ANNALS OF FACULTY ENGINEERING HUNEDOARA
– INTERNATIONAL JOURNAL OF ENGINEERING
Tome X (Year 2012) – FASCICULE 3 (ISSN 1584 – 2673)

[1-2.] Tadej TAŠNER, [2.] Darko LOVREC, [3.] Frančišek TAŠNER, [4.] Jörg EDLER

COMPARISON OF LabVIEW AND MATLAB FOR SCIENTIFIC RESEARCH

[1]HAWE HIDRAVLIKA D.O.O., PETROVČE 225, PETROVČE, SLOVENIA
[2-3]UNIVERSITY OF MARIBOR, FACULTY OF MECHANICAL ENGINEERING, SMETANOVA ULICA 17, MARIBOR, SLOVENIA
[4]GRAZ UNIVERSITY OF TECHNOLOGY, INSTITUTE OF PRODUCTION ENGINEERING, KOPERNIKUSGASSE 24, GRAZ, AUSTRIA

ABSTRACT: *Nowadays engineers, researchers and scientists use advanced software suites for their work in the field of signal analysis and simulation of dynamic systems. Among the most commonly used tools are LabVIEW and MATLAB. Therefore a question arises – which one to choose? In this paper both tools are briefly presented, followed by a comparison based on practical examples. Featuring four comparisons – computation with matrices, FFT calculation, Bode plot and DC motor simulation.*
KEYWORDS: *LabVIEW, MATLAB, Simulink, comparison, simulation*

INTRODUCTION

*LabVIEW and MATLAB were developed by two different American companies. The first is developed by National Instruments and the second by MathWorks. LabVIEW. LabVIEW (**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) is a graphical programing environment based on graphical programming language G. MATLAB (**MAT**rix **LAB**oratory) is the name used for the programming language and programing environment specialized for numerical calculations. Both platforms help engineers and scientists around the world in various stages of design, modeling, simulation, prototype testing, or deployment of new technologies.*

Scientists mainly use MATLAB for their simulations, especially due to plenty of additional libraries and the Simulink add-on. Libraries contain specific higher-level functions of a particular field. Such functions speed up the development of advanced applications. As soon as a functional and intuitive graphical user interface or interaction with hardware (signal acquisition and generation) is required instead of MATLAB, LabVIEW is mostly used. Some more experienced users also choose a combination of both tools, which communicate via either the API or DLL libraries. Data can be exported from one and imported into another program for further processing. There is also a possibility to implement MATLAB code in LabVIEW. Most of MATLAB functions can be integrated into LabVIEW by using MathScriptfunction module (available with »MathScript RT Module« add-on). [1] [2] [3] [4]

LabVIEW

The beginnings of LabVIEW back to the mid-eighties when Macintosh Company produced first computer with graphical user interface. Graphical user interface enabled visualization of flow charts on the computer screen, which inspired Jeff Kodosky for graphical programming. Since he was using mostly data acquisition for his work, he began creating a graphical programming language based on dataflow rather than sequential processing, which is most common for John von Neumann's computer architecture. In 1986 the first version of LabVIEW programing environment was issued (Fig. 1), which already featured programming by connecting the function blocks, as we know it today. The main objective of the programming environment was to simplify data acquisition from GPIB bus. Therefore, the very first function blocks enabled data acquisition, which is still a main function of LabVIEW today. [5]

Figure 1–LabVIEW 1.0 on Macintosh [6]

Today, LabVIEW programming environment is the leader in the field of computer based measurement and data acquisition. It features exceptional compatibility with National Instruments hardware and also other devices. Moreover it offers easy-to-use construction of graphical user interface. There are also plenty of add-ons which can be used to deploy code to standalone devices, analyze and process signals, control, simulation, system analysis, report creation and database connection.

MATLAB/Simulink

Mathematician and Professor Cleve Moler began to develop MATLAB in late seventies in order to simplify usage of libraries for numerical computation (Linpack, EISPACK), which were programmed

using FORTRAN programming language. The main advantages over the use of libraries within FORTRAN were simple input and output of data and arithmetic operations on vectors and matrices as a whole. The potential of MATLAB was later discovered by Jack Little, who founded the MathWorksCompany in 1984 along with Cleve Moler. The company first translated MATLAB from FORTRAN to C, updated the user interface, added M-files and libraries. [7]

Nowadays MATLAB is widespread across all scientific disciplines mainly due to a simple syntax, stability and a wide range of applicability. There is also a lot of additional libraries and add-ons from various areas such as control and identification of systems, neural networks, fuzzy logic, statistics, symbolic math… The most important add-on for engineers is certainly Simulink, which allows modeling, simulation and analysis of dynamic systems.

COMPARISON OF LabVIEW AND MATLAB

Below, LabVIEW and MATLAB/Simulink are compared in four different areas: calculation with matrices, fast Fourier transform (FFT), calculation with transfer functions along with Bode plot and simulation of DC motor control. All comparisons will be made on the same PC (4 core Intel Core i7-2600K processor running at 3.4 GHz with Hyper-Threading enabled and 8 GB of RAM) with MATLAB R2011a and LabVIEW 2010 installed.

Each comparison area will feature comparison of how simple and surveyable the program code is and the time needed to execute the program will also be measured. The simplicity of the code will be assessed according to the number of elements or lines used. The faster the code can be updated or debugged the better surveyable the program is. Time measurements will be carried out in ten repetitions, and then maximum, minimum and average values will be calculated.

CALCULATION WITH MATRICES

A simple algorithm was selected to test the computational speed. The algorithm multiplies 1000 matrices with a size of 1000x1000 (one million elements), which contain only ones (1) and their data-type is double precision floating-point. To multiply two matrices of size 1000x1000, computer must perform 10^9 multiplications and 10^9 additions. In total such algorithm performs $99 \cdot 10^9$ multiplications and $99 \cdot 10^9$ additions.

$$X = \begin{bmatrix} 1 & \cdots & 1 \\ \cdots & \cdots & \cdots \\ 1 & \cdots & 1 \end{bmatrix}^{100} = \begin{bmatrix} 10^{297} & \cdots & 10^{297} \\ \cdots & \cdots & \cdots \\ 10^{297} & \cdots & 10^{297} \end{bmatrix} \quad (1)$$

In order to make the algorithm more complex, a new matrix is created in each iteration. The code of such algorithm in LabVIEW is shown in Figure 2 and in MATLAB the code is as follows:

```
tic;
result=ones(1000,1000,'double');
for i=1:99
result=result*ones(1000,1000,'double');
end
time=toc;
```

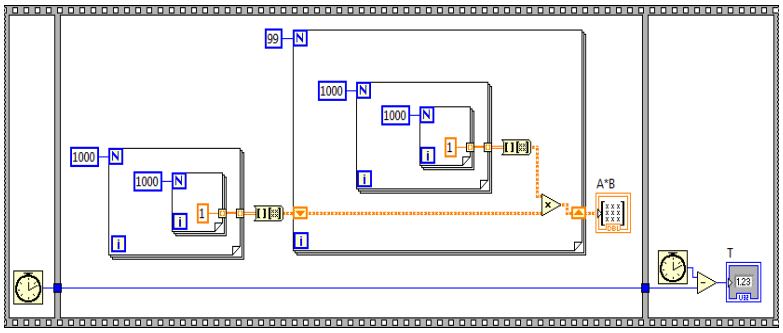

Figure 2–Program code for matrix multiplication in LabVIEW

To implement both algorithms only the basic package is needed (no add-ons required). The results of the comparison of calculation with matrices are shown in Table 1. The results show that MATLAB is nearly three times faster than LabVIEW. Moreover code in LabVIEW looks very complicated and unclear. For matrix computations MATLAB is the way to go.

Table 1. Comparison of calculation with matrices

programming environment	execution time [s]			code	
	average	minimum	maximum	simplicity	surveyability
MATLAB	6,23	6,052	6,349	+	+
LabVIEW	15,618	15,372	15,807		

FAST FOURIER TRANSFORM (FFT)

For their work engineers frequently use fast Fourier transform, which is used for spectral analysis of either, generated or acquired signals. For the test a signal with length of 10 s and 1.000.000 samples is generated. The sample signal is a sum of two sine waves with frequencies of 5 and 50 Hz. After the signal is generated, FFT is performed. Generated signal, magnitude and amplitude plots are shown on the screen.

Program code to implement the above algorithm in LabVIEW is shown in Figure 3 and the program code in MATLAB is written as follows:

```
tic;

fs = 1e3; %frekvencavzorčenja [Hz]
len = 1e6; %številoodtipkov [-]
Ts=1/fs; %odtipničas [s]
T=(0:len-1)*Ts; %časovnivektor

f1 = 5; %frekvenca 1. sinusnegasignala [Hz]
f2 = 50; %frekvenca 2. sinusnegasignala [Hz]
signal=sin(2*pi*f1*T)+sin(2*pi*f2*T);%signal

spectrum=fft(signal,len);
spectrum=spectrum(1:len/2);
f = fs/2*linspace(0,1,len/2);

figure(1);
subplot(3,1,1);
plot(T(1:1000),signal(1:1000));
subplot(3,1,2);
amplitude=20*log10(abs(spectrum));
amplitude=amplitude-max(amplitude);
plot(f,amplitude);
axis([0,100,min(amplitude),max(amplitude)]);
subplot(3,1,3);
phase=unwrap(angle(spectrum))*180/pi;
plot(f,phase);
axis([0,100,min(phase),max(phase)]);
time=toc;
```

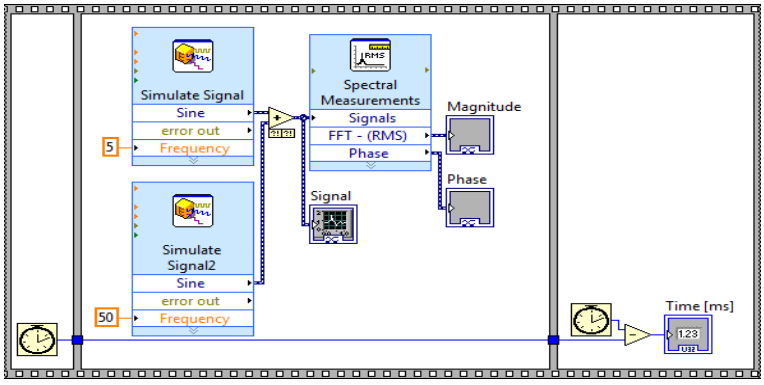

Figure 3 – LabVIEW code for FFT calculation

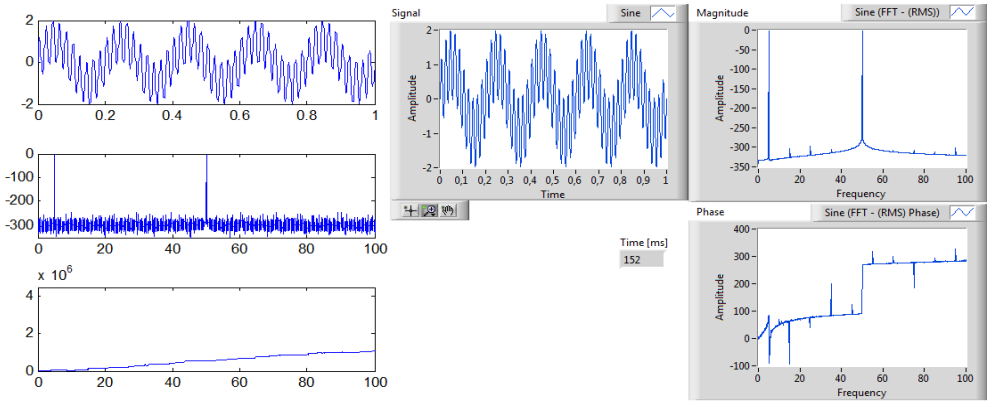

Figure 4 – FFT results in MATLAB (left) and LabVIEW (right)

MATLAB has a built in function for spectral analysis, if we want to use a built in function in LabVIEW, LabVIEW Full Development System is needed. Results of both algorithms are shown on figure 4. Both algorithms produce different results. Amplitude spectrum in Matlab has more noise and the phase diagrams are completely different. Other results of the comparison are included in Table 2. The program code in LabVIEW is obviously simpler and more surveyable, moreover the graphical user interface is easily created and adapted in LabVIEW. In addition to uncomplicated development the execution time is also shorter. LabVIEW is a clear winner in spectral analysis.

Table 2 – FFT calculation comparison

programming environment	execution time [ms]			code	
	average	minimum	maximum	simplicity	surveyability
MATLAB	289	271	632		
LabVIEW	162	144	188	+	+

TRANSFER FUNCTION BODE PLOT

Indispensable tool for stability analysis and design of control systems is Bode plot. Bode plot can be drawn if the system parameters are known, i.e. its transfer function. For test purposes we will use three specific transfer functions, which will be multiplied and the Bode diagram of the product will be displayed. The transfer functions used are: PI controller (2) with time constant T_I=0,01 and two first order transfer functions (3) with gains K_1=10, K_2=10 and time constants T_1=0,1, T_2=0,001.

$$F(s) = 1 + \frac{1}{sT_I} = \frac{1 + sT_I}{sT_I} \tag{2}$$

$$F(s) = \frac{K}{sT + 1} \tag{3}$$

Described algorithm can be implemented in LabVIEW using the code shown in figure 5 and in MATLAB with the following program code. To be able to use the code we need "Control Design and Simulation" add-on for LabVIEW and "Control System Toolbox" for MATLAB.

```
tic;
Tl=0.01;
K1=10;
T1=0.1;
K2=10;
T2=0.001;
Fl=tf([Tl 1],[Tl 0]);
F1=tf(K1, [T1 1]);
F2=tf(K2, [T2 1]);
figure(1);
bode(Fl*F1*F2);
h=gcr;
h.AxesGrid.Xunits='Hz';
h.AxesGrid.Grid='On';
time=toc;
```

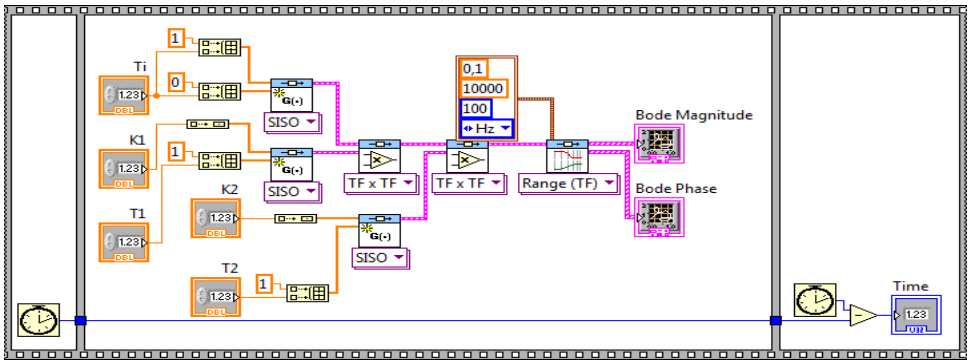

Figure 5 – Code for Bode plot in LabVIEW

Results of both algorithms are shown in Figure 6. We can see that both Bode diagrams are practically identical. The biggest difference can be seen in code. Code for the creation and multiplication of transfer functions in LabVIEW is much more complex. The same thing can be done in MATLAB using just a few lines.

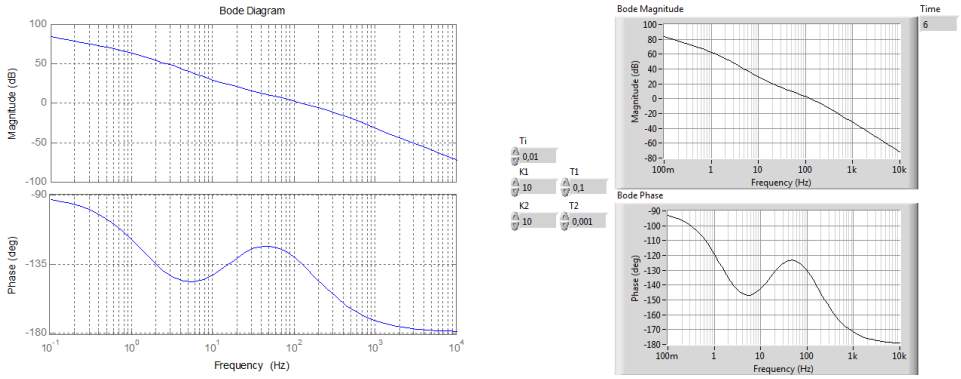

Figure 6 – Bode plot in MATLAB (left) and LabVIEW (right)

Other results of the comparison are shown in Table 3, which reveals that MATLAB needs more time to create the plot. The maximum time of the first run of the MATLAB program is longer, because the bode function needs to be loaded into memory.

Table 3 – Comparison of drawing the Bode plot

programming environment	execution time [ms]			code	
	average	minimum	maximum	simplicity	surveyability
MATLAB	188	152	1755	+	+
LabVIEW	90	60	140		

DC MOTOR CONTROL SIMULATION

LabVIEW can simulate mathematical models if "Control Design and Simulation Module" is installed. The simulations are similar to MATLAB with the Simulink add-on. User must first draw a block diagram of the simulation model. In LabVIEW the user can also create graphical user interface where the simulation parameters can be set and the responses displayed. The simulations can include most of the features available for "classical" programming using LabVIEW or MATLAB.

For the sake of simplicity, we compare DC motor control. The motor parameters are shown in Table 4.

Table 4 – DC motor characteristics

Rotor resistance	R_a = 2.5+2*0.38 Ω
Rotor inductance	L_a = 0.3+2*1.5 mH
Back EMF constant	K_e = 0.0195 Vs/rad
Torque constant	K_m = 0.0195 Nm/A
Rotor inertia	J = 9.87e-6 kg.m^2
Friction coefficient	B = 1.42e-6 Nm s/rad

Dynamic behavior of DC motor can be divided into electrical (4) and mechanical (5) part. Both parts are composed of a first-order differential equation, which are linked through the torque constant (6) and through the induced voltage (7). [8]

$$U_a = I_a \cdot R_a + L_a \cdot \frac{\mathrm{d}I_a}{\mathrm{d}t} + E_b \qquad (4)$$

$$T_E = B \cdot \omega + J \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} + T_L \tag{5}$$

$$T_E = K_m \cdot I \tag{6}$$

$$E_b = K_e \cdot \omega \tag{7}$$

Simulation model is designed according to equations (4-7)with two (current and speed) PI controllers added for cascade control. Simulation model can be designed in MATLAB/Simulink and import it into LabVIEW. Figure 7 shows both simulation models. [8]

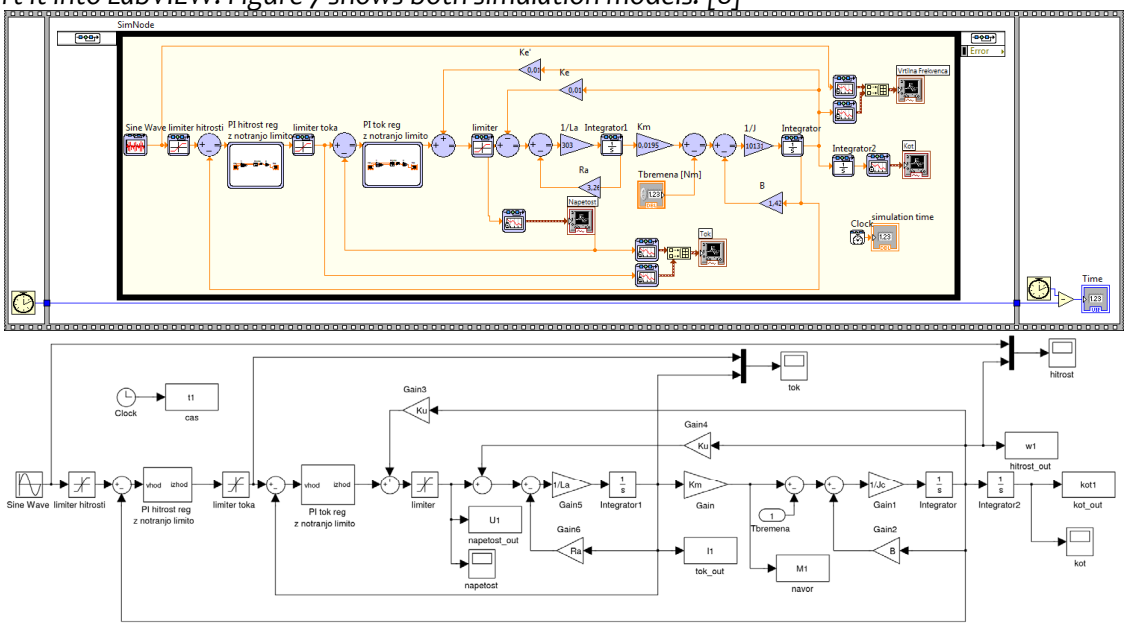

Figure 7 – DC motor simulation in LabVIEW (above) and Simulink (below)

For comparability reasons the numerical simulation in LabVIEW an Simulink were executed with a fixed time-step of 0,1 ms and using the same solver (Runge-Kutta 4). The results of both algorithms are shown in Figure 8. Except for small initial voltage instability in a simulation performed with LabVIEW, there are no significant differences.

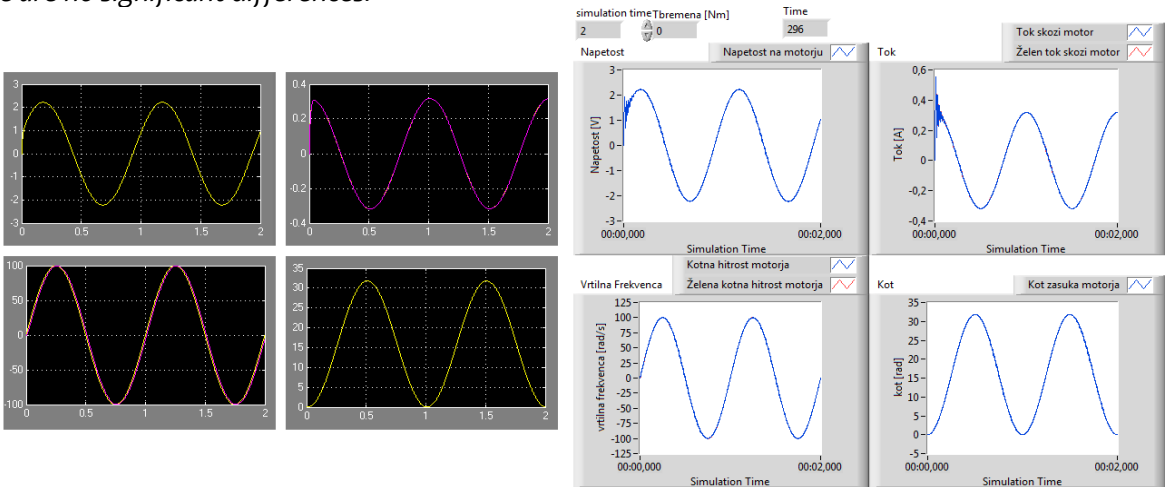

Figure 8 – Simulation results in Simulink (left) and LabVIEW (right)

Execution time in Simulink was measured using "tic" and "toc" functions defined in StartFcn and StopFcn. The maximum execution time in Simulink was measured in first run and appeared due to model compilation on first run. LabVIEW compiles the code as it is changed and therefore are no delays on first runs. The complexity is similar in both programs, but surveyability is much better in LabVIEW because of colors. Moreover LabVIEW offers easy construction of graphical user interface, which simplifies updating of parameters and displaying of results.

Table 5 – Comparison of simulation in MATLAB/Simulink and in LabVIEW

programming environment	execution time [ms]			code	
	average	minimum	maximum	simplicity	surveyability
MATLAB	215	135	529	+	
LabVIEW	309	295	322	+	+

CONCLUSIONS

The comparison showed that MATLAB is much better for computation than LabVIEW, mostly because classical program code is much more appropriate for calculations than block diagrams. Simulations also execute faster in MATLAB, but the first run delay is sometimes annoying. It is clear that the functions for classical engineering calculations (Bode, FFT) are very well optimized in LabVIEW and faster than in MATLAB. The biggest advantage of LabVIEW is fast and simple construction of the graphical user interface that facilitates the updating of parameters (no need to interfere with the code) and elegant presentation of the results. Creating a comparable user interface in MATLAB could be more painful and limited. Another advantage of LabVIEW is that most MATLAB functions are accessible from LabVIEW via the MathScript Node, which can actually pass data to m code, execute it and get results back.

If you have some basic experience in programming, then learning both programming languages should not be a problem. In case that you had never programmed before and you are slightly familiar with flow-charts or process diagrams, then LabVIEW would be easier to learn.

To sum up, we could say that in both software packages the original purpose is still visible – MATLAB for computation, and LabVIEW for acquiring, processing and displaying signals. If you are mainly solving equations and evaluating mathematical expressions then MATLAB is the way to go. But on the other hand if you will use the software mostly for signal acquisition and processing then go for LabVIEW.

It would be interesting to compare the two software packages in the signal acquisition and generation, and to compile their code for use on microcontrollers.

ACKNOWLEDGEMENT

Operation part financed by the European Union, European Social Fund. Operation implemented in the framework of the Operational Programme for Human Resources Development for the Period 2007-2013, Priority axis 1: Promoting entrepreneurship and adaptability, Main type of activity 1.1.: Experts and researchers for competitive enterprises.

REFERENCES

[1.] *Gross, B., Kozek, M., Jörgl, H.: Identification and Inversion of Magnetic Hysteresis Using LabVIEW and MATLAB, International Journal of Online Engineering iJOE, 1 (2005).*
[2.] *Xiong, Y., Qin, B., Wu, M., Yang, J., Fan, M.: LabVIEW and MATLAB-Based Virtual Control System for Virtual Prototyping of Cyclotron, Proceedings of PAC07, New Mexico, USA, 2007, str. 281-283.*
[3.] *Coito, F., Almeida, P., Palma, L.: SMCRVI-A LabVIEW/MATLAB Based Tool for Remote Monitoring and Control, 10th IEEE Conference on Emerging Technologies and Factory Automation, Catania, 2005, str. 1039-1044.*
[4.] *Tekin, R.: MATLAB and LabVIEW in Modeling, Analysis and Real Time Control of a Motion Control System, 8th IEEE International Conference on Control and Automation, Xiamen, China, 2010, str. 2077-2081.*
[5.] *Kodosky, J.: The History of LabVIEW, http://zone.ni.com/wv/app/doc/p/id/wv-2868/upvisited/y, 2011-11-17*
[6.] *Grossman, L.: LabVIEW 1.0, it was cool to see you, http://www.dmcinfo.com/Portals/0/Blog%20Pictures/LabVIEWv1small.jpg, 2011-11-17*
[7.] *Moler, C.: The Origins of MATLAB, http://www.mathworks.com/company/newsletters/news_notes/clevescorner/dec04.html, 2011-11-17*
[8.] *Tašner, T., Lovrec, D.: Napredna raba LabVIEW za sisteme nadzora in regulacij v hidravliki, Fluidna tehnika 2011, Maribor, 2011, str. 299-310*

ANNALS OF FACULTY ENGINEERING HUNEDOARA
– INTERNATIONAL JOURNAL OF ENGINEERING