the Magnetic Levitation
(c) InTeCo 2005

Parameters

Initial state (position [m], velocity [m/s], current [A]):
[x10 x20 x30]

Ball mass [kg]
0.0571

Gravity constant [m/s²]
9.81

EM Forceparameters (FemP1 [H], FemP2 [m]):
[1.7521e-2, 5.8231e-3]

Actuator parameters (ki [A], ci[A]):
[2.5165, 0.0243]

Actuator parameters fi (fiP1 [m*s], fiP2 [m]):
[1.4142e-4, 4.5626e-3]

Constraints (umin, umax, x3min, x3max):
[0.00498, 1, 0.03884, 2.38]

OK Cancel Help Apply

Fig. 33. Mask of the Magnetic Levitation model

In Fig. 32 enter into the *File* option and choose *Look under mask*. The interior of the *Magnetic Levitation model* block shown in Fig. 34 opens.

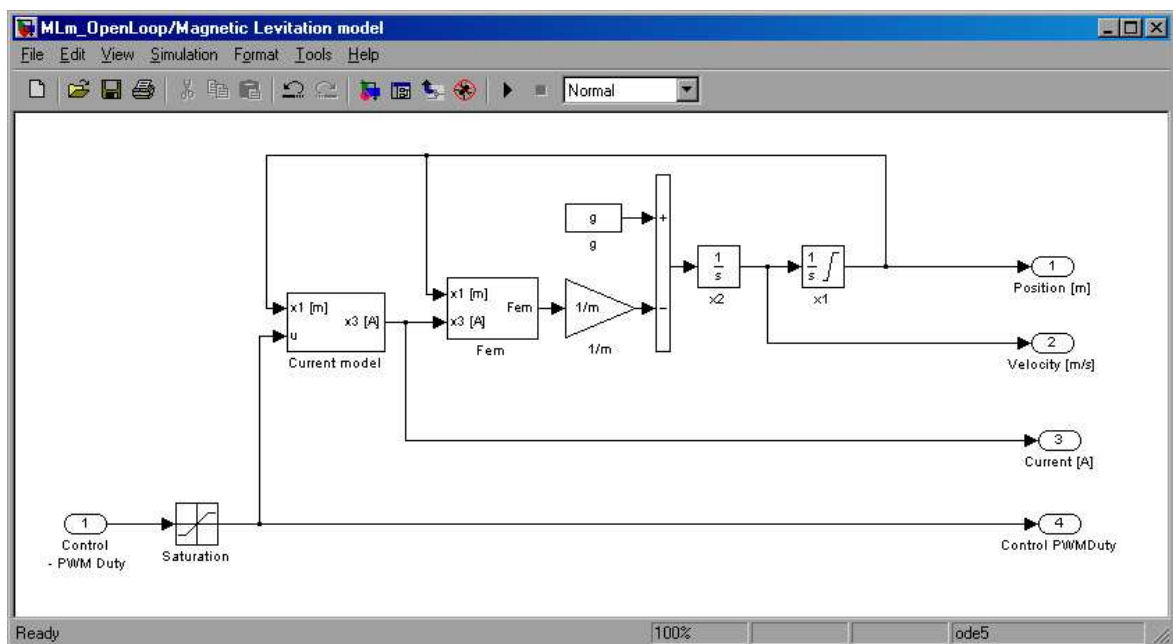


Fig. 34. Interior of the ML model

Notice two integrator blocks in Fig. 34. In fact we deal with third order dynamical system. The third integrator related to the coil current is visible in Fig. 35.

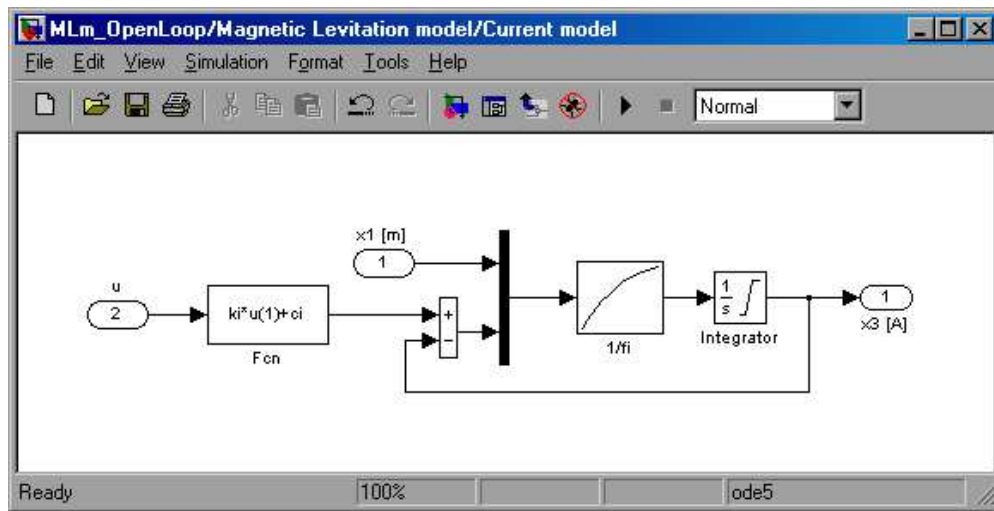


Fig. 35. Interior of the *Current model* block

The Simulink model is also equipped with the animation block. When a simulation starts the following window opens (see Fig. 36). The animation screen is updated in every sample time. All state variables: the ball position and velocity, and also the coil current are animated.

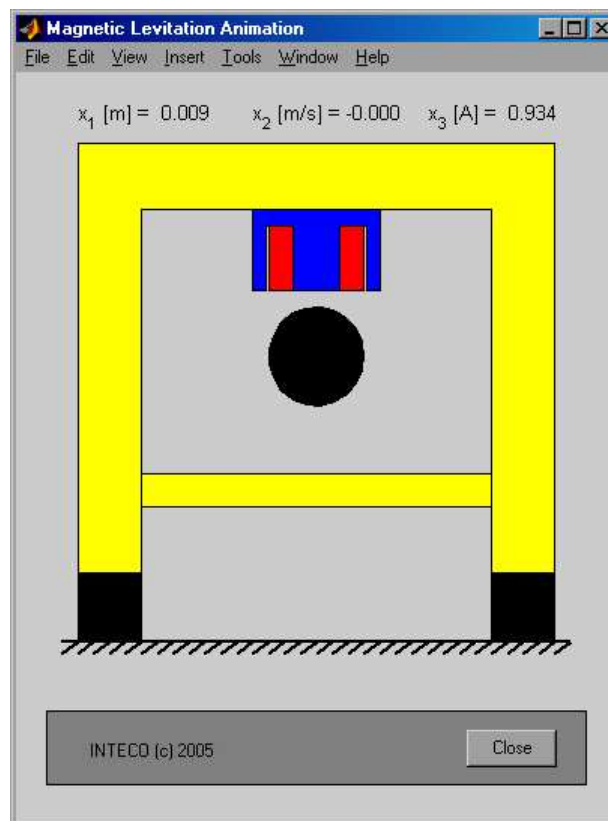


Fig. 36. ML animation

▪ Mathematical model

The Simulink model is consistent with the following nonlinear mathematical model
Model nieliniowy

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{F_{em}}{m} + g \\ \dot{x}_3 &= \frac{1}{f_i(x_1)}(k_i u + c_i - x_3) \\ F_{em} &= x_3^2 \frac{F_{emP1}}{F_{emP2}} \exp\left(-\frac{x_1}{F_{emP2}}\right) \\ f_i(x_1) &= \frac{f_{iP1}}{f_{iP2}} \exp\left(-\frac{x_1}{f_{iP2}}\right)\end{aligned}$$

where:

$$\begin{aligned}x_1 &\in [0, 0.016], \quad x_2 \in \Re, \quad x_3 \in [x_{3MIN}, 2.38] \\ u &\in [u_{MIN}, 1]\end{aligned}$$

The parameters of the above equations are given in the table below

Parameters	Values	Units
m	0.0571 (big ball)	[kg]
g	9.81	[m/s ²]
F_{em}	function of x_1 and x_3	[N]
F_{emP1}	$1.7521 \cdot 10^{-2}$	[H]
F_{emP2}	$5.8231 \cdot 10^{-3}$	[m]
$f_i(x_1)$	function of x_1	[1/s]
f_{iP1}	$1.4142 \cdot 10^{-4}$	[m·s]
f_{iP2}	$4.5626 \cdot 10^{-3}$	[m]
c_i	0.0243	[A]
k_i	2.5165	[A]
x_{3MIN}	0.03884	[A]
u_{MIN}	0.00498	

The electromagnetic force vs. position diagram is shown in Fig. 37 and the electromagnetic force vs. coil current diagram is shown respectively in Fig. 38.

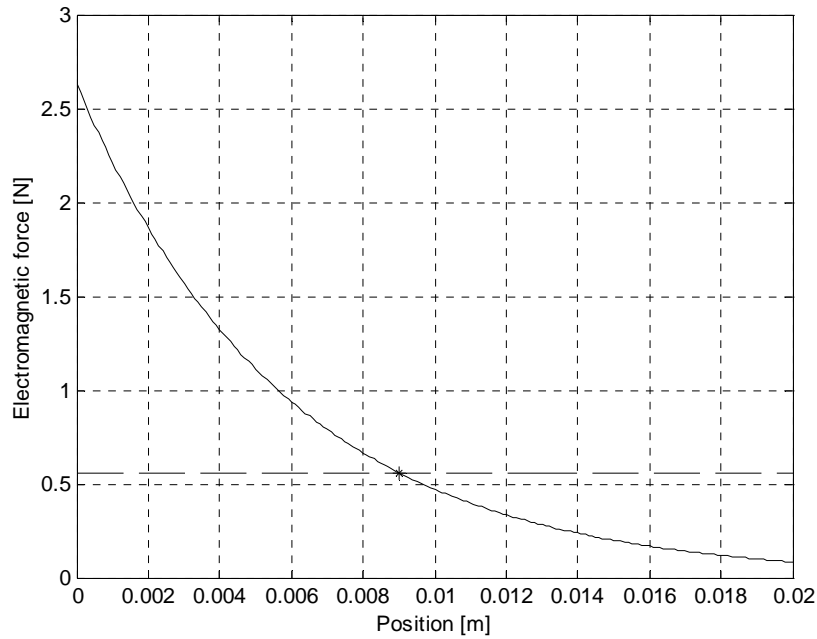


Fig. 37. Electromagnetic force vs. position. The gravity force of the big ball (dashed horizontal line) is crossing the curve at the 0.009 m distance from the electromagnet

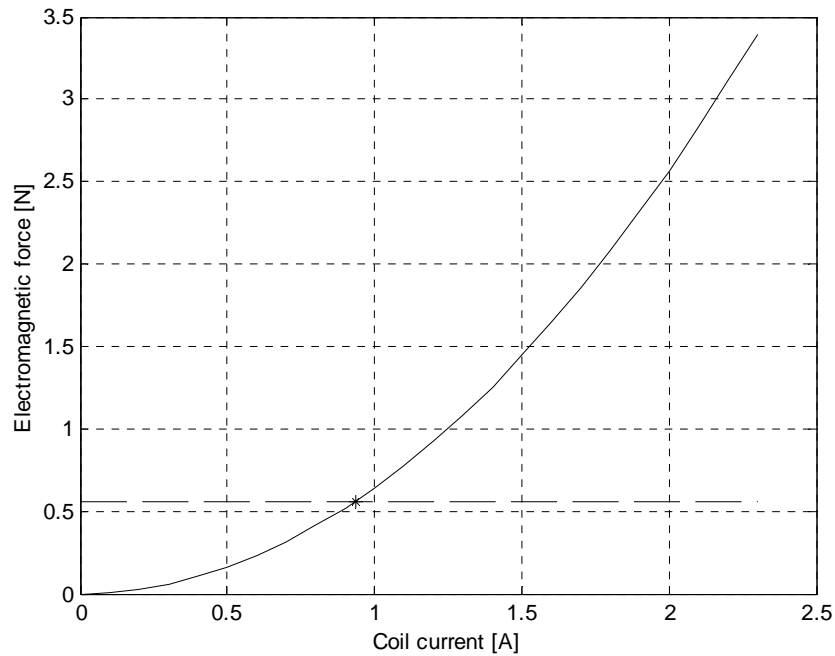


Fig. 38. Electromagnetic force vs. coil current. The gravity force of the big ball (dashed horizontal line) is crossing the curve at the 0.9345 A coil current

The electromagnetic force depends on two variables: the ball distance from the electromagnet and the current in the electromagnetic coil. This is clearly presented in Fig.

37 and Fig. 38. We can show these dependencies in three dimensional space (see Fig. 39). The ball is stabilized at $[x_1, x_2, x_3] = \text{col}(9 \cdot 10^{-3}, 0, 9.345 \cdot 10^{-1})$. It means that the ball velocity remains equal to zero. The ball is levitating kept at the 9 mm distance from the bottom of the electromagnet. The 0.9345 A current flowing through the magnetic coil is the appropriate value to balance the gravity force of the ball.

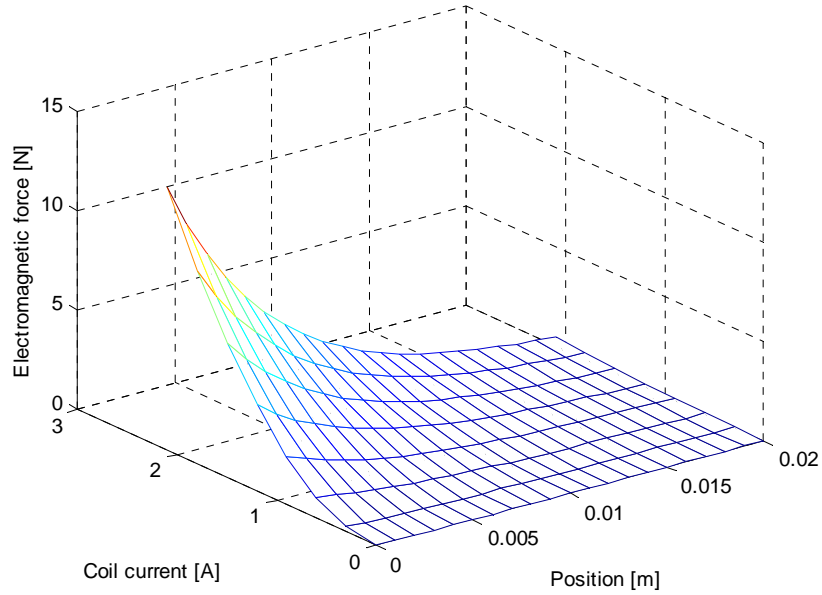


Fig. 39. Electromagnetic force vs. coil current and distance from the electromagnet.

In Fig. 40 the $f_i(x_1)$ diagram is shown.

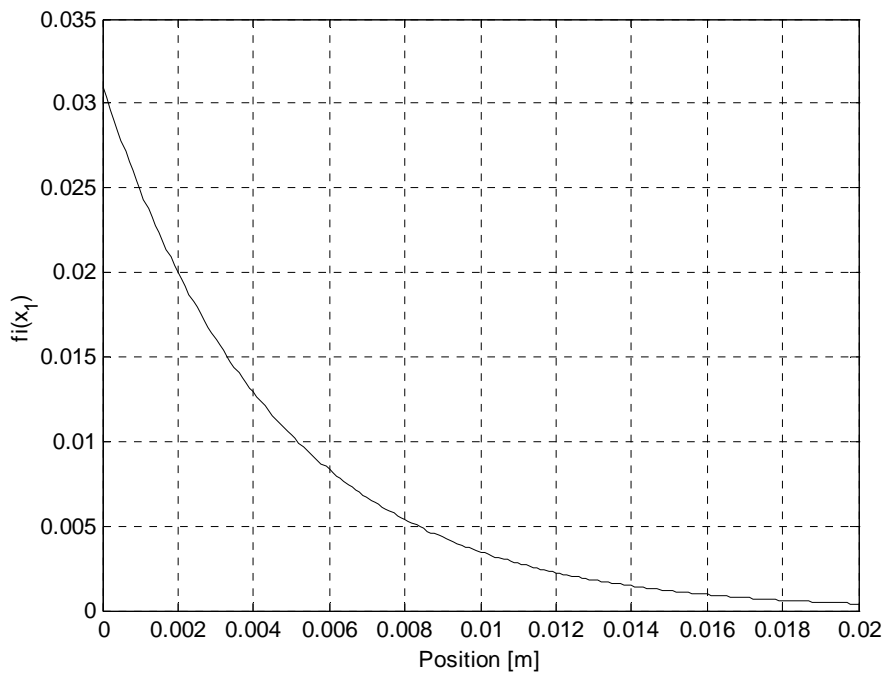


Fig. 40. Function $f_i(x_1)$

▪ Linear continuous model

ML is a highly nonlinear model. It can be approximated in an equilibrium point by a linear model. The linear model can be described by three linear differential equations of the first order in the form:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ a_{2,1} & 0 & a_{2,3} \\ a_{3,1} & 0 & a_{3,3} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ b_3 \end{bmatrix}$$

The elements of the A matrix are expressed by the nonlinear model parameters in the following way:

$$a_{2,1} = \frac{x_{30}^2}{m} \frac{F_{emP1}}{F_{emP2}^2} e^{-\frac{x_{10}}{F_{emP2}}}$$

$$a_{2,3} = -\frac{2x_{30}}{m} \frac{F_{emP1}}{F_{emP2}} e^{-\frac{x_{10}}{F_{emP2}}}$$

$$a_{3,1} = -(k_i u + c_i - x_{30}) \left(-\frac{f_{iP1}}{f_{iP2}^2} e^{-\frac{x_{10}}{f_{iP2}}} \right)^2$$

$$a_{3,3} = -f_i^{-1}(x_{10})$$

$$b_3 = k_i f_i^{-1}(x_{10})$$

The C vector elements correspond to an applied controller. For example, The PID controller shown in the next subsection requires C in the form:

$$C = [1 \quad 0 \quad 0].$$

2.3.2 PID

If you click the PID button the following windows open (see Fig. 41). The interior of the *Magnetic Levitation model* block has been shown in Fig. 34. The PID controller is built in the form:

$$u(t) = K_p \cdot e(t) + K_i \int e(t) dt + K_d \frac{d}{dt} e(t)$$

$$e(t) = x(t) - x_0(t)$$

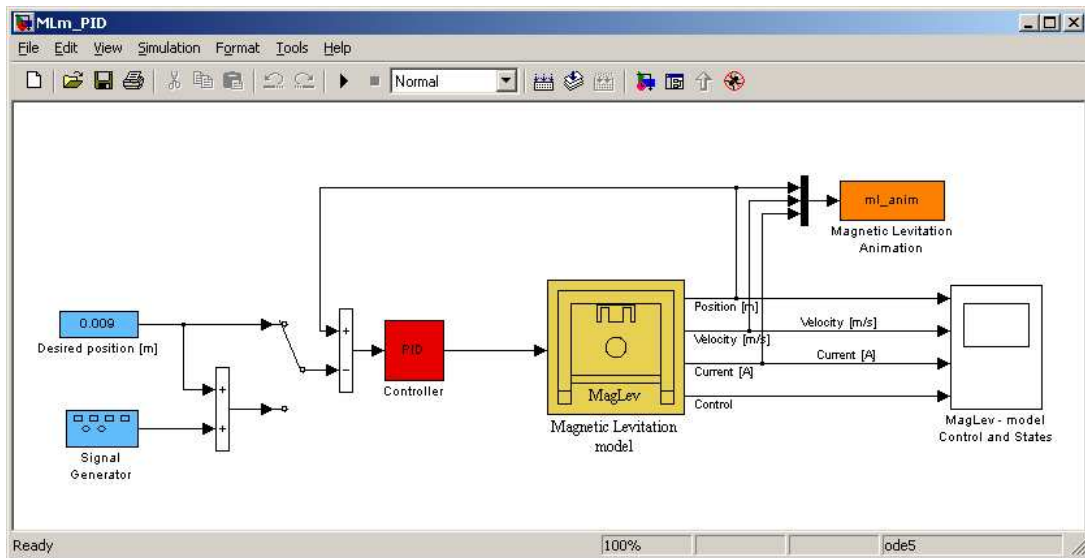


Fig. 41. PID simulation

The parameters given bellow are used in the PID controller.

K_P	K_I	K_D
130	500	6

The simulated stabilization results are shown below.

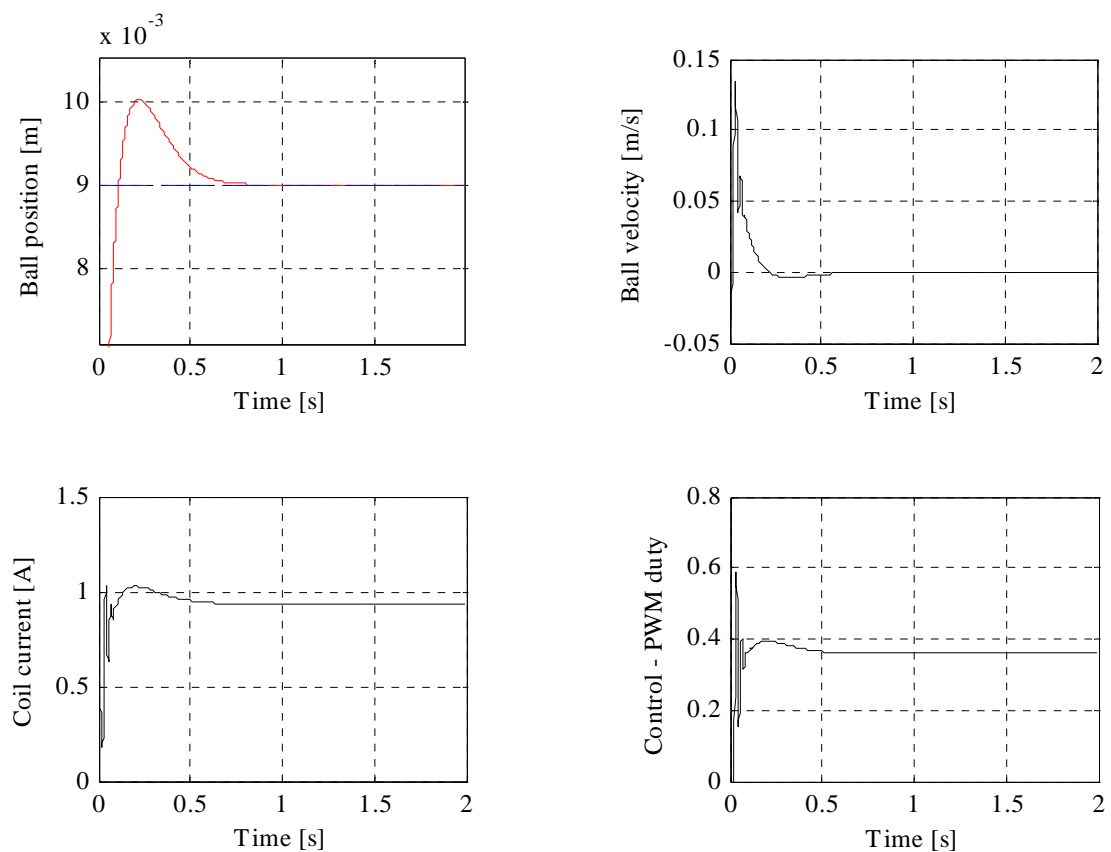


Fig. 42. PID simulation – the desired position is a constant.

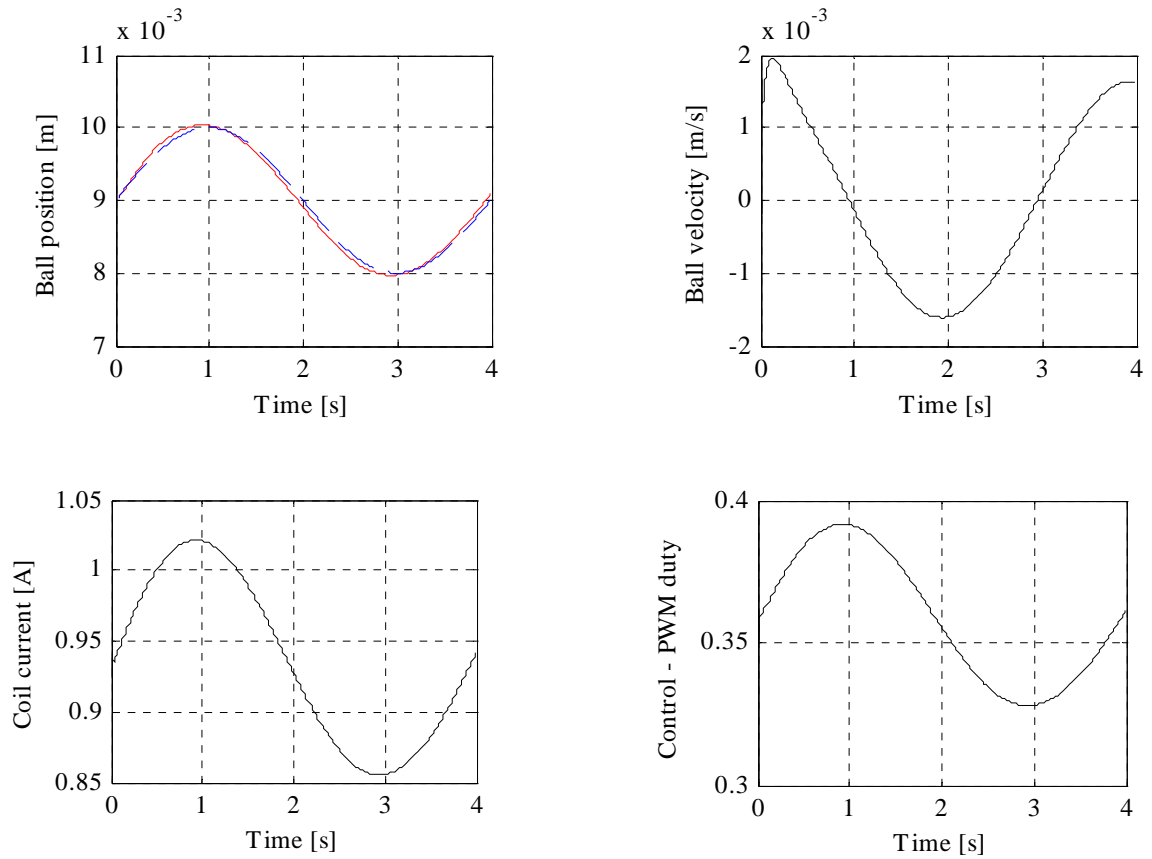


Fig. 43. PID simulation – the desired position is in a sine wave form.

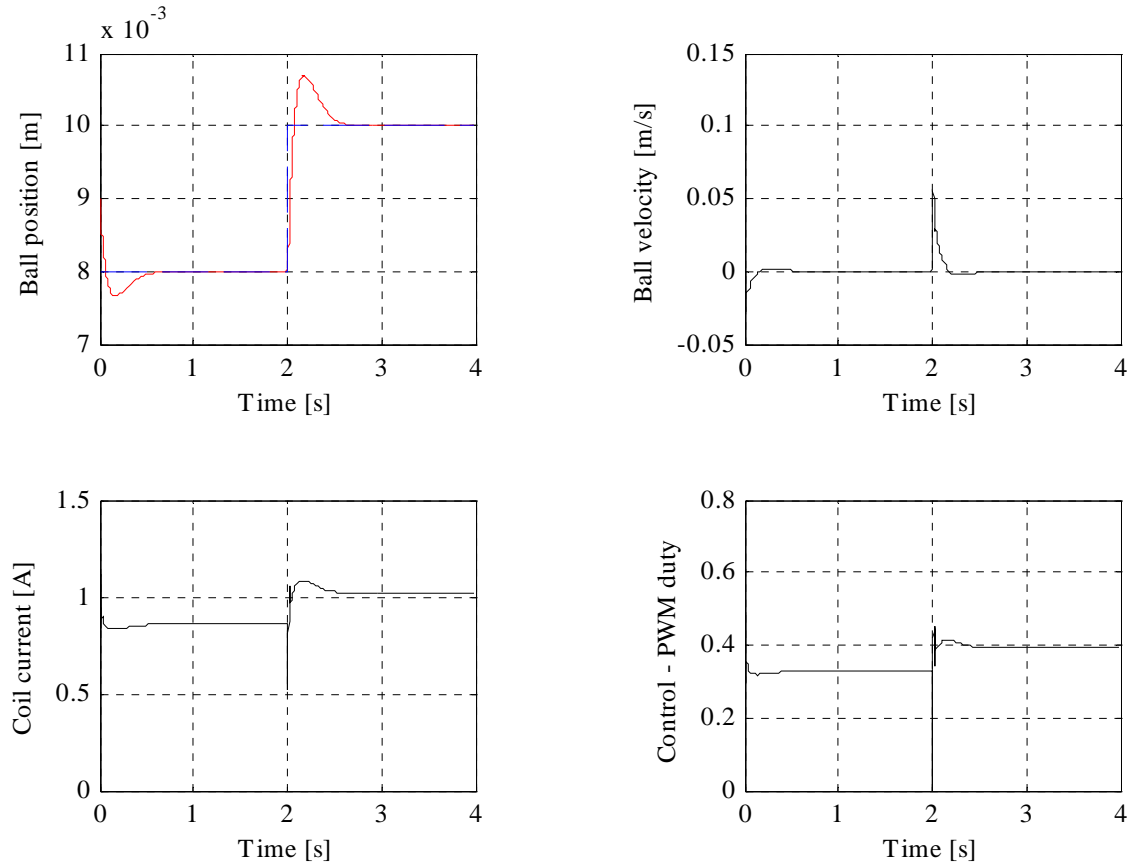


Fig. 44. PID simulation – the desired position is in a square wave form.

2.3.3 LQ

If you click the LQ button the following windows open (see Fig. 45).

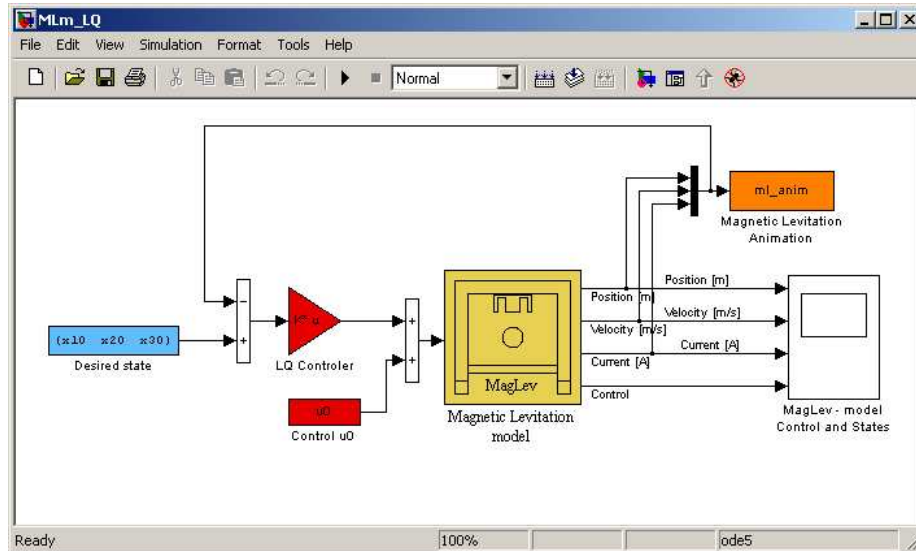


Fig. 45. LQ simulation

The continuous LQ regulator is depicted on the basis of `ml_model4lq.mdl` (see Fig. 46). The user can use two files

- `ML_calc_steady_state.m`
- `ML_calc_lq.m`

The first one calculates the equilibrium point of the system. The second one calculates the LQ controller parameters using `linmod` and `lqr`. `linmod` obtains linear models from systems of ordinary differential equations. In the `ML_calc_lq.m` file we encounter the following command:

```
[A, B, C, D] = linmod('ml_model4lq');
```

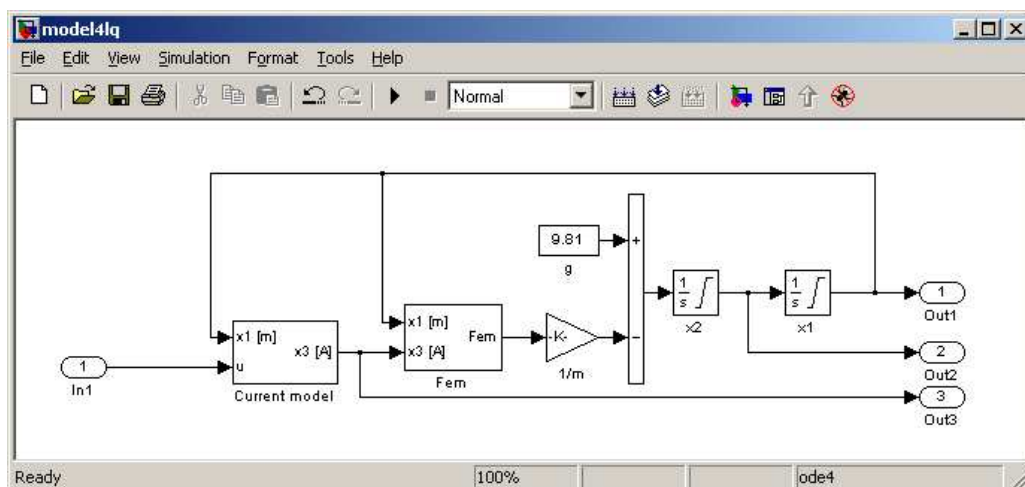


Fig. 46. `ML_model4lq.mdl` to extract an LQ regulator

The state-space linear model of the system of ordinary differential equations described in the block diagram 'model4lq' is returned in the form of the A, B, C, D matrices. The state variables and inputs are set to the defaults specified in the block diagram. Having obtained the linear model calculated at x_{10} , x_{20} , x_{30} equilibrium point for the assumed value u_0 of the control we are ready to calculate the K gains of the LQ controller. We only need to assume the Q and R matrices. From the *ML_calc_lq.m* file we have

$$Q = \text{eye}(3,3); \quad Q(1,1) = 300; \quad Q(2,2) = 0.001; \quad Q(3,3) = 10; \quad R = 10.5;$$

The following assumptions corresponding to the Q and R weighting matrices have to be satisfied:

$$Q \geq 0 \quad R > 0$$

The following command from the *ML_calc_lq.m* file:

$$[K, S, E] = \text{lqr}(A, B, Q, R)$$

calculates the optimal gain matrix K such that the state-feedback law $u = -Kx$ minimizes the cost function $\int (x'Qx + u'Ru)dt$ subject to the state dynamics $\frac{dx}{dt} = Ax + Bu$.

Now, the gain vector K can be used as the optimal feedback (see the Simulink diagram in Fig. 45). We start the LQ simulation for a constant desired value and for the desired position assumed in a sine wave form. We obtain the results shown in Fig. 47 and Fig. 48.

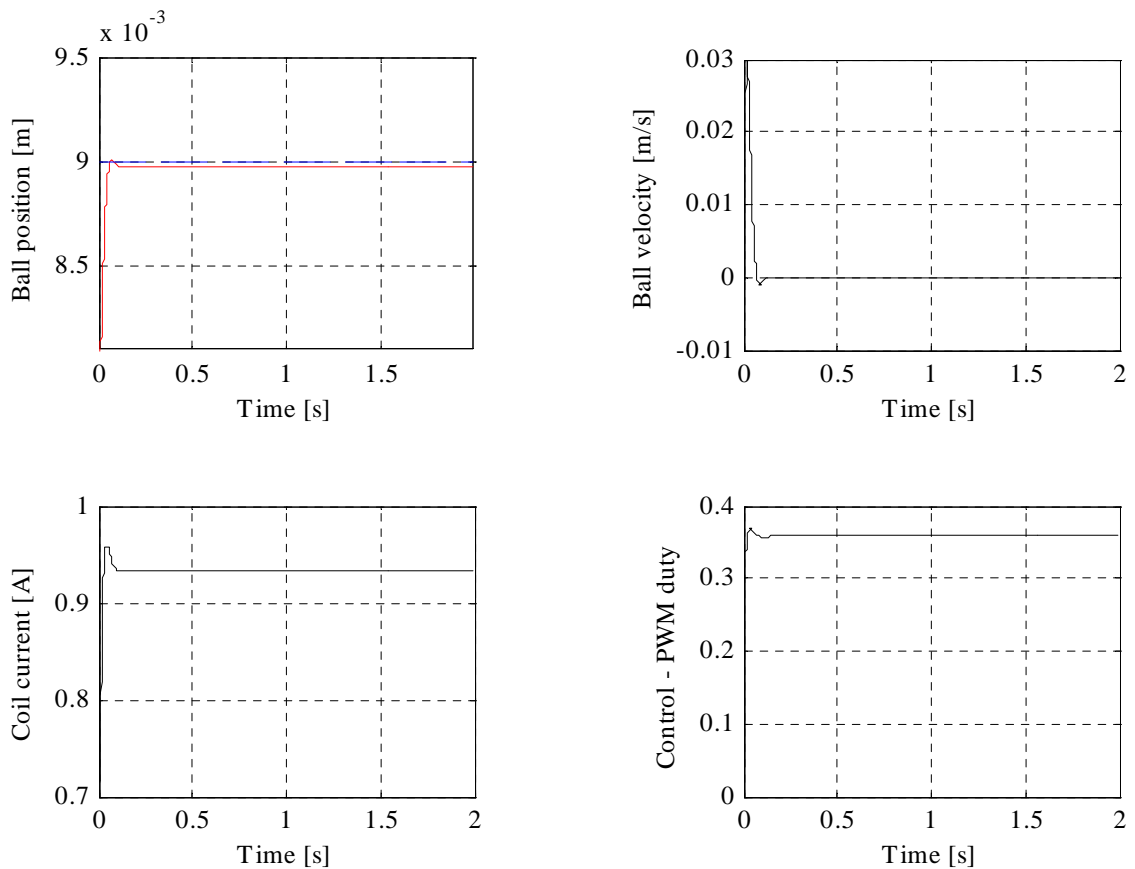


Fig. 47. LQ simulation – the desired position is a constant

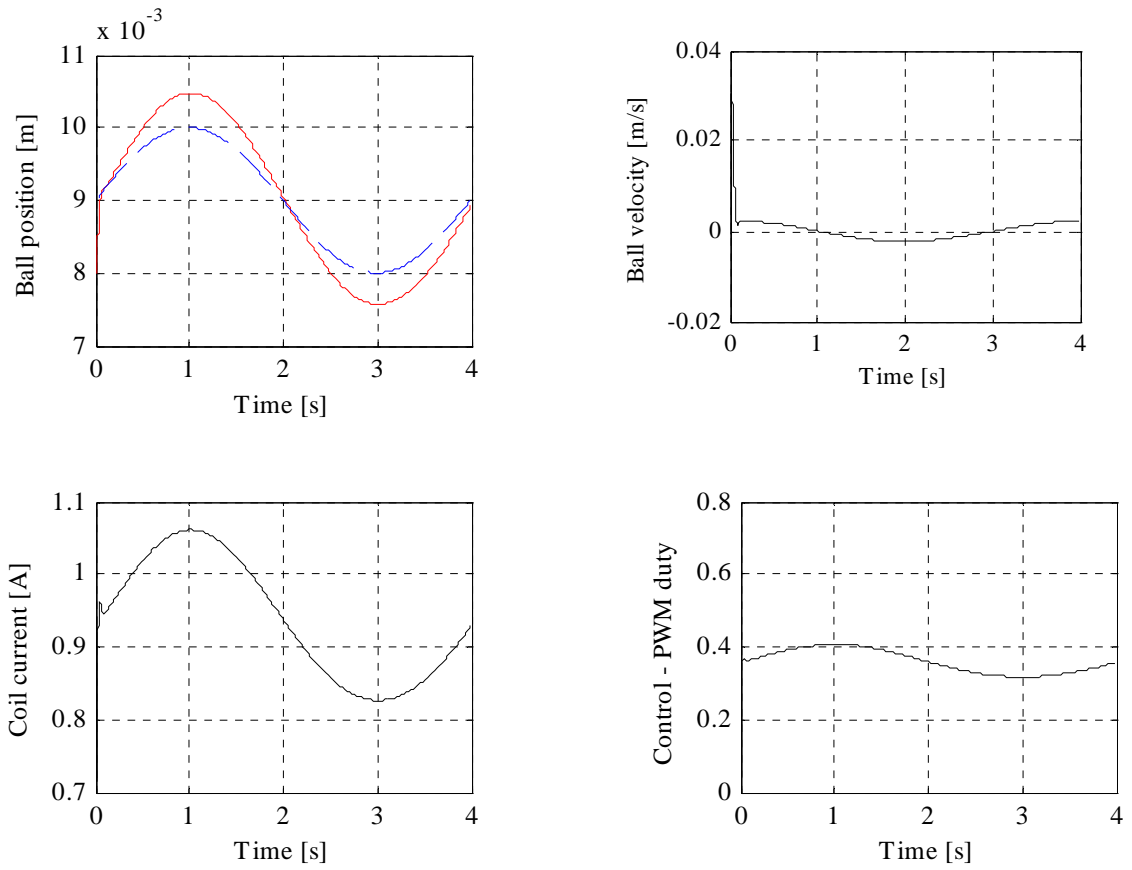


Fig. 48. LQ simulation – the desired position is in a sine wave form

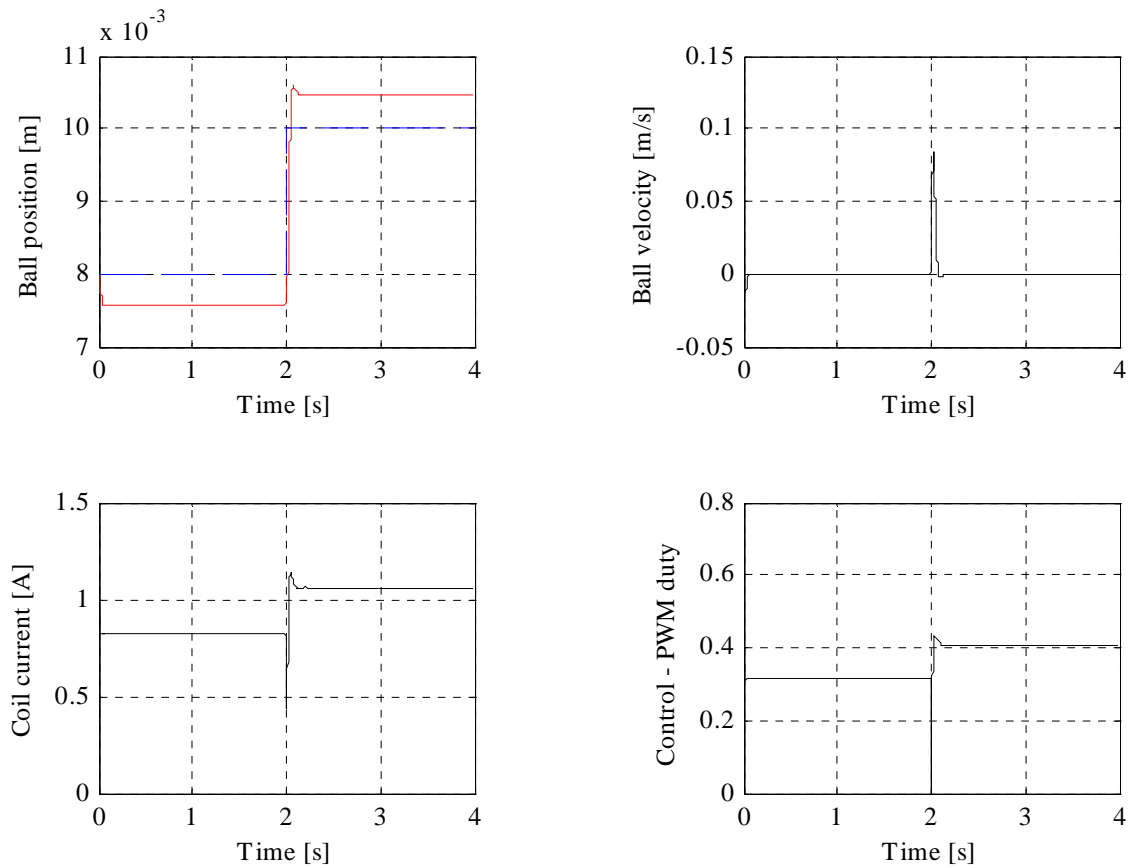


Fig. 49. LQ simulation – the desired position is in a square wave form

Similarly, we perform the LQ simulation for the desired position assumed in a square wave form. The simulation results are consequently shown in Fig. 49.



Remember that the obtained results are correct as long as the control and state variables do not saturate. Otherwise, the control algorithm has nothing to do with the LQ policy.

2.3.4 LQ tracking

If you click the LQ tracking button the following windows open (see Fig. 50). We do remember that the LQ control policy has been calculated for a given equilibrium point. To improve the LQ control action we introduce the LQ tracking policy. For each new value of the ball position the ball velocity, coil current values and the control are recalculated on the basis of nonlinear dynamical equations of ML (the *ML_GetStState* s-function is used). In fact we should introduce a no-stationary LQ – it means solve the Riccati equation for every new equilibrium point to obtain a new value of the gain vector K .

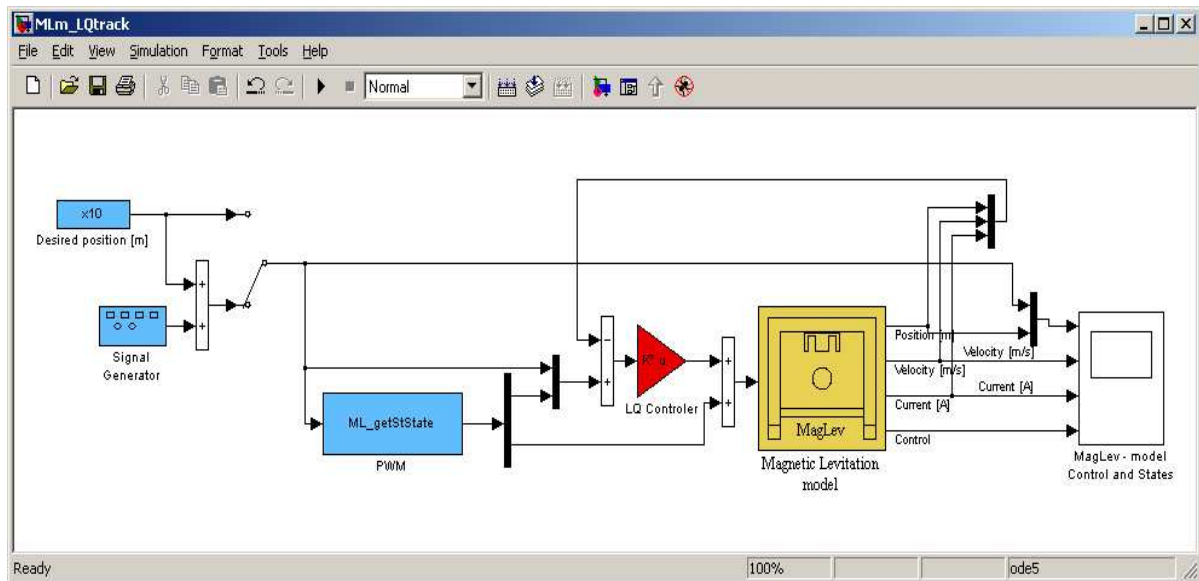


Fig. 50. LQ tracking simulation

Now, the gain vector K can be used as the optimal feedback (see the Simulink diagram in Fig. 50). We start the LQ tracking simulation for a constant desired value and for the desired position assumed in sine and square wave forms. We obtain the results shown in Fig. 51, Fig. 52 and Fig. 53.

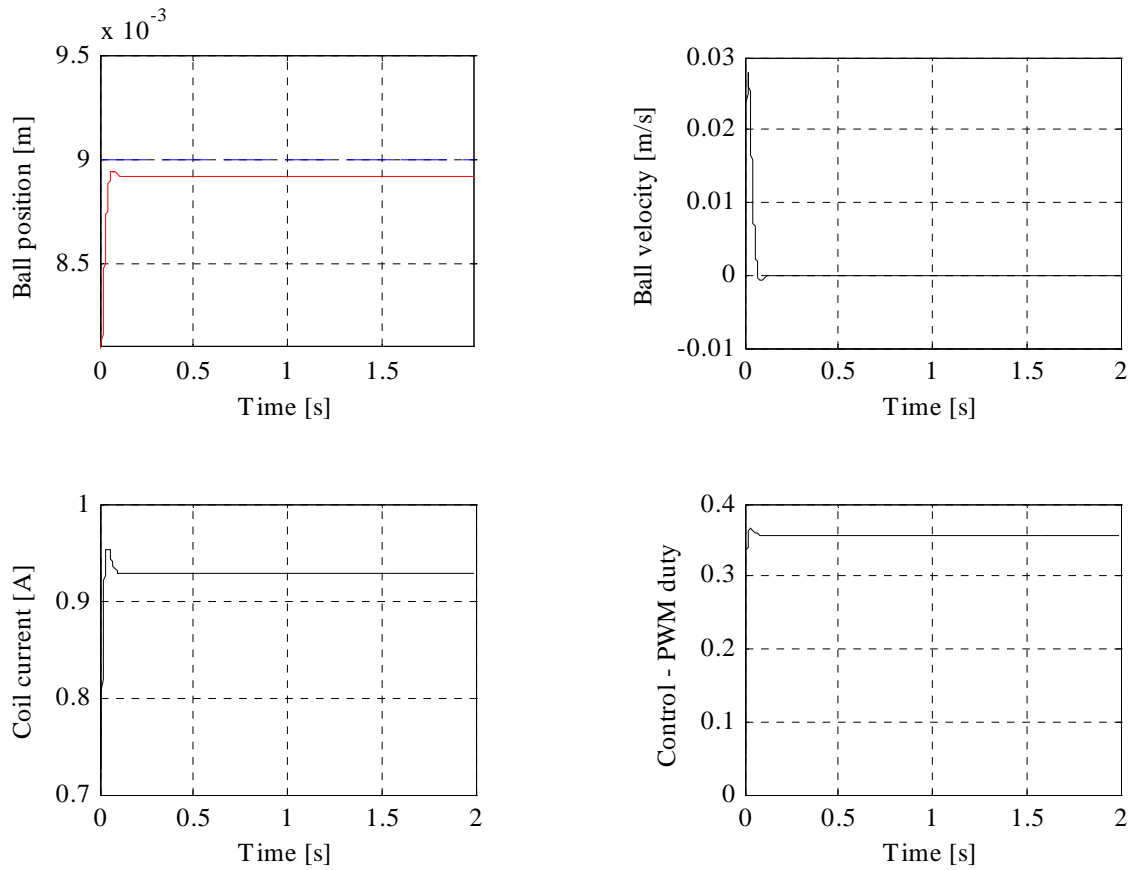


Fig. 51. LQ tracking simulation – the desired position is a constant

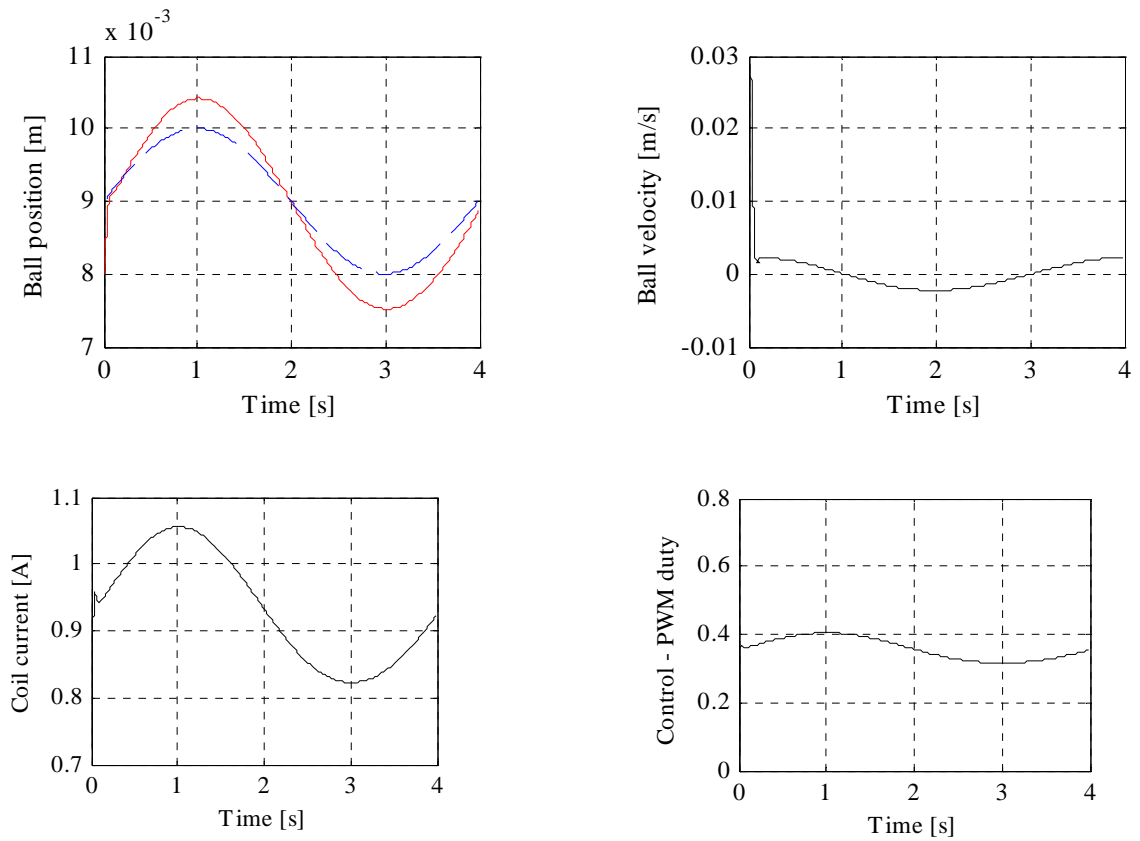


Fig. 52. LQ tracking simulation – the desired position is in a sine wave form.

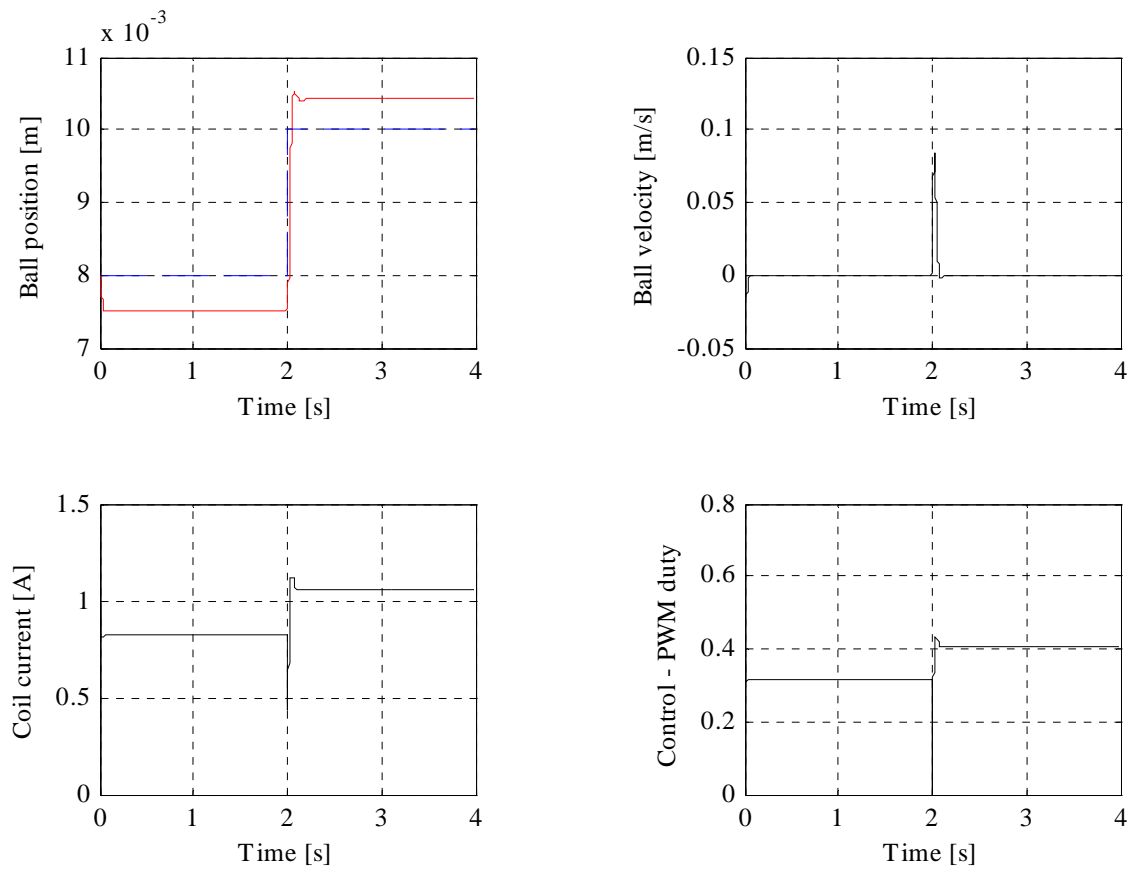


Fig. 53. LQ tracking simulation – the desired position is in a square wave form.

2.4 Levitation

All simulation experiments can be repeated as real-time experiments. In this way one can verify accuracy of modelling. If we double click the *levitation* button in the ML *Main* window the following window opens (see Fig. 54).



Fig. 54. Experimental controllers

Now, we can choose the controller we are interested in. We start from the PID control.

2.4.1 PID

Double click the PID button. The real-time PID controller opens (see Fig. 55). The results of the real-time experiment are shown in: Fig. 56, Fig. 57 and Fig. 58.

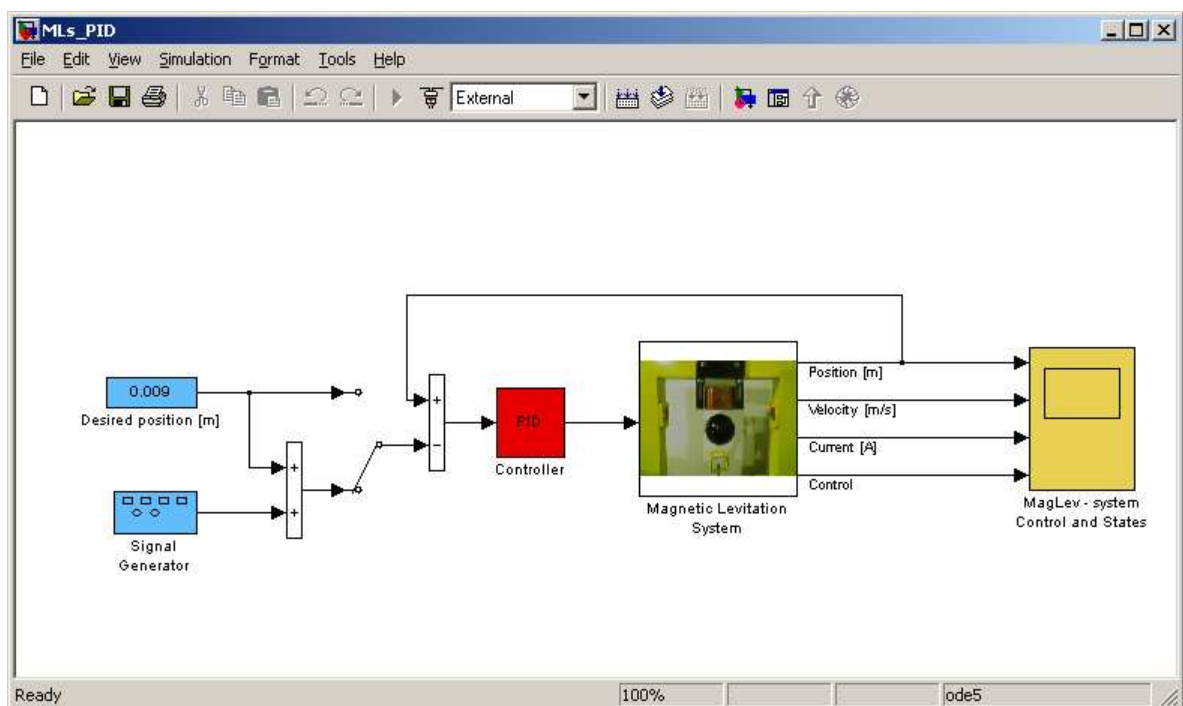


Fig. 55. PID real-time experiment.

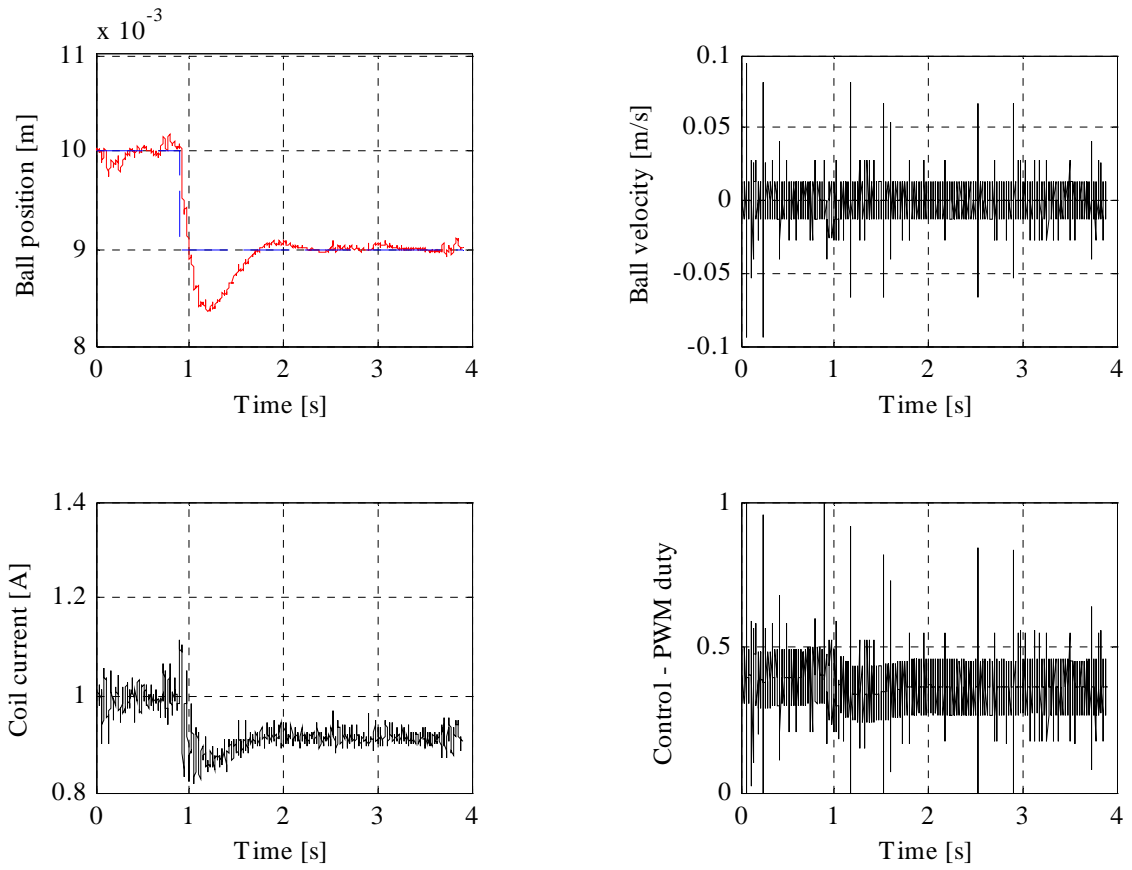


Fig. 56. PID real-time experiment. The desired position as a constant.

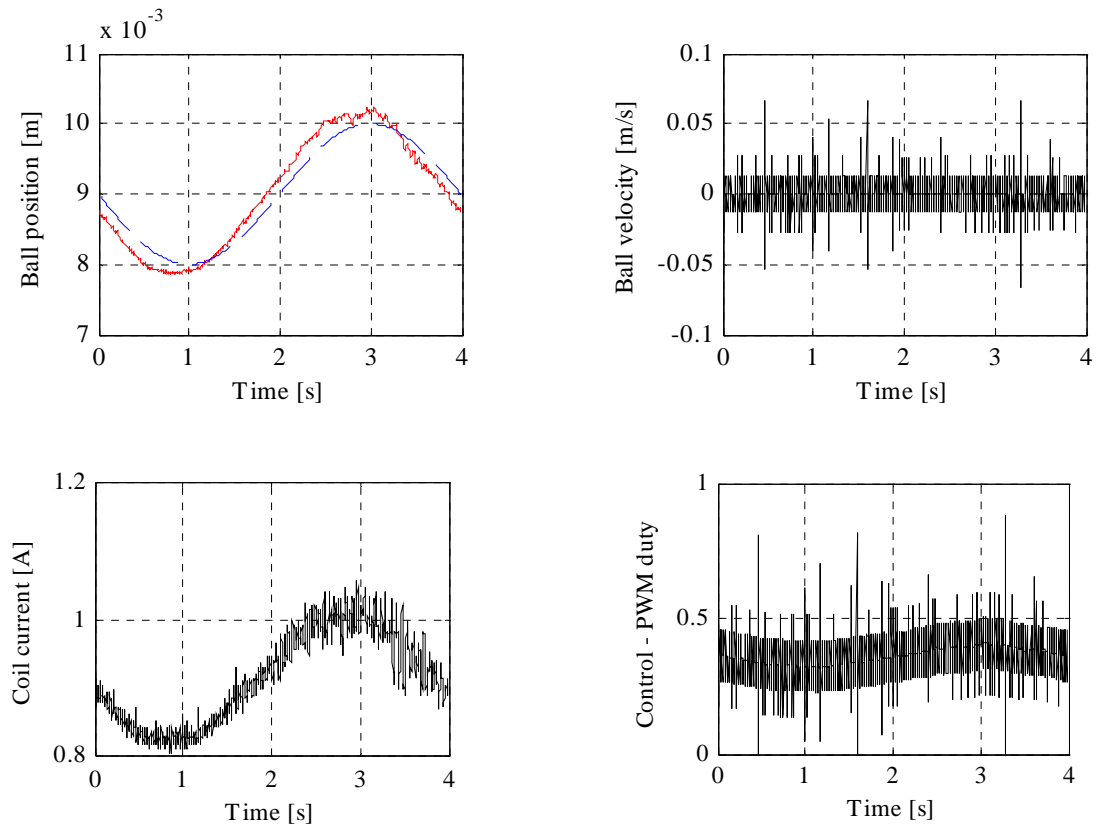


Fig. 57. PID real-time experiment. The desired position in a sine wave form.

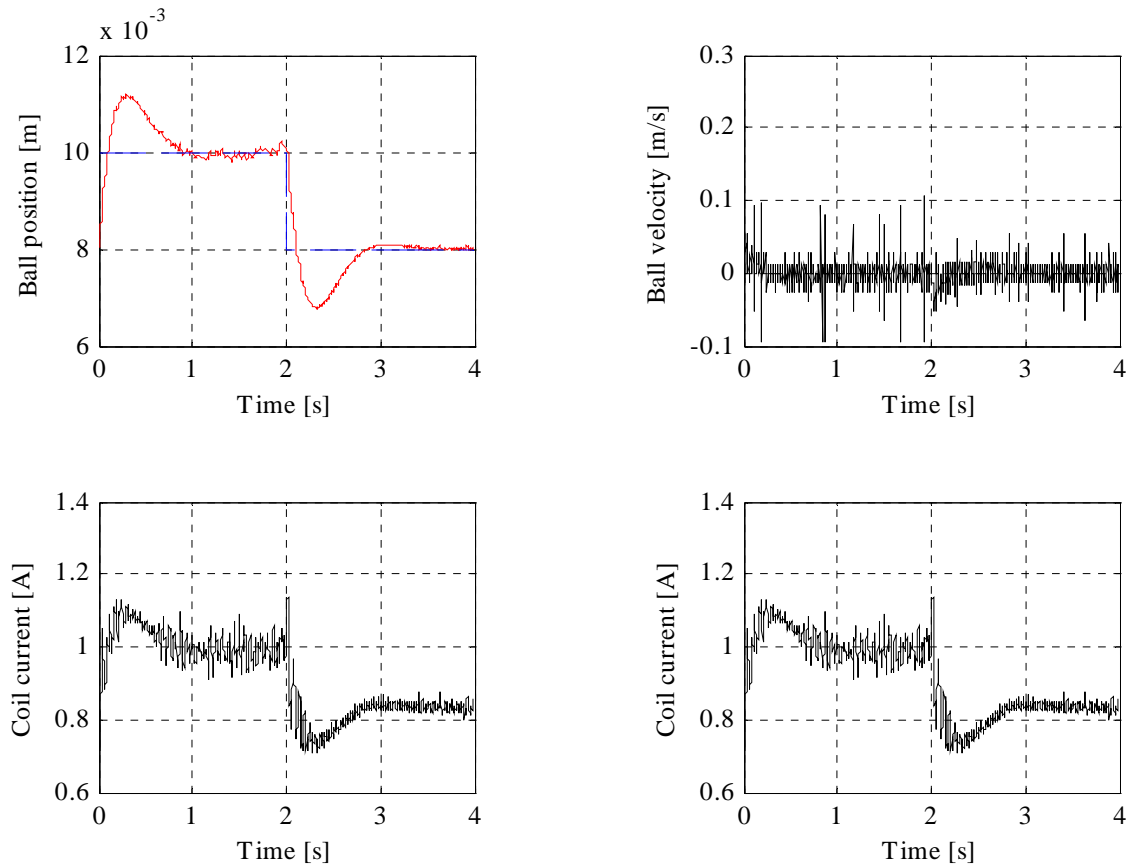


Fig. 58. PID real-time experiment. The desired position in a square wave form.

2.4.2 LQ

Double click the LQ button. The real-time LQ controller opens (see Fig. 59). The results of the real-time experiment are shown in: Fig. 60, Fig. 61 and Fig. 62.

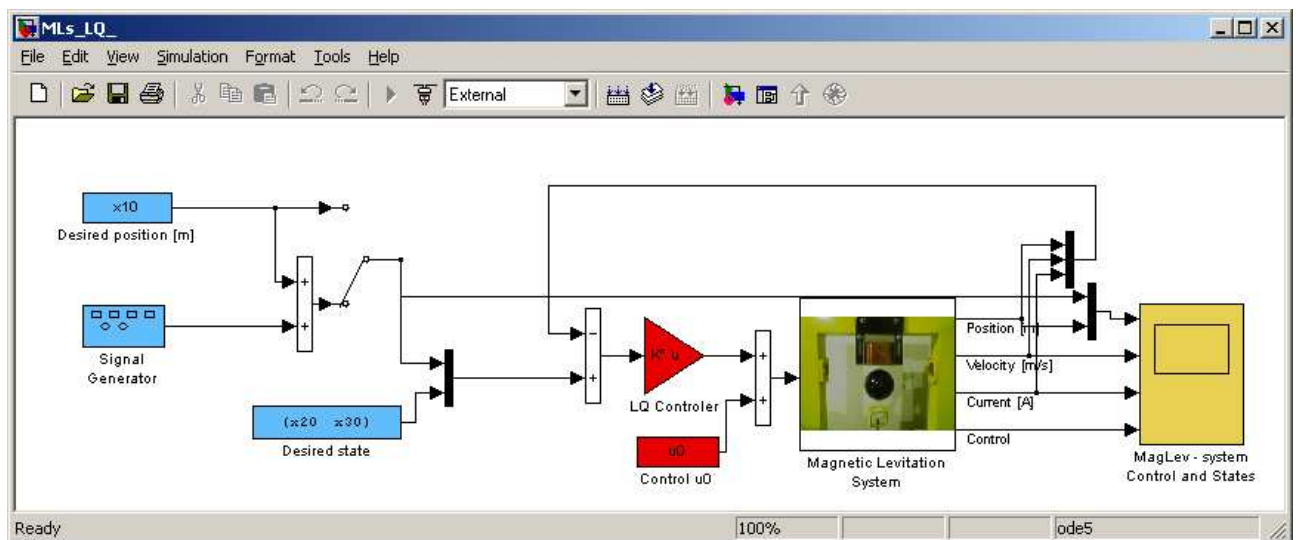


Fig. 59. LQ real-time experiment.

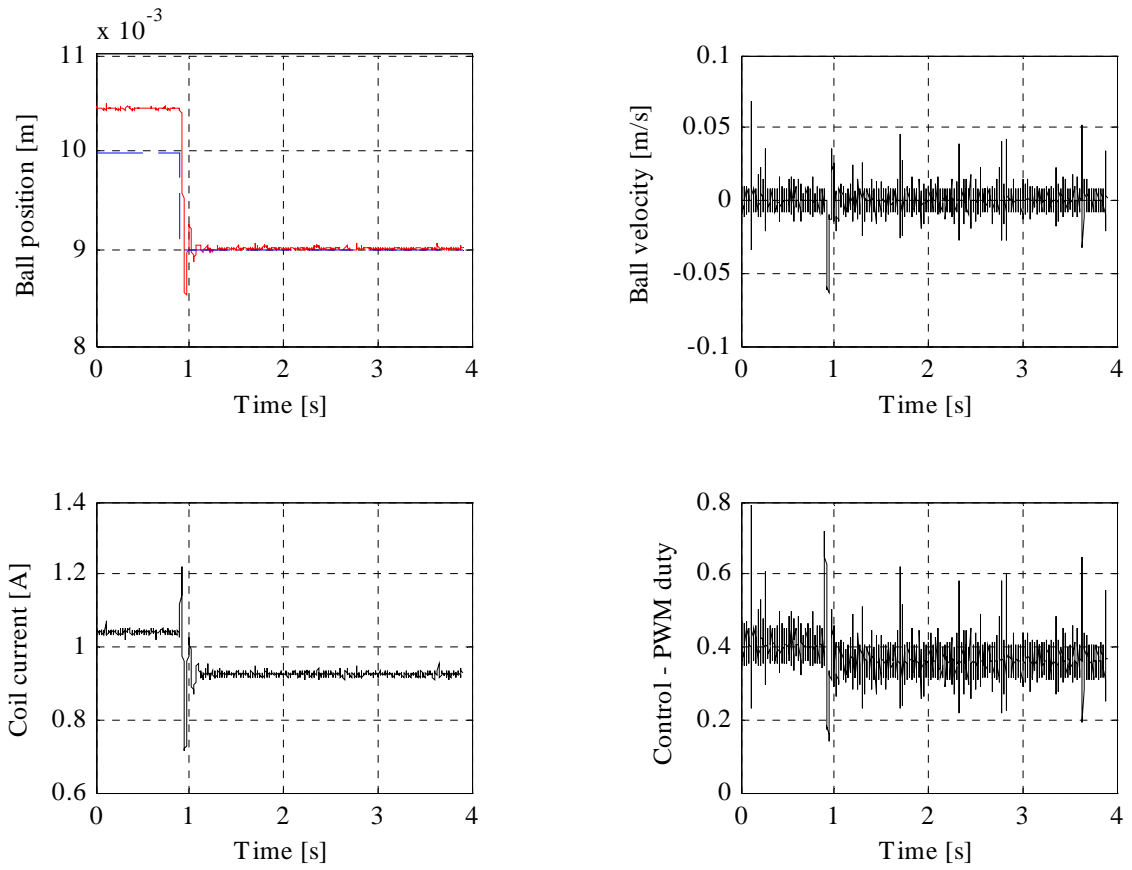


Fig. 60. LQ real-time experiment. The desired position as a constant.

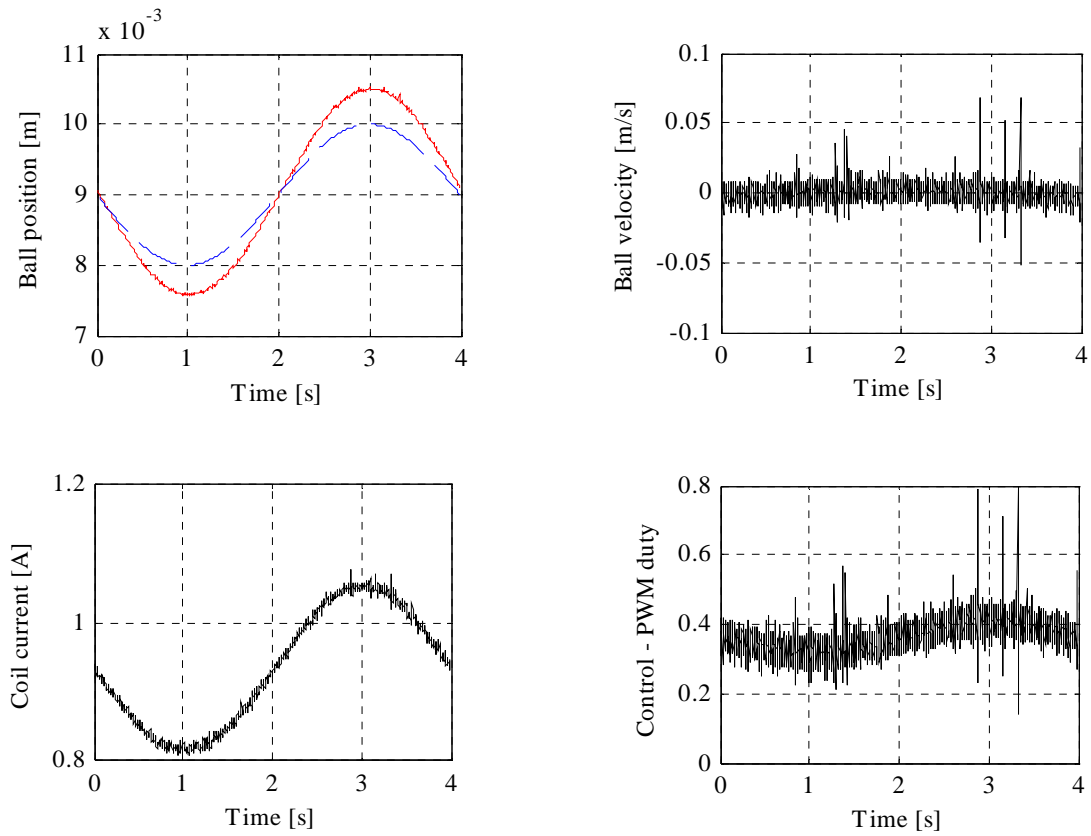


Fig. 61. LQ real-time experiment. The desired position in a sine wave form.

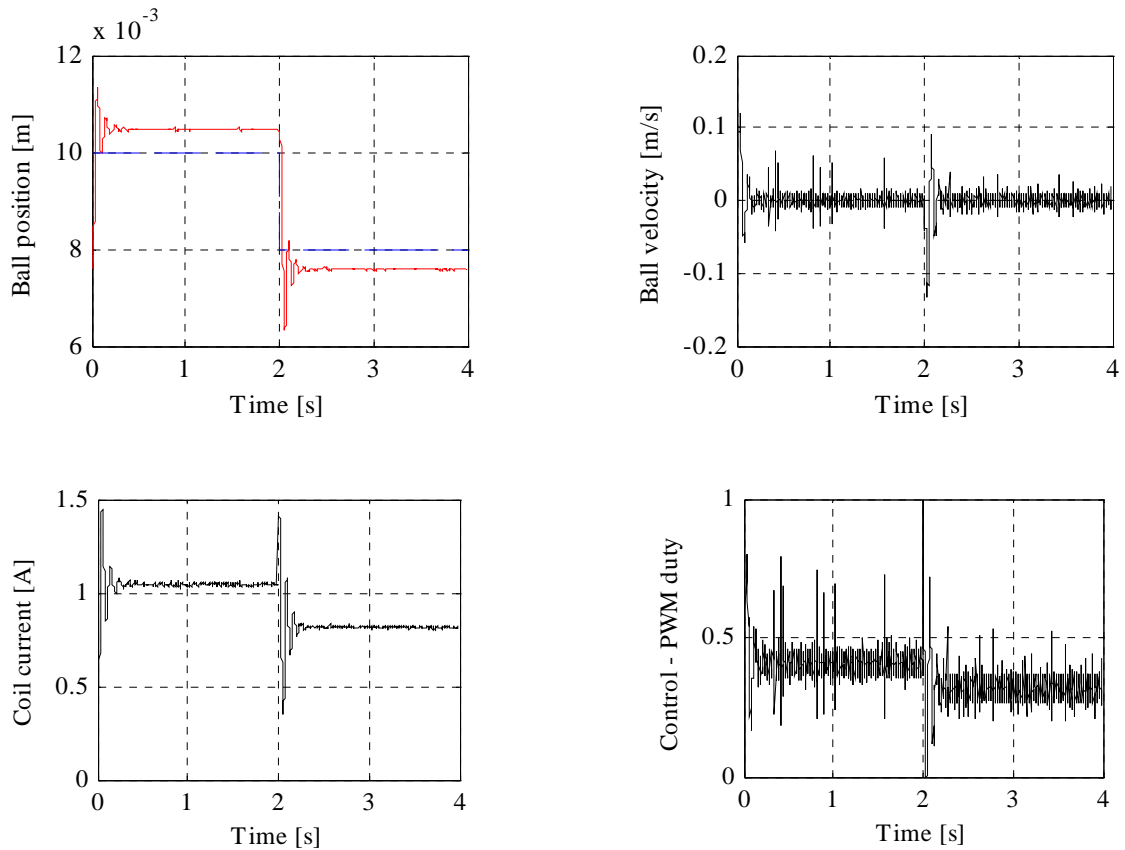


Fig. 62. LQ real-time experiment. The desired position in a square wave form.

2.4.3 LQ tracking

Double click the LQ tracking button. The real-time LQ tracking controller opens (see Fig. 63). The results of the real-time experiment are shown in Fig. 64, Fig. 65 and Fig. 66.

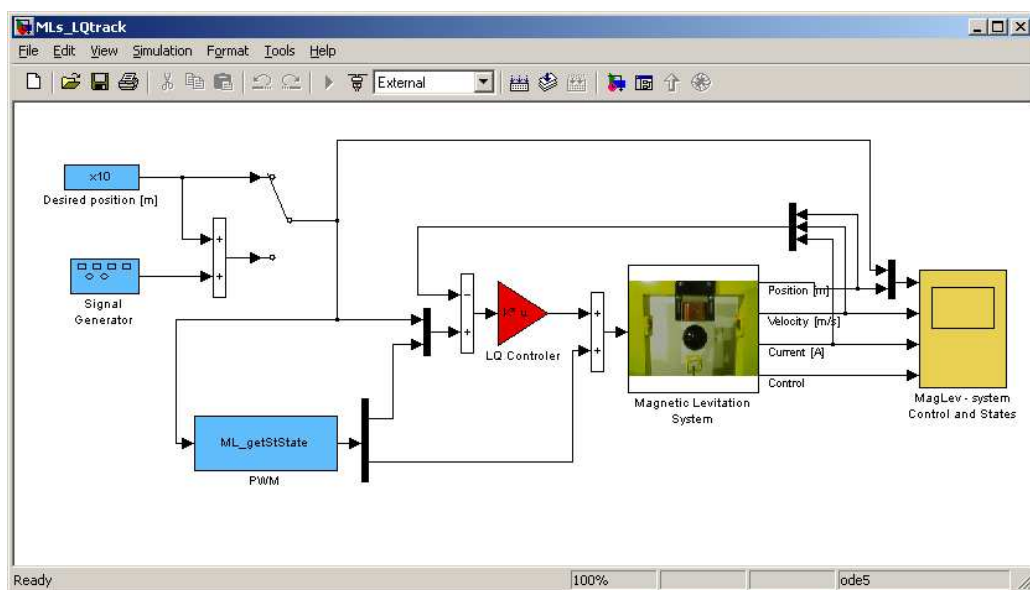


Fig. 63. LQ tracking real-time experiment.

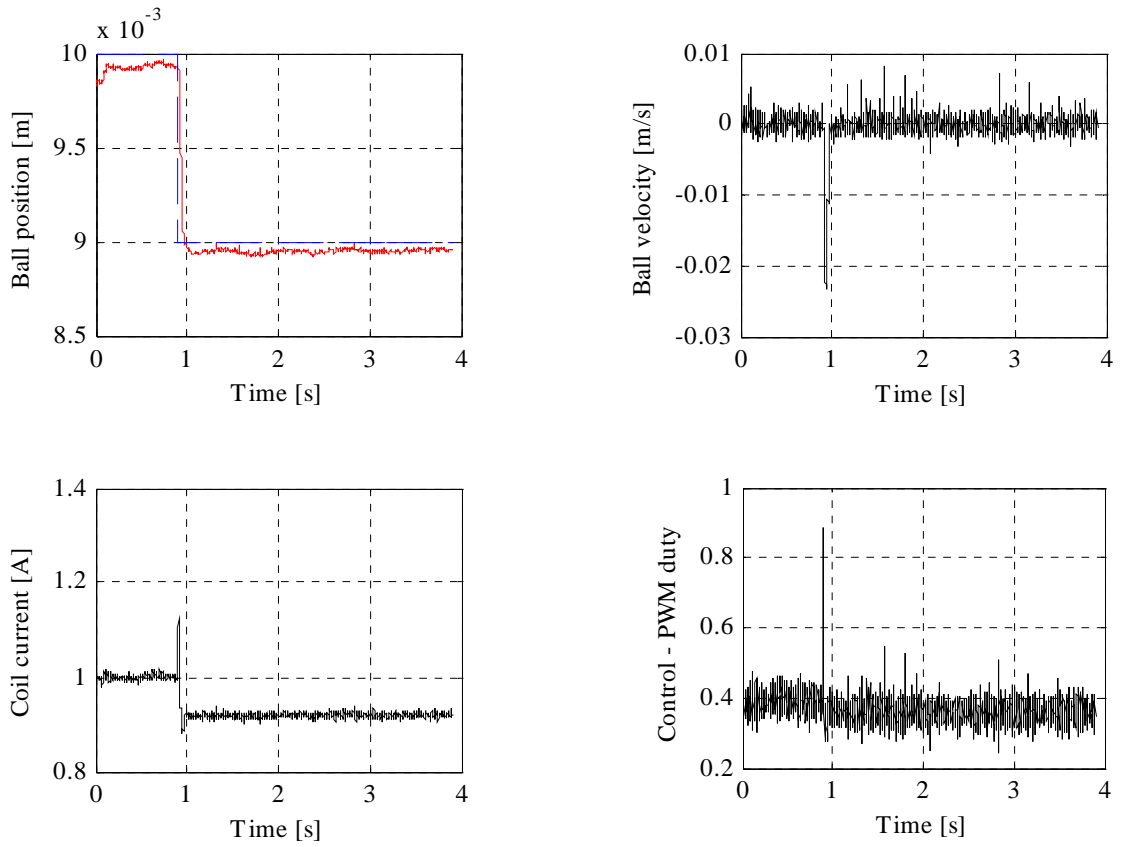


Fig. 64. LQ tracking real-time experiment. The desired position as a constant.

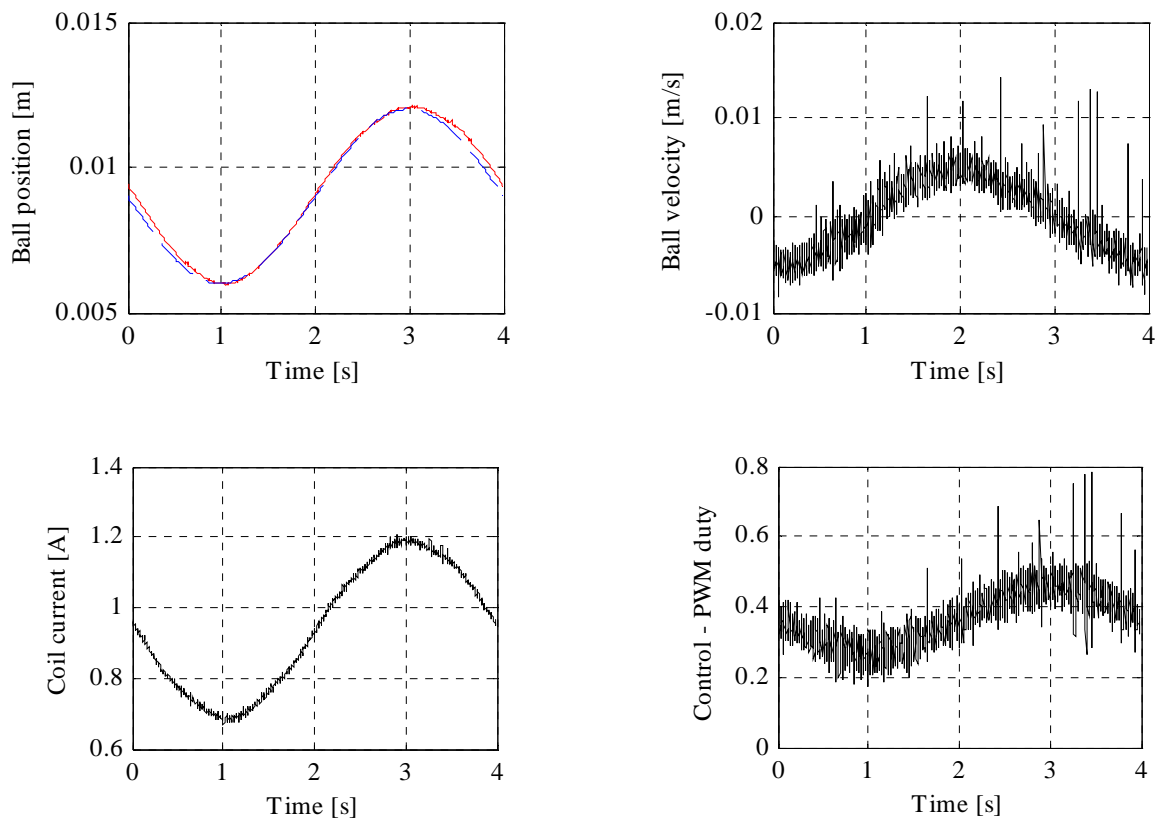


Fig. 65. LQ tracking real-time experiment. The desired position in a sine wave form.

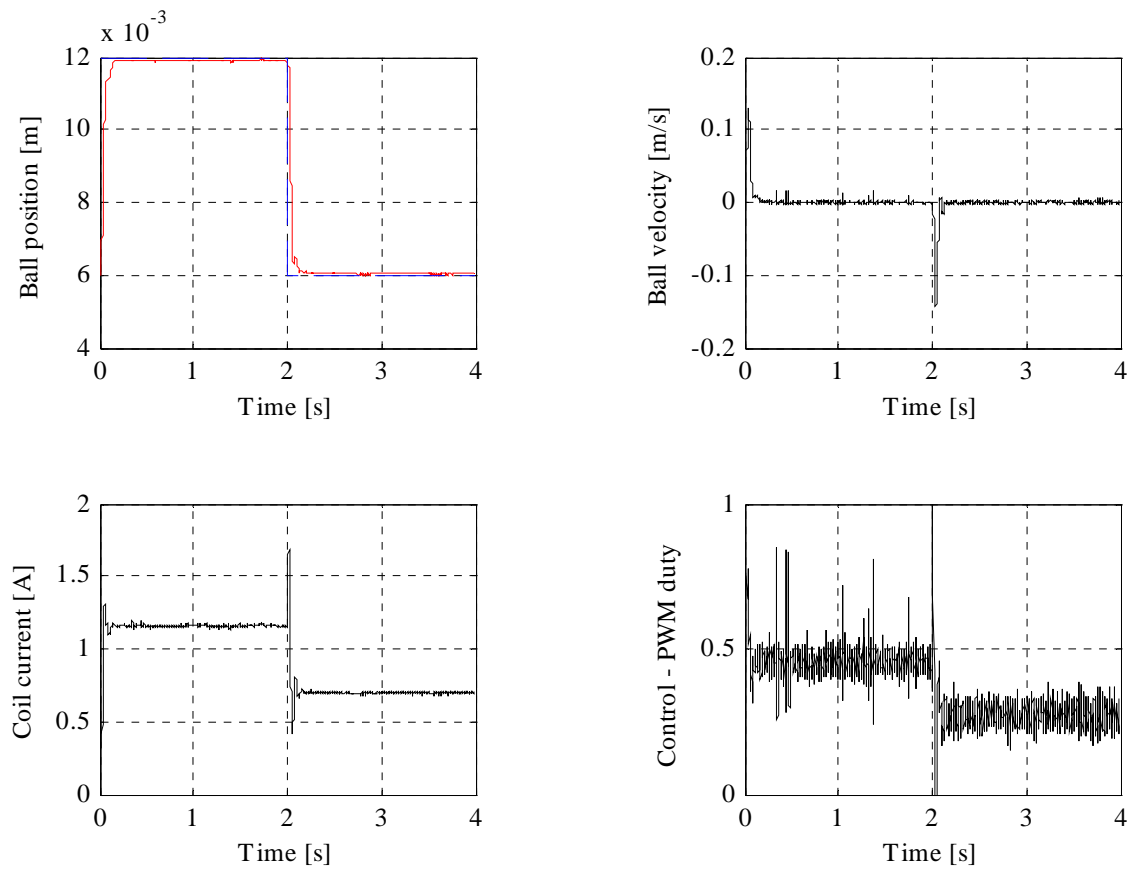


Fig. 66. LQ tracking real-time experiment. The desired position in a square wave form.

3 Description of the Magnetic Levitation class properties

The *MagLev* is a MATLAB class, which gives the access to all the features of the RT-DAC4/PCI board supported with the logic for the MLS model. The RT-DAC4/PCI board is an interface between the control software executed by a PC computer and the power-interface electronic of the modular servo model. The logic on the board contains the following blocks:

- PWM generation block – generates the Pulse-Width Modulation output signal. Simultaneously the direction signal and the brake signal are generated to control the power interface module. The PWM prescaler determines the frequency of the PWM wave;
- power interface thermal status –the thermal status can be used to disable the operation of the overheated actuator unit;
- interface to the on-board analog-to-digital converter. The A/D converter is applied to measure the position of the ball (light sensor) and to measure the coil current of the actuator.

All the parameters and measured variables from the RT-DAC4/PCI board are accessible by appropriate properties of the *MagLev* class.

In the MATLAB environment the object of the *MagLev* class is created by the command:

object_name = *MagLev*;

for example *ml* = *maglev*;

The *get* method is called to read a value of the property of the object:

property_value = *get*(*object_name*, '*property_name*');

The *set* method is called to set new value of the given property:

set(*object_name*, '*property_name*', *new_property_value*);

The *display* method is applied to display the property values when the *object_name* is entered in the MATLAB command window.

This section describes all the properties of the *MagLev* class. The description consists of the following fields:

Purpose	Provides short description of the property
Synopsis	Shows the format of the method calls
Description	Describes what the property does and the restrictions subjected to the property
Arguments	Describes arguments of the set method
See	Refers to other related properties
Examples	Provides examples how the property can be used

3.1 BaseAddress

Purpose: Read the base address of the RT-DAC4/PCI board.

Synopsis: *BaseAddress = get(ml, 'BaseAddress');*

Description: The base address of RT-DAC4/PCI board is determined by computer. Each *CML* object has to know the base address of the board. When a *CML* object is created the base address is detected automatically. The detection procedure detects the base address of the first RT-DAC4/PCI board plugged into the PCI slots.

Example: Create the MagLev object:

```
ml = MagLev;
```

Display their properties by typing the command:

```
ml
```

```
Type:                InTeCo ML object
BaseAddress:         54272 / D400 Hex
Bitstream ver.:      x901
Input voltage:       [ 0.8451  0.0244 ][V]
PWM:                 [ 0 ]
PWM Prescaler:       [ 0 ]
Thermal status:      [ 0 ]
Time:                0.00 [sec]
```

Read the base address:

```
BA = get( ml, 'BaseAddress' );
```

3.2 BitstreamVersion

Purpose: Read the version of the logic stored in the RT-DAC4/PCI board.

Synopsis: *Version = get(ml, 'BitstreamVersion');*

Description: The property determines the version of the logic design of the RT-DAC4/PCI board. The magnetic levitation models may vary and the detection of the logic design version makes it possible to check if the logic design is compatible with the physical model.

3.3 PWM

Purpose: Set the duty cycle of the PWM wave.

Synopsis: *PWM = get(ml, 'PWM');*
set(ml, 'PWM', NewPWM);

Description: The property determines the duty cycle and direction of the PWM wave. The PWM wave is used to control the electromagnet so in fact this property is responsible for the electromagnet control signal. The *NewPWM* variable is a scalar in the range from 0 to 1. The value of +1 means the maximum control, 0.0 means zero control.

See: *PWMPrescaler*

Example: `set(ml, 'PWM', [0.5]);`

3.4 PWMPrescaler

Purpose: Determine the frequency of the PWM wave.

Synopsis: `Prescaler = get(ml, 'PWMPrescaler');`
`set(ml, 'PWMPrescaler', NewPrescaler);`

Description: The prescaler value can vary from 0 to 16. The 0 value generates the maximal PWM frequency. The value 16 generates the minimal frequency. The frequency of the generated PWM wave is given by the formula:

$$PWM_{\text{frequency}} = 40\text{MHz} / 4095 * (\text{Prescaler} + 1)$$

See: *PWM*

3.5 Stop

Purpose: Sets the control signal to zero.

Synopsis: `set(ml, 'Stop');`

Description: This property can be called only by the set method. It sets the zero control of the electromagnet and is equivalent to the `set(ml, 'PWM', 0)` call.

See: *PWM*

3.6 Voltage

Purpose: Read two voltage values.

Synopsis: *Volt = get(ml, 'Voltage');*

Description: Returns the voltage of two analog inputs. Usually the analog inputs are applied to measure the ball position and the coil current.

3.7 ThermStatus

Purpose: Read thermal status flag of the power amplifier.

Synopsis: *ThermSt = get(ml, 'ThermStatus');*

Description: Returns the thermal flag of the power amplifier. When the temperature of a power amplifier is too high the flag is set to 1.

3.8 Time

Purpose: Return time information.

Synopsis: *T = get(ml, 'Time');*

Description: The *MagLev* object contains the time counter. When a *MagLev* object is created the time counter is set to zero. Each reference to the *Time* property updates its value. The value is equal to the number of milliseconds which elapsed since the object was created.

3.9 Quick reference table

Property name	Operation *	Description
<i>BaseAddress</i>	R	Read the base address of the RT-DAC4/PCI board
<i>BitstreamVersion</i>	R	Read the version of the logic design for the RT-DAC4/PCI board
<i>PWM</i>	R+S	Read/set the parameters of the PWM wave
<i>PWMPrescaler</i>	R+S	Read/set the frequency of the PWM wave
<i>Stop</i>	S	Set the control signal to zero
<i>Voltage</i>	R	Read the input voltages
<i>ThermStatus</i>	R	Read the thermal flags of the power amplifier
<i>Time</i>	R	Read time information

- R – read-only property, S – allowed only set operation, R+S –property may be read and set