Modeling of linear dynamical systems using open tools

Zoltán Magyar, Tomáš Starý, Ladislav Szolik, Ľudovít Vörös, Katarína Žáková
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Ilkovičova 3, 812 19 Bratislava, Slovakia
katarina.zakova@stuba.sk

Abstract

The aim of the article is to present a new possible way of an online interactive teaching. The effort is to substitute proprietary software that is commonly used in educational process by open technologies. Since the subject of this article is oriented to the automation area we devoted our attention to open software such as SciLab, OpenModelica and Maxima. Moreover, these tools can be available not only locally on the desktop computer or the notebook but also online via Internet. In the paper we also introduce examples of the open tools implementation.

1. Introduction

The Control Theory is one of the fundamental subjects that are taught at universities of technical orientation. At the Faculty of Electrical Engineering and Information Technology (Slovak University of Technology) in Bratislava it is placed into the second year of the study.
For the control design it is really important to find the exact possible mathematical model of a system. Mathematical models can be described by different forms. Depending on the particular system and the particular circumstances, one mathematical model can be better than another one. For example, in optimal control problems, it is advantageous to use state-space representations. On the other hand, for the transient response or frequency response analysis of single input single output, linear, time-invariant systems, the transfer function representation may be more convenient than any other [7]. Another possibility is to use the differential equation description. Once a mathematical model of a system is obtained, various analytical and computer tools can be used for analysis and synthesis purposes.

2. Open technologies

In the last decade the development of the low cost software (free and open source software) takes increasingly bigger and bigger importance. It brings competition to the proprietary software products including open source platforms. Open standards are also supported by European Union that would like to create conditions for the market development and to avoid the constraints given in the case when the only exclusive producer creates software for certain area. The direction was also supported by the speech of European commissioner Neelie Kroes with title "Being open about standards" from June 10, 2008 where she recommends to use the software based on open standards. This recommendation was headed to governments of all European countries.
This paper also tries to follow this tendency and therefore it is oriented to open technologies that can be used for the topic that is included in the basic Control Theory subject – Modeling of linear dynamical systems.

3. Open tools implementation

In this section we devote our attention to the selected open tools that can be used as keystones at the building of virtual laboratory in the area of automation, cybernetics, mechatronics or industrial informatics.

Maxima

Maxima is the software environment that enables to accomplish various mathematical operations including symbolic calculations. In this way it can be an alternative for such programs as Maple or Mathematica are.
To include it to our virtual laboratory it was necessary to build an interface that would enable to approach the program kernel. Actually, we decided to implement two interfaces:

1. graphical user interface that offers users to use all Maxima functions via the command line available on the web page (Fig.3).
2. program interface that enables to exploit the Maxima functionality in other web applications.

Graphical user interface
The graphical user interface is supported by other complementary technologies such as JQuery, JSMath and Yetii.
Users can use the application of Maxima installed on the server by typing in Maxima commands into the command line/area displayed on the webpage. After submitting the form browser sends the commands to the server side PHP script (web-interface.php). This script validates the data

and forwards them to another PHP script (php-MAXIMA.php), that is able to send data to Maxima. Before sending it decides whether the data represent the graph plotting command or a single equation command. If the command is the graph plotting command it generates the filename for the image file which is going to carry the image of the plotted graph. After that decision the script creates the shell command string containing Maxima commands and executes it via shell_exec() PHP function.
The system shell on the server gets the command and executes it by starting the installed version of Maxima (Fig.1).

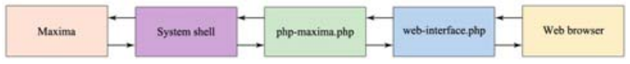
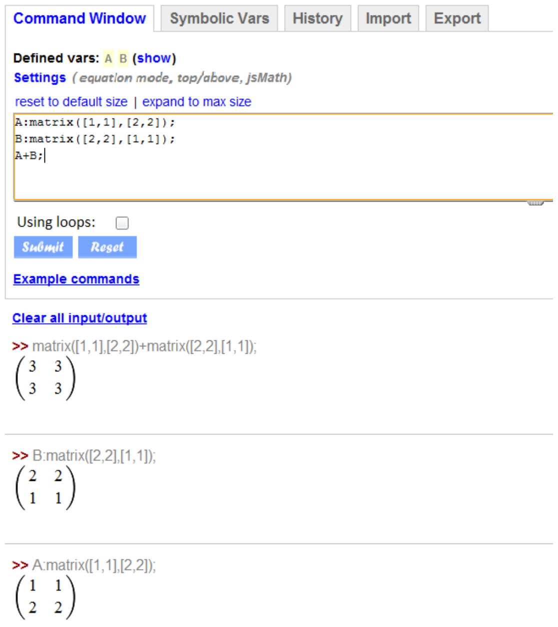


Fig.1. Schematic block scheme of Maxima online implementation

Maxima gets the batch string and executes it. If the batch string contains graph plotting commands it generates the necessary output and call GNUPLOT (for 2D and 3D plotting) or IMAGEMAGICK (for fractal plotting) applications to generate PNG image file from Maxima output with the already generated filename. After that Maxima sends its output as the text string back to the shell and the shell forwards it to the PHP script (php-MAXIMA.php).
The PHP script receives the data as the string from the shell_exec() function. The script has to remove all unnecessary characters and then the user input and the Maxima output are formatted into the TeX format and sent to other PHP script (web-interface.php) which generates the web page with the Maxima output.
Since the user can specify two formats of the displayed mathematical results (JSMath or PNG images) the web-interface.php script has to distinguish between both outputs. If the mathematical results should be displayed by JSMath tool the output is generated with the TeX representation of received strings and JSMath will take care about all the remaining job. Otherwise the PNG images of results are generated using MATHTEX and DVIPNG tools. Then, all results are displayed in the user web browser. The detailed block scheme of the Maxima online implementation is shown in Fig. 2. The described graphical user interface is illustrated in Fig. 3.



Defined vars: A B (show)
Settings (equation mode, top/above, jsMath)
reset to default size | expand to max size

```
A:matrix([1,1],[2,2]);
B:matrix([2,2],[1,1]);
A+B;
```

Using loops:

>> matrix([1,1],[2,2])+matrix([2,2],[1,1]);

$$\begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix}$$

>> B:matrix([2,2],[1,1]);

$$\begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix}$$

>> A:matrix([1,1],[2,2]);

$$\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$$

Fig.3. Command line (area) for Maxima commands

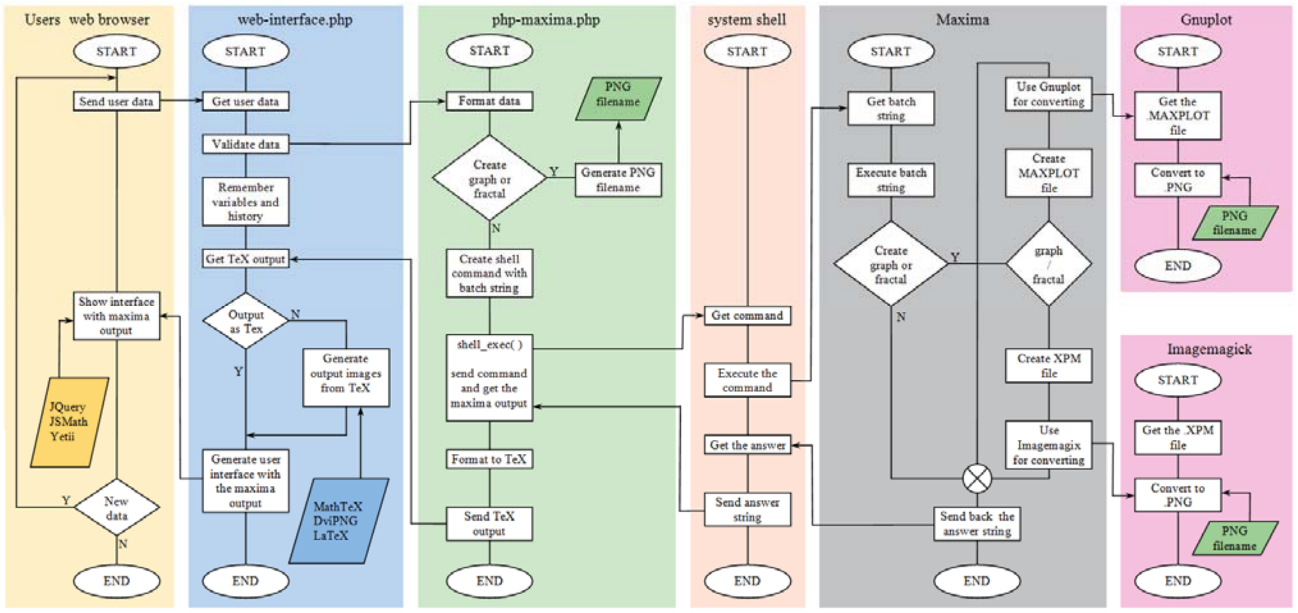
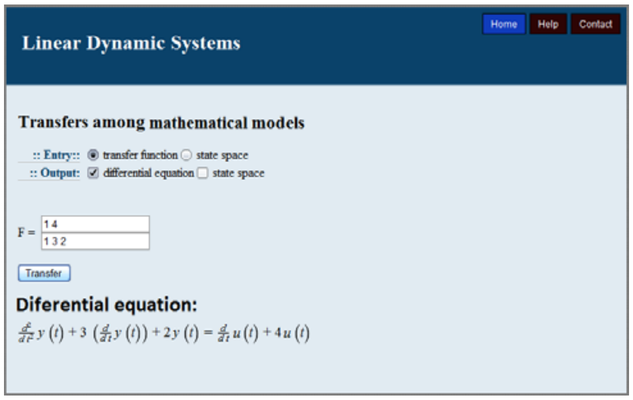


Fig.2. The detailed block scheme of the Maxima implementation on the server

Program interface

The program interface uses the same steps as they were described above except of the user interaction via web page. The mathematical tasks for Maxima solving are indicated not by the user but by other web application. Similarly, the results from Maxima are sent not to the user but they are sent back to the application that created the operation request. Actually, the user doesn't need to know about the existence of Maxima environment on the server since it is used only for mathematical computations. The program interface includes the created methods that enable interaction between Maxima and a new web application. The example of such application is shown in Fig. 4. It enables to transform one form of the dynamical system description to the other one.



Linear Dynamic Systems

Transfers among mathematical models

:: Entry:: [✓ transfer function] state space

:: Output: [✓ differential equation] state space

F = 1 4 / 1 3 2

Diferential equation:

$$\frac{d^2}{dt^2}y(t)+3\left(\frac{d}{dt}y(t)\right)+2y(t)=\frac{d}{dt}u(t)+4u(t)$$

Fig.4. Front-end of the application for model description conversion

OpenModelica

OpenModelica enables to model and simulate the behavior of the dynamical systems. In this way it is similar to Maple or Matlab.

The basic block scheme of its implementation on the server is shown in Fig.5.

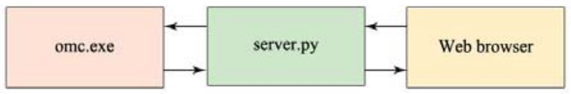


Fig.5. Schematic block scheme of OpenModelica online implementation

The communication with the server is provided via the web browser form where the user enters all necessary parameters that are sent to the OpenModelica engine installed on the server. The same web page can be also used for the result visualization (see Fig.7).

After sending data to the server, they have to be transformed to the form that can be understood by OpenModelica. Therefore the data are transformed to the string representation. For this purpose we used the Python programming language.

The Python script language was chosen because of its ability to cooperate with CORBA interface that can be used for communication with OpenModelica environment. We can run OpenModelica in interactive corba mode. This mode generates an Interoperable Object Reference (IOR) number as the reference to CORBA.

After OpenModelica receives strings generated by Python, it executes included commands and sends the result back to Python.

In the case of simulation (the model has to be pre-defined on the server) OpenModelica sends back only report about the successful simulation. The simulation data are written to the external file. Running the simulation, there is no difference between running it in the local and the interactive mode, since in the case of the local mode the data are also stored in an external file.

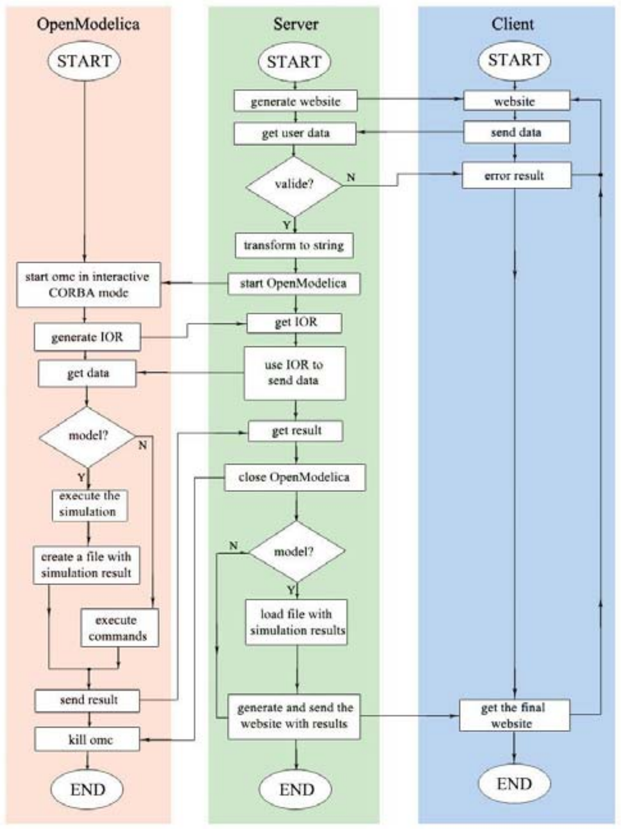


Fig.6. The detailed block scheme of the OpenModelica implementation on the server

At the simulation the Python script must also perform other tasks:

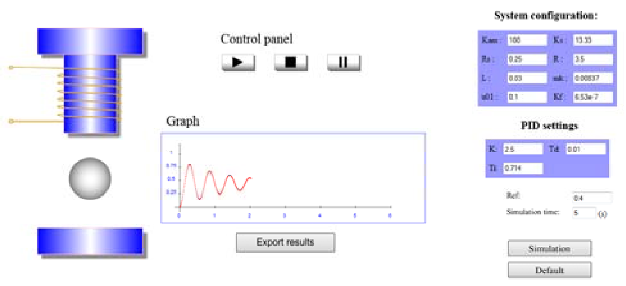
- to open the file with the simulation results. The filename and its location have to be defined in the configuration file.
- to load values from the external file. The file can contain unnecessary data that have to be filtered out by the Python script.

- to render graphics objects. The results are presented to user in the form of the graph and/or animation.

After the results are visualized on the web page the communication between the server and the client side is concluded. Later, the client can send new data that the server has to process again.

The detailed communication among all components of this client/server service is described in Fig.6.

In Fig.7 one can find the application with the magnetic levitation experiment. The task is to control the position of the ball between two coils. The control is ensured by means of PID controller.



Kam :	100	Ks :	13.33
Rs :	0.25	R :	3.5
L :	0.03	mk :	0.00837
a01 :	0.1	Kf :	6.53e-7

K:	2.6	Td:	0.01
Ti:	0.714		

Ref: 0.4
Simulation time: 5 (s)

Fig.7. Front-end of the magnetic levitation experiment

SciLab

SciLab also belongs to the software that enables to simulate dynamical systems. In its nature it reminds Matlab mainly because of its graphical interface Scicos that is very similar to Simulink.

The principle of the communication between the application SciLab and the server web application (Fig.8) is quite simple. It is based on the method of TCP communication. SciLab contains a package named TCP Socket Toolbox, which allows connecting on a listening TCP socket and to send and receive data via this socket. The only disadvantage of this toolbox is that it doesn't allow creating a listening socket. It only allows to connect to a socket that already has the status of "listening". However, this fact doesn't introduce any problem, because the most of the server side scripting languages, like PHP, supports packages, toolboxes or extensions allowing a developer to gain full control (including creation and destruction) of TCP connections.

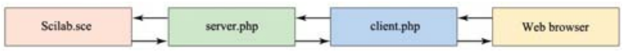


Fig.8. Schematic block scheme of SciLab online implementation

The whole communication runs in such a way that the user enters the input parameters to the form on the webpage. Then he or she submits parameters to the server, i.e. the request for processing is accomplished. The PHP script client.php formats the input of the user into SciLab instructions and sends it via session variable to the script server.php. The main task of this script is to create, maintain and close the communication with SciLab. The script server.php sends the requested instructions as a string to SciLab. Finally, the script SciLab.sce executes the received instructions and sends back the result also in a string form. The script client.php handles the result and sends the generated webpage with the requested information to the web browser of the end user.

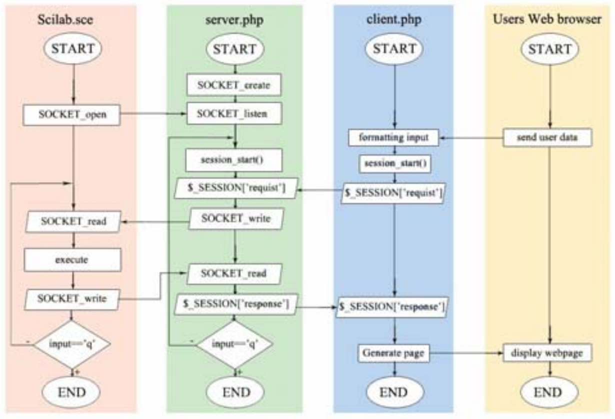
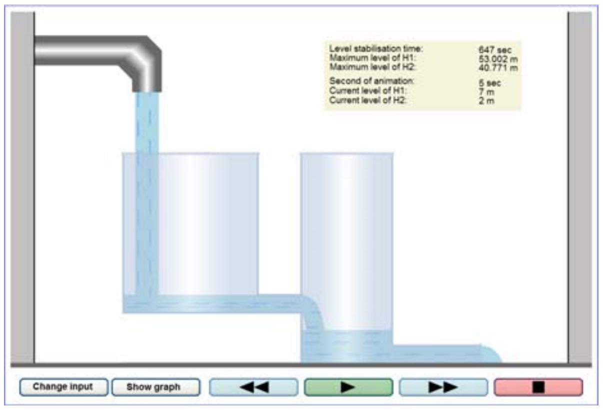


Fig.9. The detailed block scheme of the SciLab implementation on the server

The Fig.10 illustrates the next example that was considered for modeling and simulation. It is a two tank system where it is necessary to control the level of liquid in the first or the second tank. The back-end of the application was realised using the SciLab environment.



Level stabilisation time:	647 sec
Maximum level of H1:	53.002 m
Maximum level of H2:	40.771 m
Second of animation:	5 sec
Current level of H1:	7 m
Current level of H2:	2 m

Fig.10. Front-end of the two tank system experiment

4. Server

All introduced applications run on Linux server. Of course, all of them can be executed by various access rights allocated to anonymous user, student, teacher and administrator as well. However, all these application enable to access in more or less visible ways certain

commands that can damage the application or even server installation (e.g. hard disk formatting).
Since the whole application runs on Linux server, we can predict, that this system is able to provide the requested stability and security to keep usernames and passwords safe and is more avoidable against virus, malware, spyware and trojan attacks. However, it is also necessary to ensure the protection of applications against the already mentioned „self-destruction“ (mostly done by a user). It can be done by several ways. We used the Jailkit that enables to lock the folder that is used by the application. Then, it is impossible to reach any of crucial files or folders of the Linux system and it also is impossible to delete any of files that after login to the system belongs to some other user.

5. Conclusions

The paper presented an alternative way to the building of virtual laboratories. Our effort is to use the presented tools in two ways:

- to offer interested users (mainly students) the command line access to the presented software in order they could use the software online without the necessity of the installation on their own computer.
- to enable to use the engine of the mentioned software in frame of other web applications to facilitate their functionality and design.

We hope that we will be in our direction successful.

6. Acknowledgments

The work has been partially supported by Slovak Grant Agency, Grant KEGA No. 3/7245/09 and Grant VEGA No. 1/0656/09. This support is very gratefully acknowledged.
Authors would also like to thank to David Gyuras for his help with Maxima interfaces.

7. References

[1] A. Beshenov, Maxima beginners FAQ, http://cadadr.org/maxima/faq.html

[2] D. Gyurasz, Symbolické výpočty na Internete, Diploma thesis, FEI STU Bratislava, 2009 (in Slovak).

[3] S. Holzner, *Mistrovství v AJAXu*, Computer Press. Brno, 2007.

[4] M. Huba, P. Bisták, M. Fikar, M. Kamenský, ”Blended Learning Course "Constrained PID Control"”, 7*th IFAC Symposium on Advances in Control Education* ACE’06, Madrid, Spain, 2006.

[5] P. Lutus, Symbolic Mathematics Using Maxima, http://arachnoid.com/maxima/

[6] R. Nikoukhah, Scicos presentation, http://www.scicos.org

[7] K. Ogata, *Modern Control Engineering*, 3rd Edition, Prentice Hall London, 1997.

[8] OmniORBpy documentation, http://omniorb.sourceforge.net/docs.html

[9] A. Pop, P. Frizon, OpenModelica Users Guide, http://www.ida.liu.se/~pelab/modelica/OpenModelica

[10] A. Pop, P. Frizon, OpenModelica. http://www.ida.liu.se/~pelab/modelica/OpenModelica

[11] Python documentation, http://www.python.org/doc/

[12] The PHP Group. 2001-2009. PHP Documentation. http://www.php.net/docs.php

[13] R. H. Rand, Introduction to Maxima, http://maxima.sourceforge.net/docs/intromax/intromax.html

[14] J. Resig and the jQuery Team. 2009. jQuery Documentation. http://docs.jquery.com

[15] T. Reveyrand, The SOCKET Toolbox for Scilab, http://www.reveyrand.fr/

[16] F. Schauer, M. Ožvoldová, F. Lustig, „Real Remote Physics Experiments across Internet – Inherent Part of Integrated E-Learning” , *Int. Journal of Online Engineering* (iJOE), 4, No 2, 2008.

[17] Chr. Schmid, „Internet - basiertes Lernen“, *Automatisierungstechnik*, 51, No. 11, p. 485-493, 2003.

[18] Scilab consortium. Scilab, http://www.scilab.org/

[19] M. Šimunek, P., Bisták, M. Huba, „Virtual Laboratory for Control of Real Systems”, *Conference Proceedings* ICETA, Košice, Slovakia, 2005.

[20] A. J. Turgeon, "Implications of Web-Based Technology for Engaging Students in a Learning Society", *Journal of Public Service and Outreach* 2(2): 32-37.

[21] E. L. Woollett, Maxima by example, http://www.csulb.edu/~woollett/

[22] K. Žáková, M. Janotík, "Mathematical Modeling of Dynamic Systems: an interactive online lesson", 5*th international conference “Virtual University”*, Bratislava, December, 2004.

[23] K. Žáková, " Control Theory - an interactive online course", 6*th international conference “Virtual University”*, Bratislava, December, 2005.

[24] K. Žáková, M. Sedlák, “Remote Control of Experiments via Matlab”, *Int. Journal of Online Engineering* (iJOE), 2, No. 3, 2006.