6

E-Learning in Mechatronic Systems Supported by Virtual Experimentation

Viliam Fedák, František Ďurovský and Peter Keusch
Technical University of Košice
Slovakia

1. Introduction

Due to its origin, Mechatronics, consisting of symbiosis of Mechanical Engineering and Electrical Engineering, presents a complex science. Synergistic effects in mechatronic systems and mutual interactions among subsystems of various nature cause considerable difficulties at their teaching and learning. There are plenty of mechatronic systems of various kind, size and complexity which the specialist has to deal at their design with – starting from miniature ones up to large industrial mechatronic systems presented e.g. by continuous production lines.

In every system, regardless its nature, we are interesting in its performance and behavior – whether it is linear or non-linear, oscillating or non-oscillating, which are its time constants, time response, frequency response, placement of system poles and zeros in the plane, how to control the system, which control method to apply, and finally, how to realize control algorithm and its debugging. When designing control algorithms for a complex mechatronic system, students should know and verify behavior of individual subsystems, i.e. they should possess satisfactory knowledge from mechanical, electrical and control engineering.

Like in other fields, also in mechatronic engineering education, the concepts taught through lectures should be completed by practical laboratory experimentation (Cheng & Chiu, 2010). Here the students observe phenomena that are rather difficult to explain by written material. The students are interested in experiments with real models. During experimentation they get practical experience, skills and also a self-confidence that are necessary to solve real problems. But experimentation on real industrial systems usually is out of question. A lot of effort was made in searching new methods – how to create enough space for students' better acknowledgement with the systems and how to train them for practical problems solutions. One way how to fully substitute a physical system consists in system dynamics emulation, as shown in (Potkonjak et al., 2010), but this method requires an ample equipment. Another way consists in sharing expensive equipment by more institutions and creating distance experiments. Well elaborated description of distance experimentation in power electronics is shown in (Bauer, 2008, 1) or a more general approach to a set of distance experiments in several fields of electrical engineering is presented in (Bauer, 2008, 2). Although such distance laboratory satisfies needs for training of students, its development is enough time-consuming and labor-intensive. It requires special equipment connected to internet, good organization of booking system, maintenance of the server, and usually, after several years with changing the staff, the distance laboratory interrupts its activity.

These are reasons why in the first step very often we confide in the simulation resources where the mathematical model in form of differential equations is applied to calculate dynamics and by simulation provides the data that would otherwise be measured on a real system. This approach is rather sophisticated, knowledge demanding, and also time consuming. It requires analytical approach - a good knowledge of the system to be analyzed, mathematical background, and skills programming the simulation model, solving the equations, and finally – visualization of results.

A way to save time and burden at maintenance consists in development of virtual models. They can create an intermediate stage prior accessing students to experimentation on laboratory models, without any special knowledge about their background, mathematical and simulation model development. Numerous projects are running there that are focused to development of virtual models and laboratories and many examples are available nowadays. A general approach to virtual laboratories is described by (Babich & Mavrommatis, 2004) and specially, to virtual laboratory in mechatronics by (Cheng & Chiu, 2010; Pipan et al, 2008). Most effort was done for robotics (Bianchi, 1999; Potkonjak, 2010; Faculty of Electrical Engineering in Belgrade, n.d.) but also many others could be mentioned. In the electrical and control engineering some successful solutions are known, e.g. (Saadat; Petropol-Serb, 2007, Educational Matlab GUIs). Such virtual models are undamageable and their screen can be designed so that they look like apparatus. They can also serve for staff training in system control. The virtual models are accessible via internet and thus, the student can utilize all the features of e-learning - self study, regardless the place and time.

To be effective in design of virtual models there is need for a simple development process of the virtual purpose–oriented model that can run online, directly on the student's computer, and gives enough graphical and numerical information the student requires. The requirements in our case are met by developed virtual models that are based on Graphic User Interface (GUI) MATALB. This is a powerful data-processing tool allowing simulation of dynamic systems in a simple way having user friendly graphic environment.

The organization of the contribution is as follows: after general description of design methodology for virtual models in GUI MATLAB (chapter 2) in the third chapter we describe development of several typical models of simple mechatronic subsystems starting from kinematic diagram, showing briefly their mathematical and simulation models. Emphasis is put on development of the virtual model itself and description of its features. Finally in the fourth chapter we share some experiences from training of students by virtual models and in conclusion we present ideas for our future work.

2. Methodology at design of virtual models

2.1 Procedure at designing GUI MATLAB

The developed virtual models are based on GUI MATLAB. Their development starts from a kinematic scheme of the system and its description by mathematical equations. Manipulating them the mathematical and simulation model are derived. Its dynamics is verified by simulation in Simulink. The GUI MATLAB itself is developed in Graphic User Development Environment (GUIDE), which practically presents a graphic editor. GUIDE contains a set of tools (objects) for user environment design. The development procedure is depicted in Fig. 1.

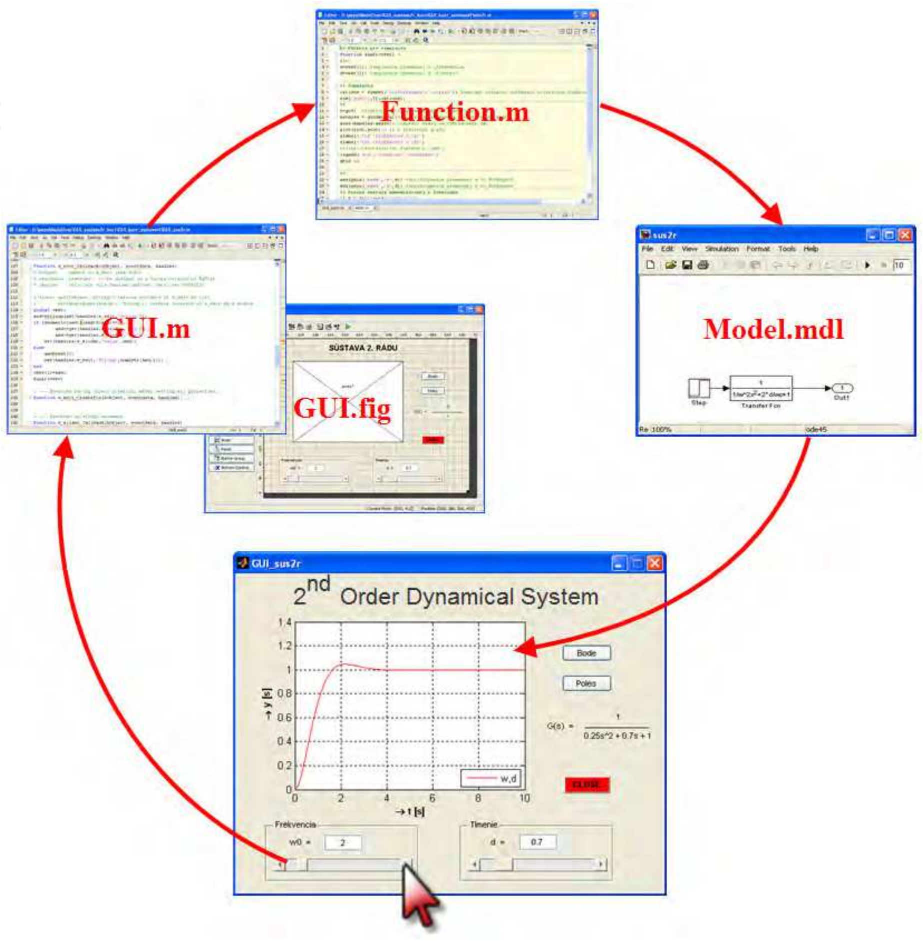


Fig. 1. Procedure of activities in development of virtual model (here based on example of the 2nd order dynamical system)

The editor enables the user to interact directly with elements presented on the screen. These elements (called objects) replace the keyed entry of commands and menus. Users typically select screen objects and actions by using a mouse. The objects navigate the screen and execute commands by using menu bars, buttons, sliders, pull-down menus and editing windows that enable to input characters (incl. numbers). MATLAB automatically generates a relative M-file (GUI.m) containing callbacks interconnected with the objects in the adjoining file GUI.fig. The program written into the callbacks is realized immediately after the object activation (by pressing a button, shifting a slider, inserting a value into editing window, etc.). In the background of GUI a function (Function.m in Fig. 1) collects variable parameters from the GUI screen and sends them into the system model that is built-up in the Simulink environment (Model.mdl). The obtained time response is displayed on a graph placed in a frame on the GUI screen. Similarly, using suitable MATLAB commands it is possible to obtain frequency characteristics, system poles placement and many other features that can be programmed in MATLAB.

2.2 Principles of Imaging and Ergonomics of the Virtual Model GUI Control Screen

The GUI providing human-computer interaction is one of the most important parts in the functional virtual model. To make learning easier, the learner should interact with the computer easy, intuitively, without any learning a complex control of the virtual model and he should get required information in a transparent, well-arranged form.

The GUI for a virtual model has two components:

- *Input*, where the user inputs system parameters, chooses mode of calculation and displays the figures (pictures like kinematic scheme, sketch/photo of device/ mechanism etc., block diagram, mathematical model – equations, etc.) and chooses number and form of required outputs – alphanumerical values and/or graphs. They are usually controlled by buttons/radio buttons, sliders, and alphanumerical editing windows.
- *Output,* where the information is displayed in usable form: in the graphic one (in our case time responses, frequency responses, zero-pole map but also figures as mentioned above) or in an alphanumeric form (values of transfer function coefficients, values of system poles and system zeros, tables, etc.).

To develop a suitable GUI screen for the mechatronic system virtual model, whose visual appearance could be enough complex, the designer must understand principles of good interface and screen design. Generally, the rules are described in (Galitz, 2007). Based on the general rules we adapt and extend them for design of virtual model GUI MATLAB screen. Some rules can be kept intuitively but for proper design, the designer should know all features and possibilities. The most important principles, when designing the placement objects on GUI screen, are:

- *Legibility* — information should be noticeable and distinguishable.
- *Readability* — information is identifiable and interpretable.
- *Coloring display features* — in order to attract and call attention to different screen elements.
- *Guiding the eye* by placement and grouping command objects by visual lines.

Further the designer should deal with the user considerations, as follows:

- Visually pleasing (user friendly) *composition of the screen*.
- *Organizing screen elements*.
- *Screen navigation* and flow.
- Choice of implicitly *pre-setting the system parameters* and their range (so that virtual models can be generally used in large ranges of parameter changes).
- *Changing the system parameters* by sliders, inputting numerical values into text editing windows.

Finally, the designer has to maintain a top-to-bottom, left-to-right flow through the screen. Visually pleasing composition of objects on the screen should keep the following qualities (where it is advantageous and meaningful):

- *Balance* – the design elements have an equal weight, left to right, top to bottom. Balance is most often informal or asymmetrical, with elements of different colors, sizes and shapes being positioned to strike the proper relationships.

- *Symmetry* – duplication across horizontal or vertical axes occurs.
- *Regularity* - uniformity of elements based on some principle or plan which is achieved by establishing standard and consistently spaced columns and rows starting points for screen elements, but it is also achieved by using elements similar in size, shape, color, and spacing.
  The opposite of regularity, *irregularity*, exists, when no such plan or principle is apparent. A critical element on a screen will stand out better, however, if it is not regularized.
- *Sequentiality* is a plan of presentation to guide the eye through the screen in a logical order, with the most important information significantly placed. It can be achieved by alignment, spacing, and grouping.
- *Unity* and *uniformity* - using similar sizes, shapes, or colors for related information.
- *Proportion* of the graphs (time responses, frequency characteristics, zero-pole placement in the plane) using the following ration of sides: square (1:1), square root of two (1:1,414), golden rectangle (1:1,618), square root of three (1:1,732), double square (1:2).
- *Simplicit*y consists in combination of elements that result in ease of comprehending and well-arranged objects (inputs, outputs, control) on the screen.
- *Alignment* – in sense of horizontal or columnar alignment of the objects on the screen.
- *Functional groupings* of associated elements - grouping screen element aids in establishing structure, meaningful relationships, and meaningful form.
- Proper *screen-based controls*. From view of ergonomics, the controlling elements should be placed in bottom or on the right side of the screen (avoiding to cross the screen by the mouse pointer when changing the input parameters or mode of the output).
- Provide effective *internationalization* so that people speaking different languages may use the developed virtual models.

3. Virtual models of mechatronic systems

As already mentioned in the Introduction, the mechatronic systems are of complex nature, and consist basically of mechanical, electrical and control subsystems (but also those of other nature, e.g. hydraulic and pneumatic systems are not excluded). Perfect understanding of mechanical subsystems behavior at various combinations of mechanical and technological parameters presents a presumption for further correct design of the control strategy. Thus, there exists a variety of subsystems the learner should learn to be able to understand their behavior and mutual effects.

In this subchapter we are going to present a set of virtual models and describe the screens that were designed applying the principles shown in the precedent subchapter. Some screens of virtual models are more-or less sophisticated but their use and control of modes are intuitive. After presenting system and its virtual model, more examples and variations of applications are presented there.

3.1 Multi-mass rotating systems with elastic connection

In the first phase we have developed virtual models of simple mechanical systems, starting from mechanical multi-mass oscillating systems (in our case 2- and 3-mass rotating and translation ones), which occurs often in industry, like:

- in flexible joints of robots and manipulators,
- in case of drive systems connected with the load by a long flexible shaft,
- in multi-motor drive systems with flexible connection through processed strip,
- at cranes and lifts with elastic rope, etc.

A system with two rotating masses connected by an elastic coupling element presents a very schematic and visual example of an elastic connection (Fig. 2).

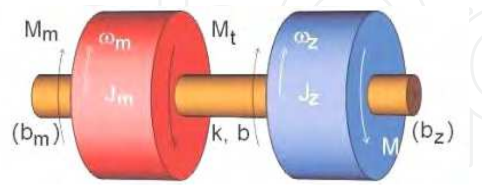


Fig. 2. Kinematic diagram of a rotating system with elastic connecting shaft

The left side consist of a mass disk of the driving motor (moment of inertia J_m and the driving torque M_m) and the load side disk on the right side with the load moment of inertia J_z is loaded by a load torque M_z. The connecting torsional shaft is characterized by its torsional elasticity k and torsional damping b. Similarly, mechanical losses on the driving side are characterized by damping b_m and on the load side by damping b_z).

The system dynamics is described easily by three differential equations:

Motion equation on the motor side: $M_m - M_t = J_m \dfrac{d\omega_m}{dt} + b_m.\omega_m$ (1)

Motion equation on the load side: $M_t - M_z = J_z \dfrac{d\omega_z}{dt} + b_z.\omega_z$ (2)

Torsional torque: $M_t = k(\phi_m - \phi_z) + b.(\omega_m - \omega_z)$ (3)

where the angles are: $\phi_m = \int \omega_m \, dt$ and $\phi_m = \int \omega_m \, dt$ (4)

The corresponding block diagram is shown in Fig. 3. This creates basis for Simulink scheme and core for the virtual model.

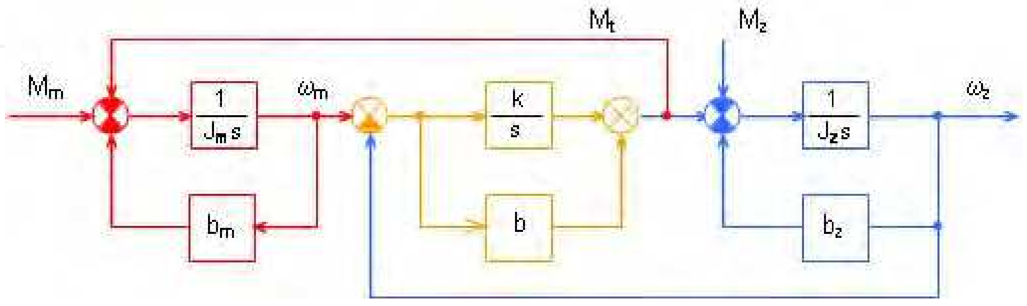
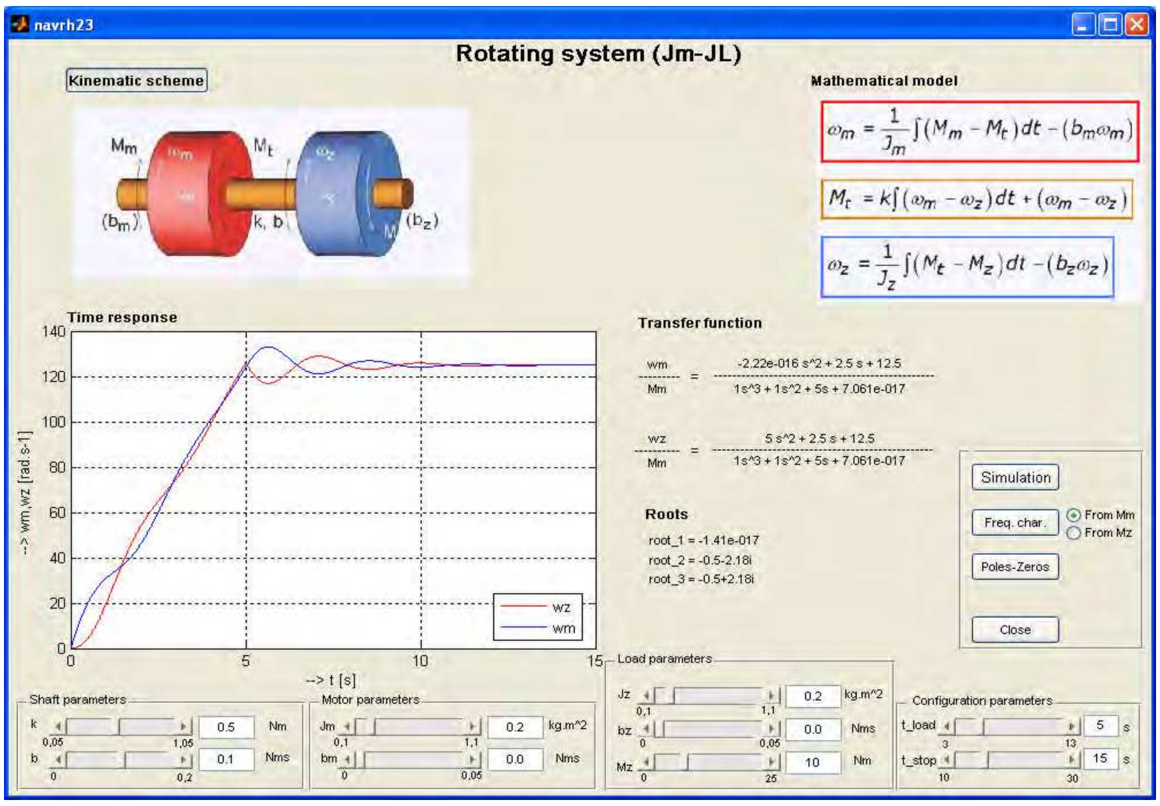


Fig. 3. Block diagram of two-mass rotating system with elastic connection

The GUI screen of the virtual model is shown in Fig. 4. The virtual model response starts immediately after any parameter change. The GUI screen (Fig. 4) consists of several parts:



Rotating system (Jm-JL)

Mathematical model

$\omega_m = \frac{1}{J_m}\int(M_m - M_t)dt - (b_m\omega_m)$

$M_t = k\int(\omega_m - \omega_z)dt + (\omega_m - \omega_z)$

$\omega_z = \frac{1}{J_z}\int(M_t - M_z)dt - (b_z\omega_z)$

Transfer function

```
 wm        -2.22e-016 s^2 + 2.5 s + 12.5
------- = --------------------------------
 Mm         1s^3 + 1s^2 + 5s + 7.061e-017

 wz              5 s^2 + 2.5 s + 12.5
------- = --------------------------------
 Mm         1s^3 + 1s^2 + 5s + 7.061e-017
```

Roots

```
root_1 = -1.41e-017
root_2 = -0.5-2.18i
root_3 = -0.5+2.18i
```

[✓ From Mm] From Mz

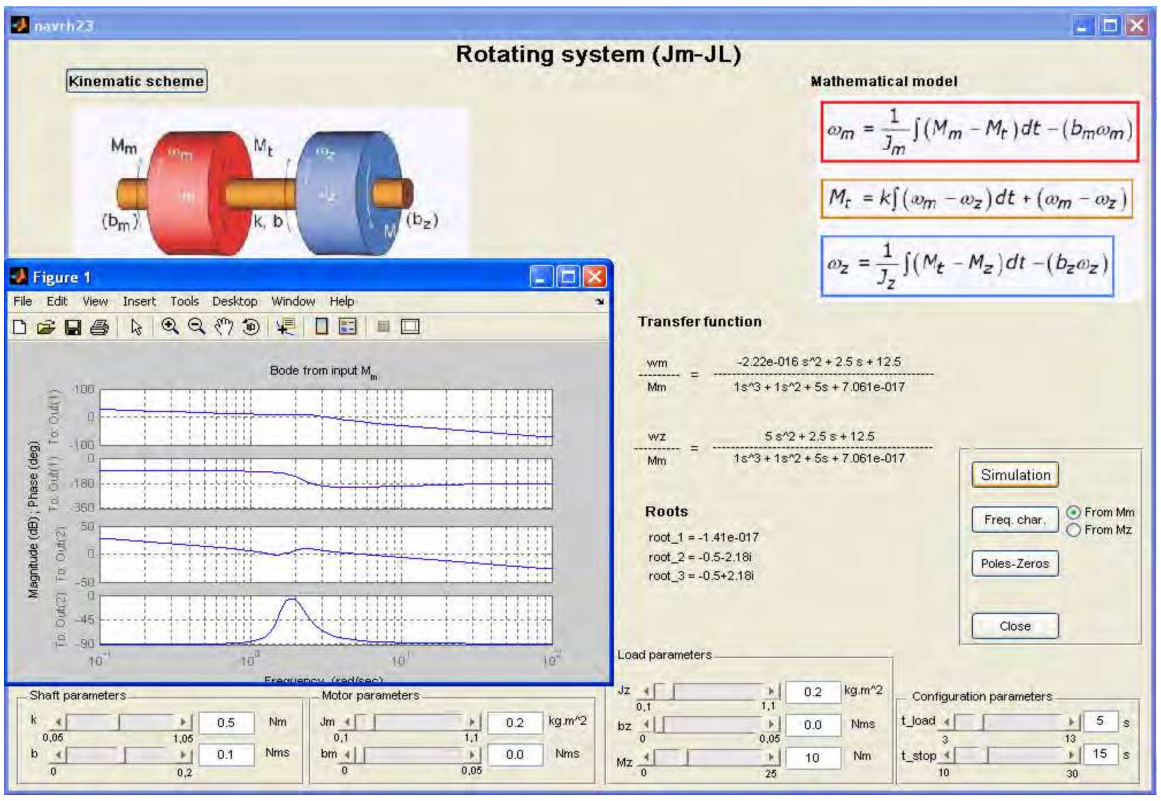
Shaft parameters: k 0.5 Nm (0,05 – 1,05); b 0.1 Nms (0 – 0,2)

Motor parameters: Jm 0.2 kg.m^2 (0,1 – 1,1); bm 0.0 Nms (0 – 0,05)

Load parameters: Jz 0.2 kg.m^2 (0,1 – 1,1); bz 0.0 Nms (0 – 0,05); Mz 10 Nm (0 – 25)

Configuration parameters: t_load 5 s (3 – 13); t_stop 15 s (10 – 30)

Fig. 4. GUI screen of the virtual model of two-mass rotating system with elastic connection



Rotating system (Jm-JL)

Mathematical model

$\omega_m = \frac{1}{J_m}\int(M_m - M_t)dt - (b_m\omega_m)$

$M_t = k\int(\omega_m - \omega_z)dt + (\omega_m - \omega_z)$

$\omega_z = \frac{1}{J_z}\int(M_t - M_z)dt - (b_z\omega_z)$

Transfer function

```
 wm        -2.22e-016 s^2 + 2.5 s + 12.5
------- = --------------------------------
 Mm         1s^3 + 1s^2 + 5s + 7.061e-017

 wz              5 s^2 + 2.5 s + 12.5
------- = --------------------------------
 Mm         1s^3 + 1s^2 + 5s + 7.061e-017
```

Roots

```
root_1 = -1.41e-017
root_2 = -0.5-2.18i
root_3 = -0.5+2.18i
```

[✓ From Mm] From Mz

Shaft parameters: k 0.5 Nm (0,05 – 1,05); b 0.1 Nms (0 – 0,2)

Motor parameters: Jm 0.2 kg.m^2 (0,1 – 1,1); bm 0.0 Nms (0 – 0,05)

Load parameters: Jz 0.2 kg.m^2 (0,1 – 1,1); bz 0.0 Nms (0 – 0,05); Mz 10 Nm (0 – 25)

Configuration parameters: t_load 5 s (3 – 13); t_stop 15 s (10 – 30)

Fig. 5. GUI screen of the virtual model of two-mass rotating system with elastic connection, in the new window there is the frequency characteristic

- *System description* in the upper third of the screen:
  – panels Kinematic scheme/Block diagram - a switch above the picture.
  – panel *Mathematical description* (mathematical model).
- *Inputs*, i.e. system parameters ergonomically placed in the bottom part
- *Outputs*: time responses, frequency characteristics (Fig. 5), transfer functions and position of system poles. Value of parameters are set by a slider or inserted into editing boxes:
  – *Shaft parameters*: *k* – shaft elasticity, *b* – shaft damping,
  – *Motor (mechanical) parameters*: J_m – motor moment of inertia,
  b_m – damping corresponding to friction in the motor bearings,
  – *Load parameters*: J_z – load moment of inertia,
  b_Z damping corresponding to friction in the load bearings.
- Controls, placed on the right side: Simulation, Frequency characteristics and Poles-Zeros (a map of system poles and zeros position being drawn in a new window). In the panel Configuration parameters there are: t_stop – simulation time and t_load – time, in which the load torque is applied.

Finally, the button Close closes the virtual model window. The described virtual model is relatively simple. It enables to observe system dynamical analysis in time and frequency domains at varying mechanical parameters and thus the student learns influence of the system parameters to time response, frequency characteristics and position of system poles.

3.2 Variations of mechanical systems with elastic connections

Based on previous example variations of virtual models of multi-mass elastic joints were developed. As a practical example of such system a model of a car axle suspension is shown here (Fig. 6) that consists of a wheel (a simplified model of a car - so called one-quarter car model), its passive suspension and driver seat.

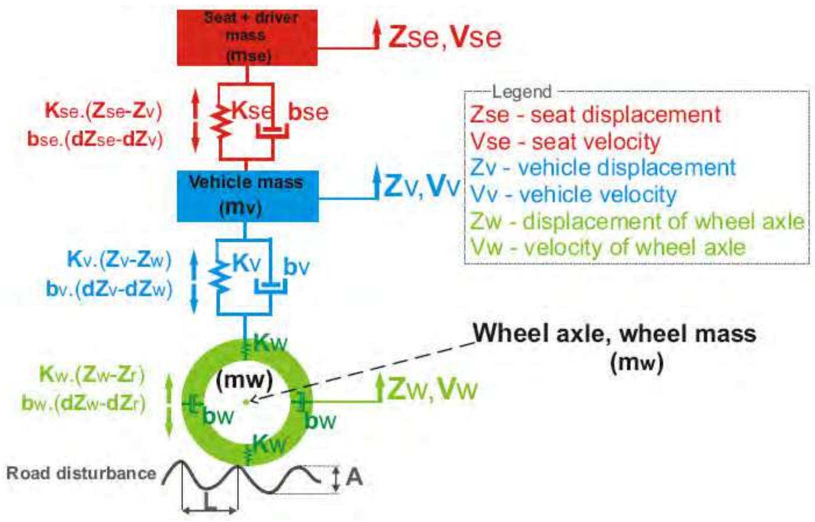


Fig. 6. Kinematic diagram of a car suspension simplified model

As the input variable to the system a disturbance in form of a harmonic signal occurs there that presents a cart-track. The output variables are: vertical position and speed of oscillating masses – of the seat with driver, chassis and wheel itself. Fig. 7 shows a block diagram. The simplified model of the car suspension presents a highly oscillating linear system of the 6th order.

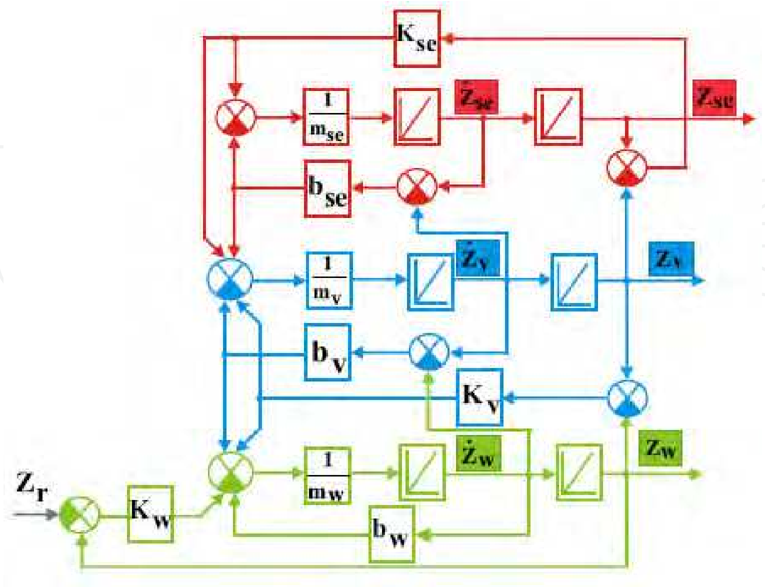
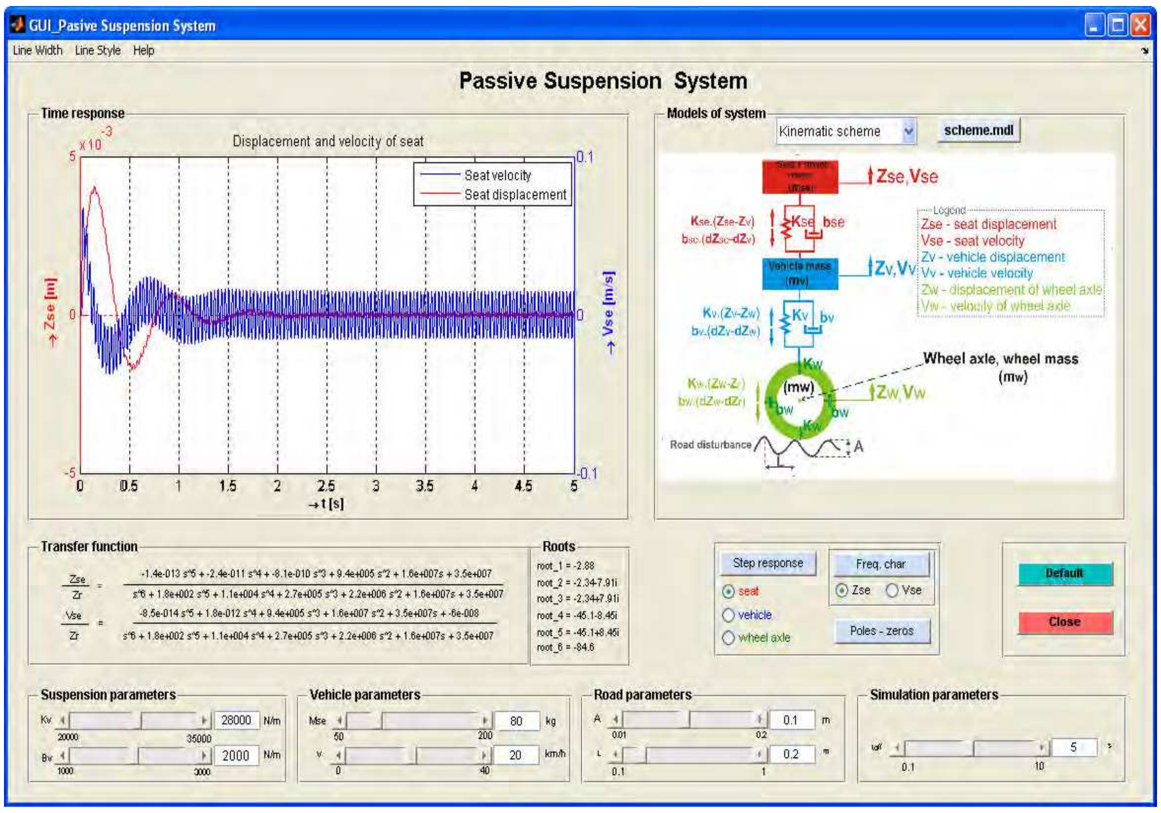


Fig. 7. Block diagram of a car suspension simplified model

Finally, Fig. 8 shows the virtual model. Here the students can tune the parameters of the springs and dampers (for prescribed masses and character of the road surface) and to observe position oscillation of the seat with driver.



Transfer function

$$\frac{Zse}{Zr} = \frac{-1.4e\text{-}013\ s^5 + -2.4e\text{-}011\ s^4 + -8.1e\text{-}010\ s^3 + 9.4e{+}005\ s^2 + 1.6e{+}007s + 3.5e{+}007}{s^6 + 1.8e{+}002\ s^5 + 1.1e{+}004\ s^4 + 2.7e{+}005\ s^3 + 2.2e{+}006\ s^2 + 1.6e{+}007s + 3.5e{+}007}$$

$$\frac{Vse}{Zr} = \frac{-8.5e\text{-}014\ s^5 + 1.8e\text{-}012\ s^4 + 9.4e{+}005\ s^3 + 1.6e{+}007\ s^2 + 3.5e{+}007s + -6e\text{-}008}{s^6 + 1.8e{+}002\ s^5 + 1.1e{+}004\ s^4 + 2.7e{+}005\ s^3 + 2.2e{+}006\ s^2 + 1.6e{+}007s + 3.5e{+}007}$$

Roots

root_1 = -2.88
root_2 = -2.34-7.91i
root_3 = -2.34+7.91i
root_4 = -45.1-8.45i
root_5 = -45.1+8.45i
root_6 = -84.6

[✓ seat] vehicle wheel axle

[✓ Zse] Vse

Suspension parameters

Kv 28000 N/m (20000 – 35000)
Bv 2000 N/m (1000 – 3000)

Vehicle parameters

Mse 80 kg (50 – 200)
v 20 km/h (0 – 40)

Road parameters

A 0.1 m (0.01 – 0.2)
L 0.2 m (0.1 – 1)

Simulation parameters

toff 5 s (0.1 – 10)

Fig. 8. GUI screen of the virtual model of car suspension simplified model

3.3 Single mathematical pendulum

In many institutions, a mathematical pendulum presents one of basic mechanical subsystems for verification of properties of advanced control structures. There are several modifications of the mechanical system – simple pendulum, double pendulum, single pendulum on the chart, single pendulum on the chart with a tension spring, reverse pendulum and many other arrangements that make the mechanical subsystems more complex.

Let's present briefly its mathematical model, and based on this, the properties of its virtual model. The kinematic scheme is shown in Fig. 9.

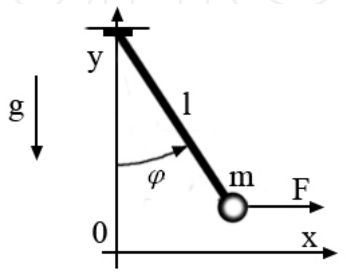


Fig. 9. Kinematic scheme of a simple mathematical pendulum

From the geometry of the pendulum there are easily derived basic equations for position of the simple pendulum:

$$x = l.\sin\phi\,, \tag{5}$$

$$y = l.(1-\cos\phi) \tag{6}$$

The mathematical model is derived using the 2nd order Lagrange equation. Firstly, the equations for the kinetic *T* and potential *V* energy of the system are derived:

$$T = \frac{1}{2}m.\dot{x}^2 = \frac{1}{2}m.\left(l.\dot{\phi}\right)^2 = \frac{1}{2}m.l^2.\dot{\phi}^2 \tag{7}$$

$$V = mgy = mgl\left(1-\cos\phi\right) \tag{8}$$

The 2nd order Lagrange equation is in the form

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} = -\frac{\partial V}{\partial q_j} - \frac{\partial D}{\partial \dot{q}_j} + Q_j \tag{9}$$

After a sort calculation and manipulation of equations we get the final differential equation describing pendulum oscillations, incl. its damping:

$$\ddot{\phi} = \frac{F - b.\dot{\phi} - m.g.l.}{m.l^2}\phi \tag{10}$$

From this equation it is easy to get the value of the angle ϕ. After completing it by the equations (4), (5) describing the system kinematics we can get the block diagram (Fig. 10).

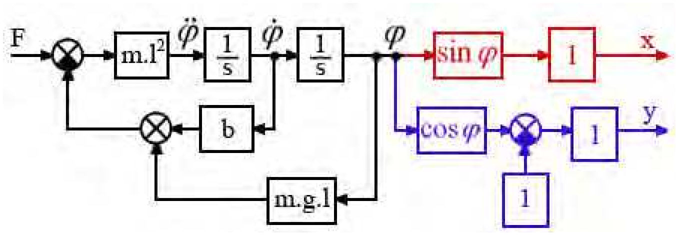


Fig. 10. Block diagram of the simple mathematical pendulum

It creates a base for the virtual model. From control point of view, it is a nonlinear 2nd order system. The corresponding Simulink scheme is shown in Fig. 11.

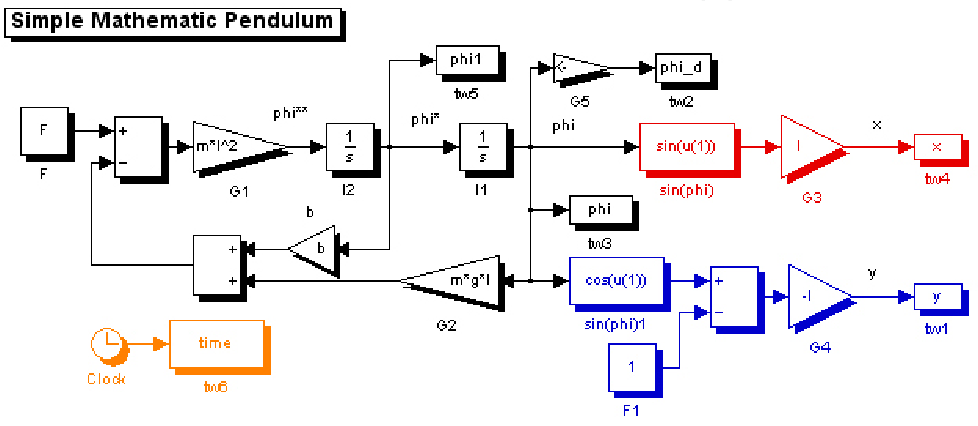


Fig. 11. Simulink block diagram of a simple mathematical pendulum

Based on the Simulink file, the GUI is designed. Its screen is shown in Fig. 12.

Here, the screen consists of 5 parts. Let's describe their functionality in detail:

- *Mechanical model of system*. In this section there occur pictures describing mechanical system model. In the left part there is a block diagram (compare with that in Fig. 6). In the right part the button *Equation/Kinematic scheme* it is possible to switch between the equations describing the mechanical system and kinematic scheme of the system (Fig. 9). The button $\boxed{\text{Mdl File}}$ serves to show Simulink program scheme (Fig. 11) - by opening the proper mdl file.
- *Output characteristics*. Here are two sub-screens to display output characteristics. *Graph A: Time responses* serves to show time responses, in this case the variables *x, y* and *φ*. *Graph B: Functions* serves to show functions - $y = f(x)$ a $\varphi = f(\dot{\varphi})$. To draw a course in some of the graphs, one has to start simulation by pushing a button in the section Simulation.
- *Simulation*. The section serves to control the simulation. The buttons trigger the graphs. It consists of two subsections, namely *Graph A* and *Graph B*.
- The subsection *Graph A* contains the buttons $\boxed{\text{x}}$, $\boxed{\text{phi}}$, $\boxed{\text{y}}$ and $\boxed{\text{x y}}$, that triggers the time courses in the graph *Graph A: Time responses* in the section *Characteristics*.Each button triggers the time course of the depicted variable, i.e. the button buttons $\boxed{\text{x}}$ triggers the time course of the variable *x, etc.* Similarly there works the buttons $\boxed{y = f(x)}$ and $\boxed{\varphi = f(\dot{\varphi})}$ in the subsection *Graph B*, but with the distinction, there are not displayed any time responses but the functions depicted on the button that are draw in the *Graph*

B: Functions. It should be noted that prior simulation the actual input parameters are read and after this the simulation starts.

- *Control Panel*. In this section there are only two buttons to control the whole screen. The first one button $\boxed{\text{Default}}$ serves to reading pre-set values of the mechanical system parameters and their immediate writing into the section *Input parameters* (pre-setting the position of sliders and numerical values in the editing screens). The button $\boxed{\text{Close}}$ finishes the work and closes the GUI screen.
- Input parameters. The section serves to system parameters entry. Each variable has a starting value and a limited range (the limits are set separately), that can be re-activated by pushing the button $\boxed{\text{Default}}$. The value can be changed either by a slider or inserting numeric value into the editing.

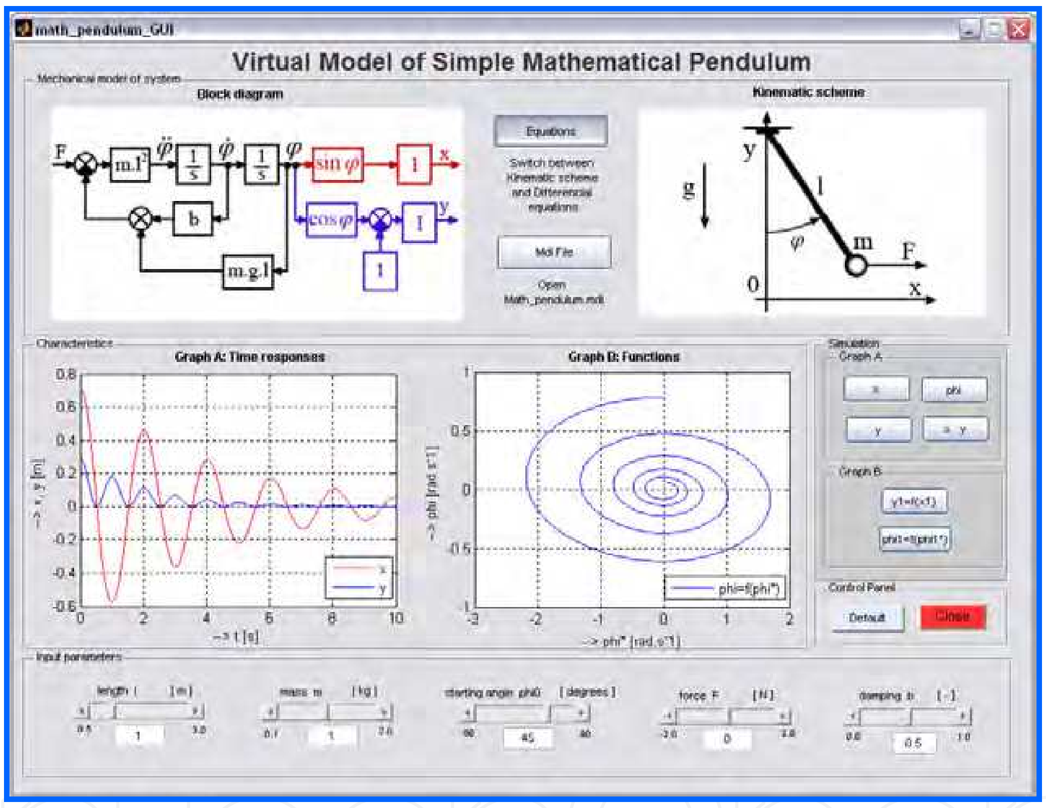


Fig. 12. GUI screen for a simple mathematical pendulum

3.3.1 variations of pendulum model

To get a more complicated system (in order to verify a more complex control algorithms) there were developed some modifications of the basic pendulum model.

Pendulum on the Cart

The pendulum is coupled to a moving cart. This is a principle of a crane trolley and the subsystems serves do develop a control algorithm preventing oscillation of the arm with a load at moving the trolley. The kinematic scheme is shown in Fig. 13.

In comparison with the previous subsystem the simple mathematical pendulum on a cart presents the 4th order non-linear system. In comparison with the previous model the virtual model contains more parameters to be set. Its functionality is similar to that of a simple pendulum; the model only contains more parameter to be set.

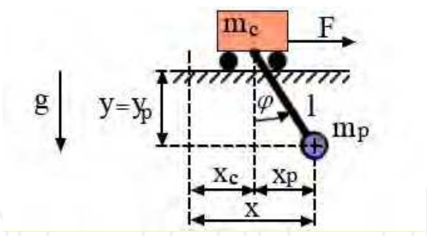


Fig. 13. Kinematic scheme of a simple mathematical pendulum on a cart

Double Mathematical Pendulum

This mechanical subsystem consists of two pendulums where the second one is connected to end of the first one. The kinematic scheme is shown in Fig. 14.

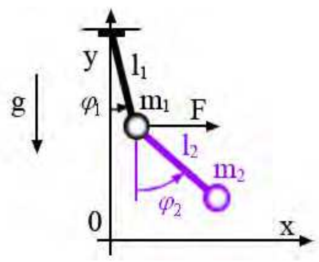
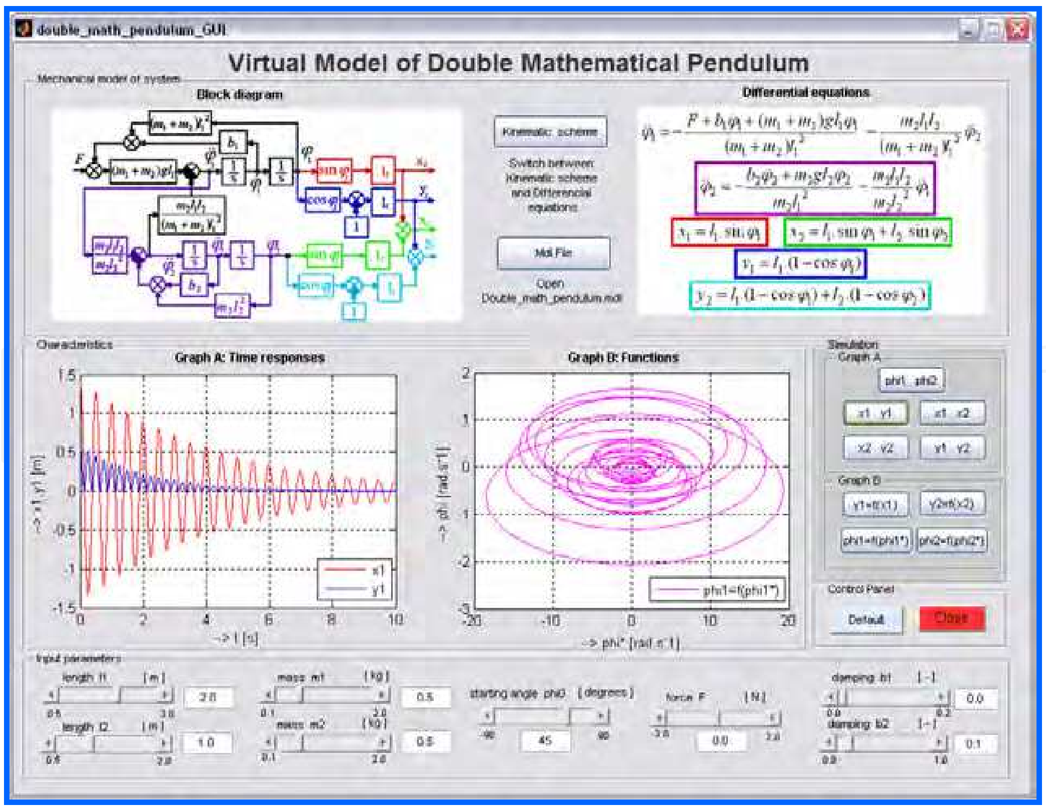


Fig. 14. Kinematic scheme of a double mathematical pendulum



$$\ddot{\varphi}_1 = -\frac{F + b_1\dot{\varphi}_1 + (m_1+m_2)gl_1\varphi_1}{(m_1+m_2)l_1^2} - \frac{m_2 l_1 l_2}{(m_1+m_2)l_1^2}\ddot{\varphi}_2$$

$$\ddot{\varphi}_2 = -\frac{b_2\dot{\varphi}_2 + m_2 g l_2 \varphi_2}{m_2 l_2^2} - \frac{m_2 l_1 l_2}{m_2 l_2^2}\ddot{\varphi}_1$$

$x_1 = l_1 \sin\varphi_1$ $x_2 = l_1\sin\varphi_1 + l_2\sin\varphi_2$

$y_1 = l_1(1-\cos\varphi_1)$

$y_2 = l_1(1-\cos\varphi_1) + l_2(1-\cos\varphi_2)$

length l1 [m]	2.0	mass m1 [kg]	0.5	starting angle phi0 [degrees]	45	damping b1 [-]	0.0
length l2 [m]	1.0	mass m2 [kg]	0.5	force F [N]	0.0	damping b2 [-]	0.1

Fig. 15. Virtual model of a double pendulum

It is the 4th order non-linear system but more complex that the previous one. The complexity of the system follows up also from the virtual model screen (Fig. 15).

3.4 Control of a direct current drive in frequency domain

Let's start to develop a virtual model in GUI MATLAB for a speed controlled direct current (DC) motor supplied by DC converter and two control loops: current and speed ones. The design is performed in the frequency domain. Although this is a simple and generally known example, we shall show some features of virtual model construction and enhance model features. The principal scheme of connection the motor (in linear region without a limiter serving for calculation of controllers only) is shown in Fig. 16 and principal connection of the speed controlled DC drive is in Fig. 17.

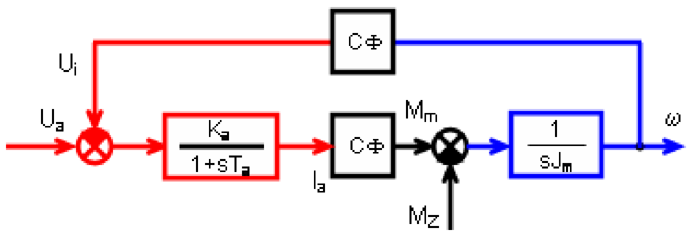


Fig. 16. Block diagram of a DC motor

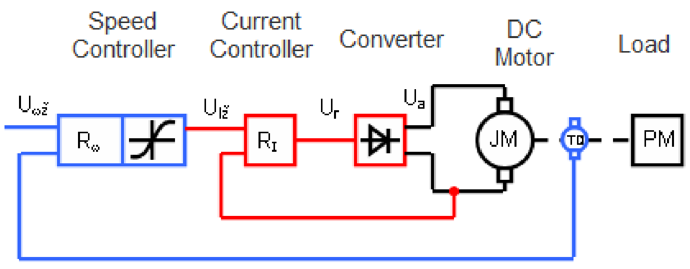


Fig. 17. Principal diagram of connection of speed controlled DC drive

Fig. 18 shows appropriate block diagram of the DC drive and its corresponding Simulink scheme is depicted in Fig. 19.

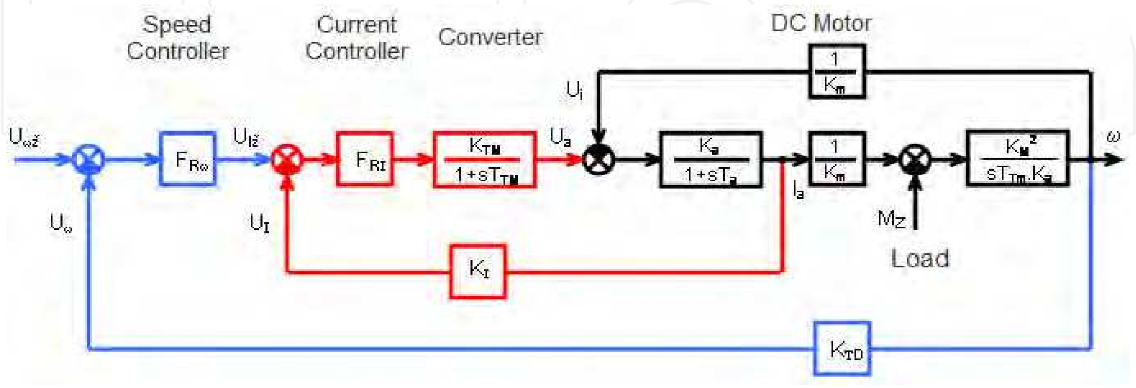


Fig. 18. Block diagram of speed controlled DC drive

The current controller of the PI type is calculated on basis of the Optimum Modulus Criterion from the drive system parameters. General form of its transfer function is:

$$F_{RI} = K_{RI} + \frac{1}{sT_{iI}} \tag{11}$$

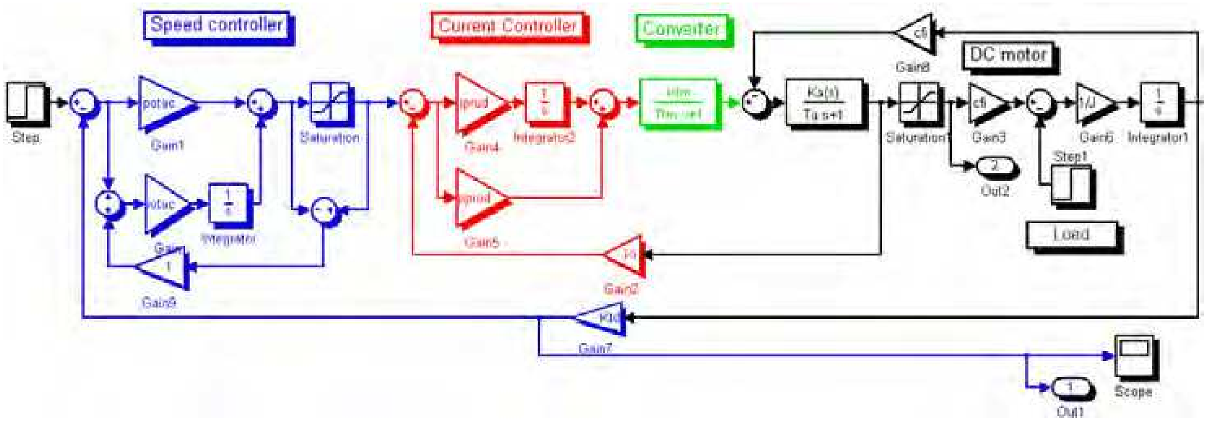
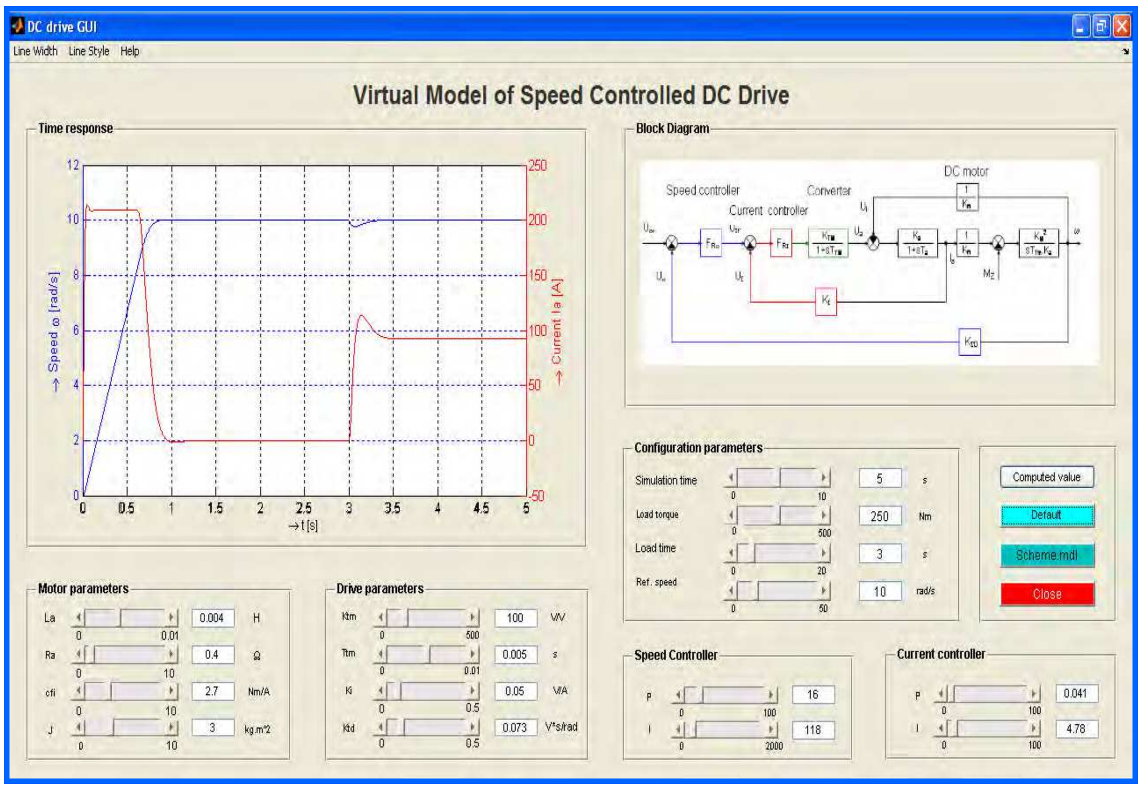


Fig. 19. Interconnection of blocks in the Simulink program

After calculation of current controller parameters and simplification of the current control loop, the speed controller is calculated based on the Symmetrical Optimum Criterion, again of PI type, having the transfer function:

$$F_{R\omega} = K_{R\omega} + \frac{1}{sT_{i\omega}} \tag{12}$$

Fig. 20. shows the basic view on the virtual GUI model of speed controlled DC drive.



Motor parameters: La 0.004 H; Ra 0.4 Ω; cfi 2.7 Nm/A; J 3 kg.m^2

Drive parameters: Ktm 100 V/V; Ttm 0.005 s; Ki 0.05 V/A; Ktd 0.073 V*s/rad

Configuration parameters: Simulation time 5 s; Load torque 250 Nm; Load time 3 s; Ref. speed 10 rad/s

Speed Controller: P 16; I 118

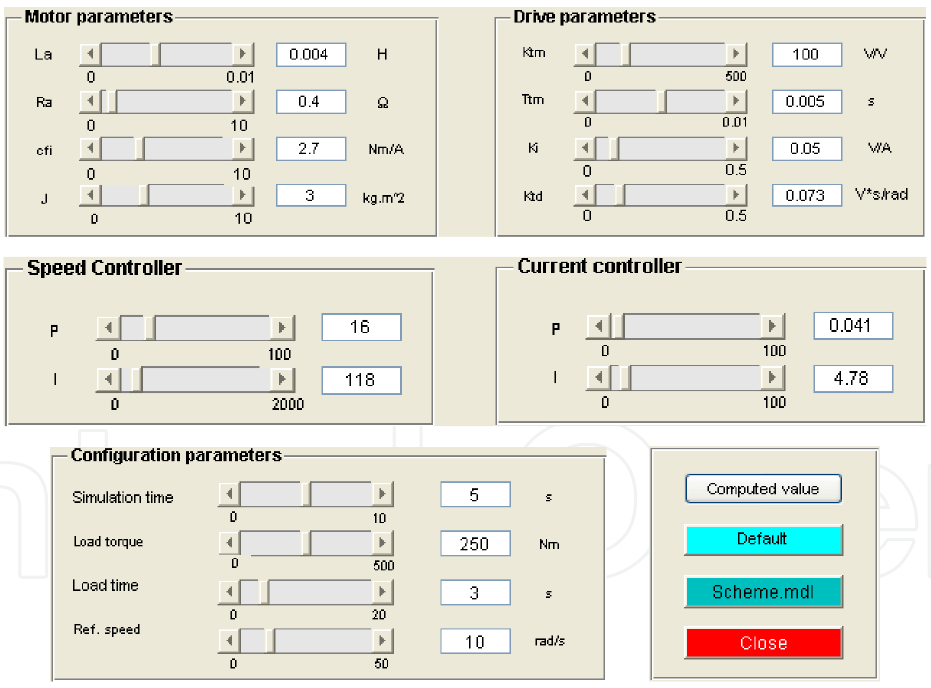
Current controller: P 0.041; I 4.78

Fig. 20. GUI screen for the virtual model of the speed controlled DC drive

Virtual model features:

- *Time response* (the graph on the left side) – here the graph with time courses of the motor current and speed are shown. Immediately after any change of a system parameter (motor -, drive -, or controller parameters) by a slider or inserting a numeric value into editing box the simulation starts and new time responses are drawn, like in a real drive.
- *Block diagram* (the picture on the right side) displays the system block diagram. In other virtual models here can be switched between kinematic diagram, mathematical model, etc.).
- *Change of the system parameters* by sliders or numerical value in the bottom part of the screen. The details of these panels are shown in Fig. 21.

Before starting the model, implicit parameters are set up and they can be changed later. After pushing the button $\boxed{\text{Computed value}}$ the parameters of controllers are calculated from the set values of parameters. There simultaneously appears a small window with the question if the calculated values of controller parameters are acceptable or not. If not, the learner can set up the own parameters and can try to tune them according to the time responses of the drive. To be returned to starting values, the learner pushes the button $\boxed{\text{Default}}$ (similar to the system restart). The advantage of the virtual model consists in the fact that the learner does not need to know the mathematical model running in background and he can concentrate himself to analysis of properties of the drive – to observe influence of a parameter on the system behavior. As shown, the learner can set up his own parameters of the controller (to tune the controllers) or used the pre-calculated parameters based on the pre-set criteria.



Motor parameters

La 0 0.01 0.004 H
Ra 0 10 0.4 Ω
cfi 0 10 2.7 Nm/A
J 0 10 3 kg.m^2

Drive parameters

Ktm 0 500 100 V/V
Ttm 0 0.01 0.005 s
Ki 0 0.5 0.05 V/A
Ktd 0 0.5 0.073 V*s/rad

Speed Controller

P 0 100 16
I 0 2000 118

Current controller

P 0 100 0.041
I 0 100 4.78

Configuration parameters

Simulation time 0 10 5 s
Load torque 0 500 250 Nm
Load time 0 20 3 s
Ref. speed 0 50 10 rad/s

Fig. 21. Panels with sliders to change system parameters, configuration parameters (parameters of simulation and motor load), and mode of the output

3.5 State control of DC motor (in time domain)

This task is dual to the previous one when the system has state-space controllers. Again it is a more complex GUI involving synthesis of the state-space controllers giving the possibility to change parameters of the calculated controllers and thus to tune the controllers based o time responses. The computing algorithm is different from the previous one and belongs to more complex one. The computation starts from the state-space model of the DC motor having the block diagram in Fig. 16 that is in the form of the state equation (13) and the output equation (14).

$$u_i = A.x + b.u + e.z = \begin{bmatrix} 0 & \dfrac{K_m}{T_m.K_a} \\ -\dfrac{K_a}{K_m.T_a} & -\dfrac{1}{T_a} \end{bmatrix} x + \begin{bmatrix} 0 \\ \dfrac{K_T.K_a}{T_a} \end{bmatrix} u + \begin{bmatrix} -\dfrac{K_m^2}{T_m.K_a} \\ 0 \end{bmatrix} z \qquad (13)$$

$$y = c^T x = \begin{bmatrix} 1 & 0 \end{bmatrix} x \qquad (14)$$

where A is system matrix, x – state vector, b – input vector, c^T - output vector e- disturbance vector, u – input variable, y - output vector. The motor parameters correspond to the Fig. 16. The control structure with the feedback through the state controller - vector $\boldsymbol{r}^T$ is shown in Fig. 22 (the integrator at the input serves to reject constant or slowly changing disturbances what is a common case).

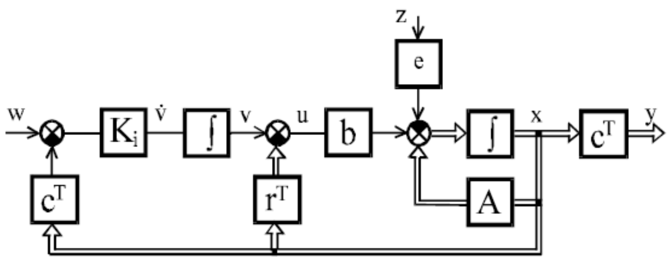


Fig. 22. Control structure with the state-space controller

The control structure is described in the state-space by the known state equation:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} A - br^T & b \\ -K_i.c^T & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ K_i \end{bmatrix} w + \begin{bmatrix} e \\ 0 \end{bmatrix} z \qquad (15)$$

The core of the virtual model consists in numerical calculation of the parameters K_i , $r_{1,}$ r_2 by the pole placement method: for a prescribed position of poles (usually a triple negative root giving the fastest system response on the aperiodicity boundary) the required polynomial is compared with the system polynomial and missing parameters of the controller are calculated from a set of linear algebraic equations.

The control structure in Simulink to simulate time responses is shown in (Fig. 23).

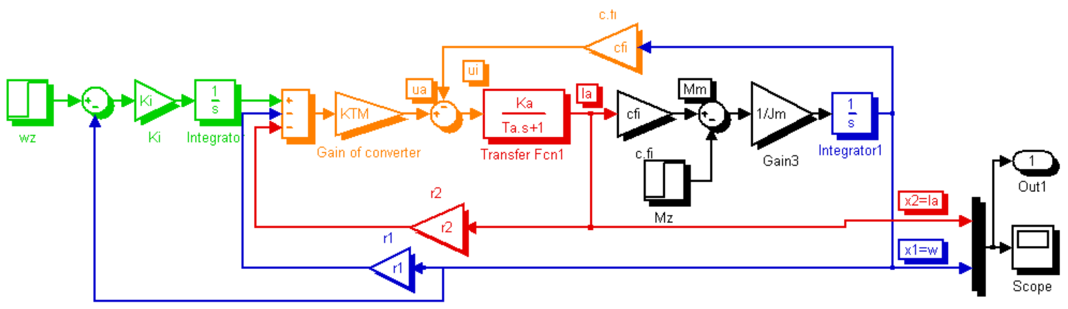
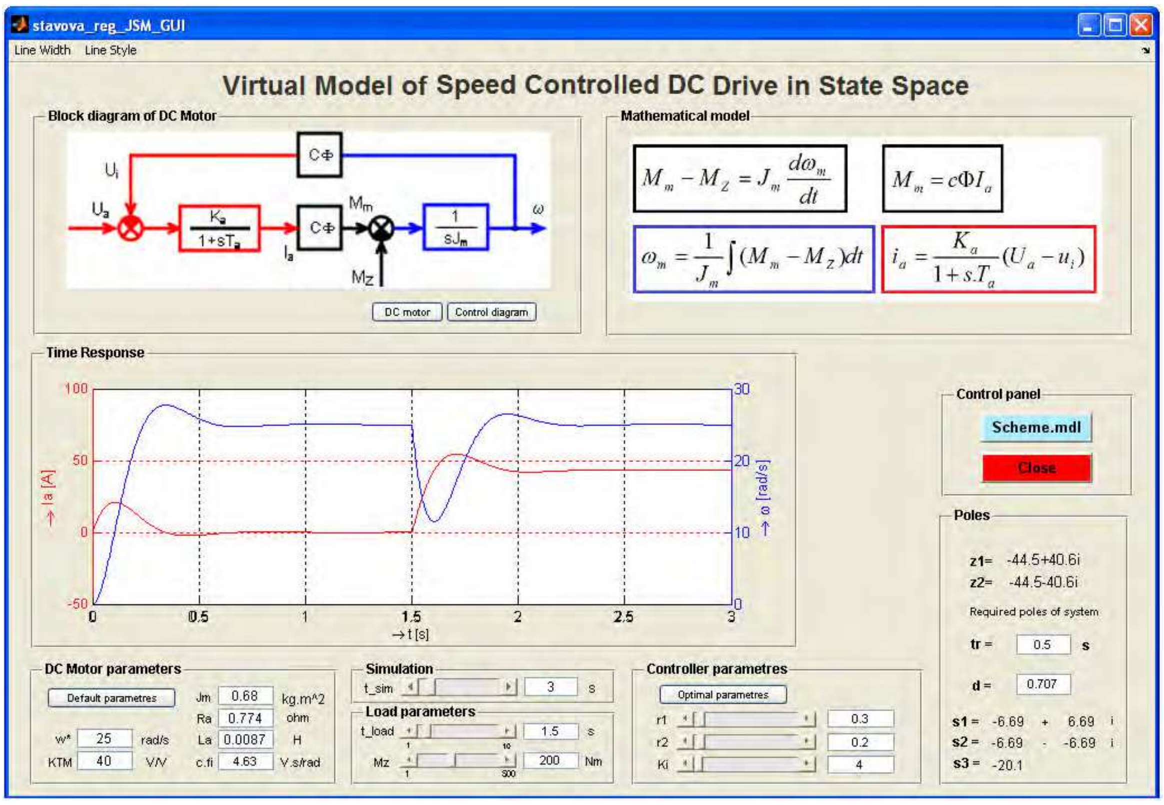


Fig. 23. Simulink model of the state-space control of DC drive

Fig. 24 shows the virtual model that enables to calculate state-space controller parameters and visualize time responses of the current and speed.



$$M_m - M_Z = J_m \frac{d\omega_m}{dt} \qquad M_m = c\Phi I_a$$

$$\omega_m = \frac{1}{J_m}\int (M_m - M_Z)dt \qquad i_a = \frac{K_a}{1+s.T_a}(U_a - u_i)$$

Jm	0.68	kg.m^2
Ra	0.774	ohm
La	0.0087	H
c.fi	4.63	V.s/rad
w*	25	rad/s
KTM	40	V/V
t_sim	3	s
t_load	1.5	s
Mz	200	Nm
r1	0.3	
r2	0.2	
Ki	4	
tr =	0.5	s
d =	0.707	

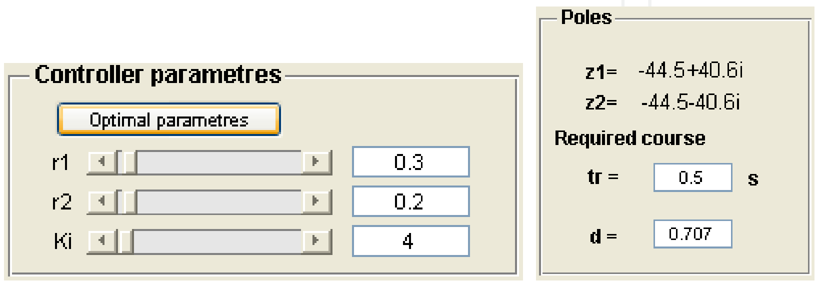
z1= -44.5+40.6i
z2= -44.5-40.6i
s1 = -6.69 + 6.69 i
s2 = -6.69 - -6.69 i
s3 = -20.1

Fig. 24. Virtual model of state-space controlled DC drive

The panel *Controller parameters* (Fig. 25a) serves to setting parameters of the state controller – by tuning or selecting the button $\boxed{\text{Optimal parameters}}$ to calculate poles position placement. Here:

r₁ – feedback from state variable x_1 (motor speed),
r₂ – feedback from state variable x_2 (motor current),
K$_i$ – gain of the integrator (to reject steady-state disturbances).

The state controller parameters are calculated automatically (Fig. 25b) on basis of required values of control time and damping (panel *Poles*, the item *Required course*). In the upper part of the panel Poles real positions of poles are shown.



r1	0.3
r2	0.2
Ki	4

z1= -44.5+40.6i
z2= -44.5-40.6i
tr = 0.5 s
d = 0.707

a) b)

Fig. 25. Panels: a) to set up controller parameters, b) to see position of real and required poles

4. Experiences with virtual models

Except of presented virtual models a series of further ones was developed to suit institutional needs in mechatronic systems teaching. The virtual modules are collected in the Virtual Laboratory of Mechatronic Systems that is available on the website (KEM TU Kosice, 2010).

A set of tens virtual models from all fields of mechatronics are used by the departmental staff as e-learning support in teaching of various subjects (just list several of them: Mechatronic Systems Modeling, Control Theory, Electrical Actuators and Drives, Servodrives, Motion Control, Control of Robots, Mechatronics of Production Systems, and others).

In order the students would get more skills and practical experiences we have divided the student laboratory work into three phases:

1. *Design and simulation* – for a given plant the student has:
   – to derive the mathematical model,
   – to compose the block diagram,
   – to design control,
   – to verify system behavior by simulation.
2. *Verification and analysis* – for a designed control algorithm the student has:
   – to verify the design using a virtual reference model and
   – to perform system analysis (small experiments round a working point) in order to get the system responses (in time and frequency domains) and to investigate system behavior at various values of system parameters and in various working points.
3. *Final verification* of the design on the laboratory model (by programming the embedded control system).

The first two phases are performed outside of the laboratory. The student has to his/her disposal guidelines and reference virtual models that are available through internet for 24/7 hours. Within the institution network they run online (due to limited SW licenses) and outside of the institution the student has to download them and run on own computer (having installed MATLAB).

The reference virtual models also facilitate the teacher review of the student projects – the teacher does not need to check in details the simulating diagrams and search for eventual errors. On other side, the student has a model (i.e. a template) to verify whether he derived and designed a proper solution (e.g. proper parameters of controllers).

Let's note that the third phase of his projects is usually done in the laboratory on the laboratory model under teacher's supervision - for complex systems we prefer the presence of the students in the lab and consultations with the teacher.

The virtual models are also used in lectures to describe and explain principles of complex system behavior at various values of parameters.

5. Conclusion

The chapter describes principles of virtual models development of electromechanical systems that support eLearning in field of mechatronics. The developed virtual models enable to perform analysis of the real system in various working points and to observe influence of the parameters changes to system behavior. The shown virtual models are of various complexity – the simpler ones enable virtual analysis of system and complex virtual models deals also with synthesis of system controllers. Using them, the students can compare and verify their own design performed on basis of design rules application.

Development procedure and features of the virtual models are presented on several typical examples. Here, MATLAB/Simulink software was chosen as one of the best and widespread programming tools for development of virtual models that enables relatively simply programming of even complex systems. Based on the procedure a whole series of virtual models (more than 30) designed in GUI MATLAB was developed at the author's institution in recent years. They are accessible trough internet – through the website of the Virtual Laboratory for Control of Mechatronic Systems (KEM TU Kosice, 2010).

A big advantage of developed virtual models consists in the fact the students do not need to know the complexity of dynamical system which simulation scheme is working in the background of; they change only system parameters on the screen – e.g. mechanical parameters in case of mechanical subsystems (moments of inertia, constants of the elasticity and damping of the flexible joints), parameters of electrical systems (resistance, capacitance, inductance, gains, …), etc., select form and input signals (shape and amplitude of forcing and load signals) and select mode of calculation and outputs (graphs displaying). The parameters of virtual models can be changed by a slider or inputting numerical values into editable boxes.

A shortcoming of developed models consists in the fact they run on computers having installed the MATLAB program. To overcome this difficulty our future work will be concentrated to application of the browser plugins, which are necessary to run MATLAB application on computers not having installed the MATLAB program, e.g. by using VCLab plugin (Ruhr-Universität Bochum, 2011). It must be noted that this application enables to run MATLAB operation without simulation (i.e. without a Simulink scheme), without 3D virtual reality views and without animation, so the advantage on one side will complicate development of the virtual modules by application more mathematical subroutines. But this restriction does not make any serious problem and can be easily solved by application several subroutines.

The developed virtual models are in intensive use and they complete basic e-learning support in various subjects from field of mechatronics. They also serve at lecturing basic subjects from fields of modeling mechanical subsystems, electrical drive systems, system control, and various electrical systems up to mechatronic ones.

6. Acknowledgement

This work was supported by Slovak Cultural and Educational Agency of the Ministry of Education of Slovak Republic under the contract KEGA 103-039 TUKE-4/2010 "Students' Skills Development for Mechatronic Systems Control".

7. References

Bauer, P.; Fedák, V. & Rompelman O. (2008). PEMCWebLab - Distance and Virtual Laboratories in Electrical Engineering – Development and Trends, *Proceedings of*

Power Electronics and Motion Control International Conference, *EPE-PEMC 2008,* ISBN 978-1-4244-1742-1, Poznaň, September 1-3, 2008

Bauer P.; Fedák V.; Hájek V. & Lampropoulos, I. (2008). Survey of Distance Laboratories in Power Electronics, *Proceedings of IEEE 39th Annual Power Electronics Specialists Conference*, ISBN 978-1-4244-1667-7, Rhodes, Greece June 15-19, 2008

Babich A. & Mavrommatis K. (2004). Virtual Laboratory Concept for Engineering Education, *Proceedings of International Conference on Engineering Education and Research "Progress Through Partnership"*, ISSN 1562-3580, VSB-TUO, Ostrava, 2004

Bianchi R. A. C. & La Neve, A. (1999). Distance Learning through a Robotic Virtual Lab, Date of access: July 31, 2011, Available from <http://fei.edu.br/~rbianchi/publications/icee1999-robot.pdf>

Cheng, K. W. E.; Chan, C. L.; Cheung, N. C. & Sutanto, D. (2004). Virtual laboratory development for teaching power electronics, *Proceedings of Power Electronics and Motion Control International Conference, EPE-PEMC 2004*, ISBN 9984-32-010-3, Riga, Latvia. September 2-4, 2004.

Cheng, M. H. M. & Chiu G. T. C. (2010). A mechatronic approach to a virtual laboratory service on internet, *International Journal of Virtual Technology and Multimedia,* Vol. 1, No. 2, 2010, pp. 140 – 154, ISSN 1741-1874

Coito F.; Gomes L. & Costa A. (2007). Simulation, Emulation and Remote Experiments, *Proceedings of the Workshop on using VR in Education*, pp. 99-110, ISBN 978-989-20-0715-1, Lisboa, March 2007,

Ďurovský, F. & Fedák, V. (2010). Integrated Mechatronic Systems Laboratory, *Proceedings of the 14th Power Electronics and Motion Control International Conference, EPE-EPMC 2010*, ISBN 978-1-4244-7854-5, Ohrid, Macedonia. September 6-8, 2010.

Ernest, E.; Sztylka, R.; Ufnalski, B. & Koczara W. (2006). Methods in Teaching Modern AC Drives: Inverter-fed Motor System with Internet-based Remote Control Panel, *12th International Power Electronics and Motion Control Conference*, *EPE-PEMC 2006,* pp. 2130 – 2133, ISBN 1-4244-0121-6, Portorož, August 30 – September 1, 2006

Faculty of Electrical Engineering in Belgrade (n.d.). Virtual Laboratory for Distance Learning in Robotics and Mechatronics. The idea of distance learning. (n.d.). Date of access: July 31, 2011, Available from <http://robot.etf.rs/ index.php/researchprojects/virtual-laboratory-for-distance-learning-in-robotics-and-mechatronics/>

Fedák, V.; Fetyko, J. & Repiščák, M. (2005). Computer Supported Education for Industrial Mechatronic Systems, *Proceedings of Computer Based Learning in Science International Conference, CBLIS 2005,* pp. 19-27, ISBN 9963-607-63-2, Žilina, July 2-6, 2005

Fedák, V.; Balogh, T.; Bauer, P. & Jusko S. (2008). Virtual and Remote Experimentation in Motion Control, *Proceedings of the 11th International Conference on Mechatronics*, ISBN 978-80-8075-305-4, AD University of Trencin, Trenčianske Teplice, June 4-6, 2008

Fedák, V.; Hric, M. & Ďurovský, F. (2010). Laboratory Model of Continuous Line with Multi-Motor Drive, *Proceedings of the XXIV. Int. Scientific Conference microCAD 2010,* *Section K: Electrotechnics and Electronics,* ISBN 978-963-661-925-1, University of Miskolc, March 18-20, 2010

Galitz, W. O. (2007). *The Essential Guide to User Interface Design. An Introduction to GUI Design Principles and Techniques,* Wiley Publishing, Inc., ISBN 978-0-470-05342-3, Indianapolis, Indiana

Georgia Institute of Technology (n.d.), *Educational Matlab GUIs,* Date of access: July 31, 2011, Available from <http://users.ece.gatech.edu/mcclella/ matlabGUIs/>

Hercog, R. D.; Jezernik, K. (2007). Advanced control course with teleoperation in the mechatronics study, *Proceedings of 16th Int. Conf. on Electrical Drives and Power Electronics*, EDPE 2007, The High Tatras, September 24-26, 2007

KEM TU Kosice (2010). Virtual Laboratory of Mechatronic Systems Control, In: *web site of the project KEGA, No 3/4203/06*, Date of access: July 31, 2011, Available from < http://andromeda.fei.tuke.sk/> (in Slovak)

MeRLab (n.d.). Innovative Remote Laboratory in the Etraining of Mechatronics, Date of access: July 31, 2011, Available from <www.merlab.eu>

Pipan, M.; Arh, T.; Blažič, B. J. (2008). Innovative Remote Laboratory in the Enhanced E-training of Mechatronics, *Proceedings of the 7th WSEAS International Conference on Circuits, Systems, Electronics, Control and Signal Processing* (CSECS'08), Puerto De La Cruz, Canary Islands, Spain, December 15-17, 2008

Petropol-Serb, G.D.; Petropol-Serb, I.; Campeanu, A. & Petrisor, A. (2007). Using GUI of Matlab to create a virtual laboratory to study an induction machine. EUROCON, 2007. *The International Conference on Computer as a Tool,* ISBN 978-1-4244-0813-9, Warsaw, September 9-12, 2007

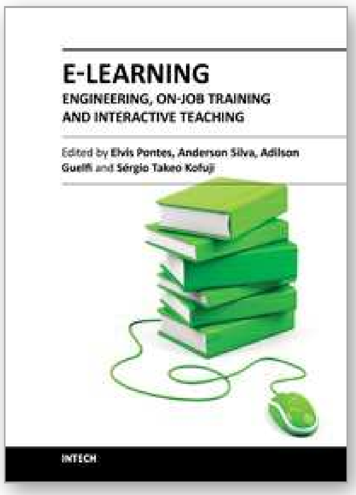
Potkonjak, V.; Vukobratović, M.; Jovanović, K. & Medenica, M. (2010). Virtual Mechatronic/Robotic laboratory - A step further in distance learning, *Computers & Education archive,* Vol. 55 , Issue 2 (Sept. 2010), pp. 465-475, ISSN 0360-1315

Ruhr-Universität Bochum (2011), Lehrstuhl für Automatisierungstechnik und Prozessinformatik. Virtual Control Lab 3.1 MATLAB Plugin. Date of access: October 31, 2011, Available from <http://www.atp.ruhr-uni-bochum.de/VCLab/ software/MatlabPlugin/MatlabPlugin.html/>

Saadat, H. (n.d.). MATLAB Graphical User Interface for EE Students. Date of access: July 31, 2011, Available from <http://people.msoe.edu/ ~saadat/matlabgui.htm>

Technical University of Kosice (n.d.). Students' Skills Development for Mechatronic Systems Control. In: *Project KEGA 103-039TUKE-4/2010*

Wong, H.; Kapila, V. & Tzes A. (2001). Mechatronics/Process Control Remote Laboratory. *Proceedings of the 2001 American Society for Engineering Education, Annual Conference & Exposition*, Albuquerque, NM, 2001.


E-LEARNING
ENGINEERING, ON-JOB TRAINING AND INTERACTIVE TEACHING
Edited by Elvis Pontes, Anderson Silva, Adilson Guelfi and Sérgio Takeo Kofuji
INTECH

E-Learning - Engineering, On-Job Training and Interactive Teaching
Edited by Dr. Sergio Kofuji

ISBN 978-953-51-0283-0
Hard cover, 238 pages
Publisher InTech
Published online 14, March, 2012
Published in print edition March, 2012