

# MATLAB environment as CAL tool

*Bogumiła MROZEK, Zbigniew MROZEK<sup>1</sup>*

## ABSTRACT

*In university environments, MATLAB is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.*

*Authors give examples of educational topics in mathematics, physics, mechanics, electrical engineering and control (numerical integration, differential equations, symbolic algebra, simulation of DC electric drive system). All examples were prepared with version 4.2c of MATLAB and 1.3c of SIMULINK and then upgraded to a new versions 5.0 and 2.0 respectively.*

*Using of MATLAB to prepare on-line presentation helps student to understand difficult parts of lecture. Student may save time speeding up programming and computation overload during labs and projects.*

---

<sup>1</sup> Cracow University of Technology (Politechnika Krakowska)  
PL 31-155 KRAKÓW, Poland, Warszawska 24  
E-mail: bmrozek@usk.pk.edu.pl. zbigniew.mrozek@pk.edu.pl.  
phone (48-12) 130076,

# 1. Introduction

Higher education institution should equip young people with the conscience and ability to function in era of information technology. Computer Aided Learning give student experience and reflection to solve real problems and to identify links between their chosen field and others. In university environments, MATLAB is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. Using MATLAB means less time consuming calculations and programming - more diligent work on problem solution.

MATLAB is a high-performance 4-th generation programming language for technical computing. It integrates computation, visualisation, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- mathematics and computation
- modelling and simulation,
- data analysis, exploration and visualisation

MATLAB is an interactive system whose basic data element is an array. This allows to solve many technical computing problems in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran.

MATLAB features a family of application-specific solutions called toolboxes. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, identification, image processing, neural networks, statistics, symbolic mathematics, fuzzy logic, wavelets, simulation, and many others.

## 2. Teaching mathematics with MATLAB

It is suggested to introduce new study program of mathematics to give students not only skills in finding analytical solution of mathematical problems, but also understanding of numerical methods. Using of MATLAB environment (e.g. cheap Student Edition of MATLAB or classroom kit licence) is very profitable, as student may test and discuss many algorithms.

### 2.1 Linear algebra

MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation and linear algebra. This allows to solve many computing problems with matrix and vector formulations.

#### 2.1.1 Matrix and table functions in MATLAB

Basic data element in MATLAB is an array that does not require dimensioning. All basic matrix operation are supported. There are also array operations, defined on element-by-element basis. The most important matrix functions are:

- **inv**, **pinv** - matrix inverse and pseudoinverse.
- **eig** - eigenvalues and eigenvectors.
- **svd** - singular value decomposition.
- matrix division.

1.  $\mathbf{A} \setminus \mathbf{B}$  is the matrix division of  $\mathbf{A}$  into  $\mathbf{B}$ , which is roughly the same as  $\text{inv}(\mathbf{A}) * \mathbf{B}$ , except it is computed in a different way (Gaussian elimination. or QR decomposition with pivoting - among others)
2.  $\mathbf{B} / \mathbf{A}$  is the matrix division of  $\mathbf{A}$  into  $\mathbf{B}$ , which is roughly the same as  $\mathbf{B} * \text{inv}(\mathbf{A})$ , except it is computed in a different way. More precisely,  $\mathbf{B} / \mathbf{A} = (\mathbf{A} \setminus \mathbf{B})'$ .
3. Array left division. -  $\mathbf{B} ./ \mathbf{A}$  denotes element-by-element right division.  $\mathbf{A}$  and  $\mathbf{B}$  must have the same dimensions unless one is a scalar. A scalar can be divided with anything.

### 2.1.2 Linear algebra example

Enclosed is an example [see Mrozek, 1996] of performing mesh analysis of sinusoidal, steady state electric circuit. A » character is MATLAB interactive prompt. The elements of complex impedance matrix of electric circuit are:

```
»z1=50+50j; z2=5+15j; z3=1; z12=-50j; z23=-25j; z13=50;
```

and a complex impedance matrix is

```
» Z=[z1+z12+z13   -z12   -z13;...
     -z12   z2+z12+z23   -z23;...
     -z13   -z23   3+z13+z23 ];
```

The electric circuit equation is described by the Ohm law :  $\mathbf{Z} * \mathbf{I} = \mathbf{U}$ . If voltage sources  $\mathbf{U} = [0 \ 0 \ 220]'$  are known, one may compute inverse of  $\mathbf{Z}$  matrix (using MATLAB command `inv(Z)`) and then find unknown currents vector  $\mathbf{I}$ .

More convenient is to solve linear matrix equation  $\mathbf{Z} * \mathbf{I} = \mathbf{U}$  using the left matrix division. Unknown values of three currents  $\mathbf{I}(1)$ ,  $\mathbf{I}(2)$ ,  $\mathbf{I}(3)$  are calculated easily as complex numbers

```
» I=Z\U
```

```
I =
1.3322 - 1.9790i
2.9878 - 2.3023i
4.9668 - 0.9701i
```

### 2.2 Numerical integration with MATLAB

As an example, an integral  $Qa = \int_0^1 \frac{1}{1+x^2} dx$  will be computed.

An exact value is known:  $qa = \pi/4$ . Using MATLAB, one has to prepare so called M-file (*Matlab's function file*) to define an expression  $(1/(1+x^2))$ .

```
function y = fun4i(x)
y = 1./(1+ x.^2);
```

When M.-file is ready, the MATLAB library quadrature function **quad** (*Simpson method*) or **quad8** (*Newton-Cotes method*) should be used. The synopsis is the same for both

```
Q = QUAD8('Fname', A,B, [rel_tol, abs_tol],TRACE)
```

where:

'Fname' - string, name of function M-file, A,B - limits of integration

rel\_tol, abs\_tol - (optional parameters) relative and absolute error

TRACE - (optional parameter) - if TRACE≠0, a progress of integration is shown on graph

To compare the algorithms, student can browse the source code of **quad.m** and **quad8.m**. Then he may compare the computational results

```
>qa= pi/4 %exact solution
qa =
0.78539816339745

>q1=quad('fun4i',0,1) %Simpson method
q1 =
0.78539812561468

>q2=quad8('fun4i',0,1,[1e-5 1e-4],1) %Newton-Cotes method
q2 =
0.78539816339744
```

Singularities in equation may cause serious troubles during computation, leading to unpredictable result. Student should be warned about it, then he should identify the problem and solve it. An expression

$$Q = \int_0^1 \frac{1}{\sqrt{x}} e^x dx$$

is singular for  $x=0$ . To solve it numerically one has to do integration by parts

$$\int_0^1 \frac{1}{\sqrt{x}} \cdot e^x dx = \frac{2}{3} e + \frac{4}{3} \int_0^1 x^{3/2} e^x dx$$

Student has to define two function M-files: *fun5ia.m* and *fun5ib.m*

```
function y = fun5ia(x)
y = x.^(-0.5).*exp(x); %original problem

function y = fun5ib(x)
y = x.^(1.5).*exp(x); %integration by parts
```

Then solution of original problem:  $q1 = \text{infinite}$  (*wrong result*) and result of integration by parts  $q2=2.92530496501295$  (*correct*) are obtained.

```
>q1=quad8('fun5ia',0,1)
q1 =
Inf

>q2 =exp(1).*2./3 + 4./3* quad8('fun5ib',0,1)
```

$$q^2 = 2.92530496501295$$

### 2.3 Solving differential equations

Student may solve an equation  $y''-y = 4\sin t+5\cos 2t$  with initial condition  $y(0)=-1, y'(0)=-2$ . The analytical solution is  $ya=-2.\sin(t)-\cos(2.t)$ . First step is to transform the second order equation into set of first order equations:

$$\begin{aligned} dy(1) &= y(2) \\ dy(2) &= 4.\sin(t) + 5.\cos(2.t) + y(1) \end{aligned}$$

The above set is used to define function M-file *eqdif1.m*

```
function dy =eqdif1(t,y)
dy = [y(2); 4.*sin(t) + 5.*cos(2.*t)+ y(1)]; %set of equations
```

One may save following lines into script M-file:

```
options= odeset('RelTol',1e-5); %relative error
[t,y]= ode45('eqdif1',[0 16],[-1 -2],options);
%Runge-Kutta method, 4/5 order
[t23,y23]= ode23('eqdif1',[0 16],[-1 -2],options);
%Runge-Kutta method, 2/3 order
[t13,y13]= ode113('eqdif1',[0 16],[-1 -2],options);
%Adams-Bashforth-Moulton method

ya=-2.*sin(t)-cos(2.*t); %known analytical solution
plot(t,ya,'r',t,y(:,1),'--',...
t23,y23(:,1),'m-.',t13,y13(:,1),'g:')
legend('ya','ode45','ode23','ode113',0)
text(5.4,-2.3,'ya = -2sin(t) - cos(2t)')
title('\bf{Analytical solution and numerical solution using}...
\it{ode45, ode23, ode113}')
```

Student should observe that even using high order method **ode45** or a variable order Adams-Bashforth-Moulton method **ode113**, the computation results for time=16 (Fig.1) are wrong. He should find that it is essential to verify the computation results (e.g. using another integration algorithm), as unpredictable results can be obtained if wrong numerical integration algorithm was chosen or due to too high computation round-off errors.

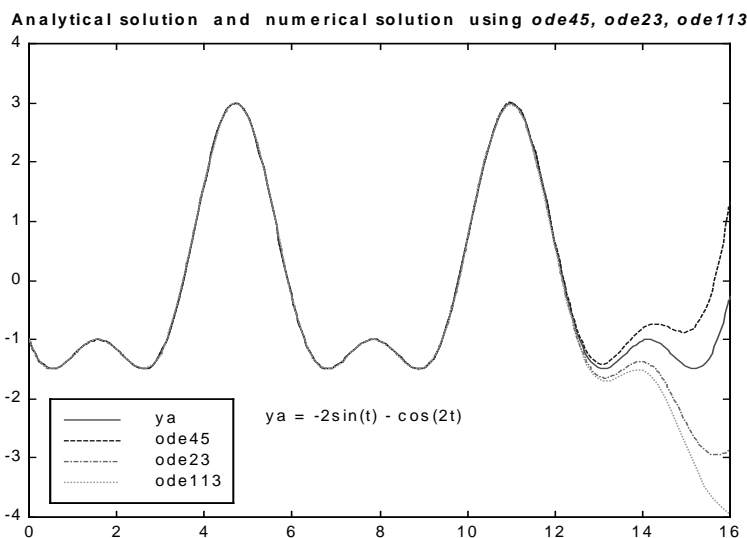


Fig. 1: Solution of differential equation, time  $\in [0,16]$

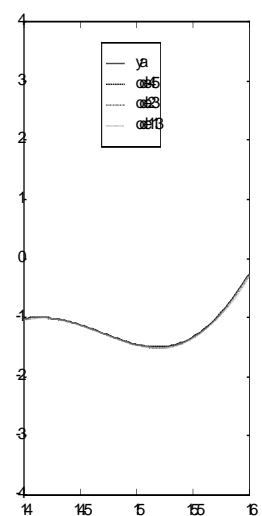


Fig.2: time  $\in [14,16]$

The correct results (see Fig. 2) can be obtained if lower values of max. step and error limits are set:

```
» options= odeset('RelTol',1e-7,'AbsTol',1e-8,'MaxStep',1e-1);
```

Using *Handle Graphics* of MATLAB, one can plot only region of interest, e.g. part of diagram for `time=[14,16]`, as on Fig. 2

```
» set(gca,'Xlim',[14,16],'YLim',[-4,4],'YLimMode','manual')
```

Integration algorithms of MATLAB are very efficient: computation time for example of Fig. 1 was only 6.98 seconds and when lower step and error limits were set (Fig. 2), computation time was 23.14 seconds.

## 2.4 Using Symbolic Math Toolbox

The **Symbolic Math Toolbox** integrates powerful symbolic and variable precision computing into the MATLAB environment. The toolbox is based on an embedded version of Maple V. Some interesting commands are:

- Calculs: **diff** - differentiate, **int** - integrate, **taylor** - Taylor series,
- Linear Algebra: **inv** - symbolic matrix inverse, **det** - symbolic determinant.  
**eig** - symbolic eigenvalues and eigenvectors.  
**poly** - symbolic characteristic polynomial.
- Solution of Equations: **solve** - symbolic solution of algebraic equations.  
**dsolve** - symbolic solution of differential equations.  
**finverse**, **compose** - functional inverse and composition.
- Simplification: **simplify**, **expand**, and **eval** - evaluate.
- Transforms. **fourier**, **laplace**, **ztrans** - Fourier, Laplace and Z transform.  
**ifourier**, **ilaplace**, **iztrans** - inverse transforms

Here is an example of **symbolic integration** (see chapter 2.2)

```
» syms x %define x as independent variable
» y = int(1./(1+ x.^2)) % symbolic integration
y =atan(x)
» y = int(1./(1+ x.^2),0,1) % exact solution for x∈[0,1]
y =1/4*pi
```

One can find **analytical solution of differential equation** described in chapter 2.3:

```
» y=dsolve('D2y-y=4*sin(t)+5*cos(2*t)', 'y(0)=-1,Dy(0)=-2')
y = -(2*exp(t)*sin(t)+2*cos(t)^2*exp(t)-exp(t))/exp(t)
» yy=simplify(y) % the above expression needs simplification
yy =-2*sin(t)-2*cos(t)^2+1
```

Substituting  $\cos 2t = \cos^2 t - \sin^2 t$  and  $1 = \cos^2 t + \sin^2 t$  into the above result, one gets the same answer as in chapter 2.3.

### 3. Solving technical problems with MATLAB

#### 3.1 An damped oscillation system used in physics, mechanics and electrical engineering.

The equation of motion of the mass  $m$ . is given by the second-order differential equation

$$mx'' + \beta x' + kx = u(t)$$

The above equation describes many oscillating phenomena in physics, mechanics and electrical engineering. Using GUI tool (*graphics user interface*) of MATLAB one can define slider (black line on right side of the picture) to change dynamically value of  $\beta$  parameter.

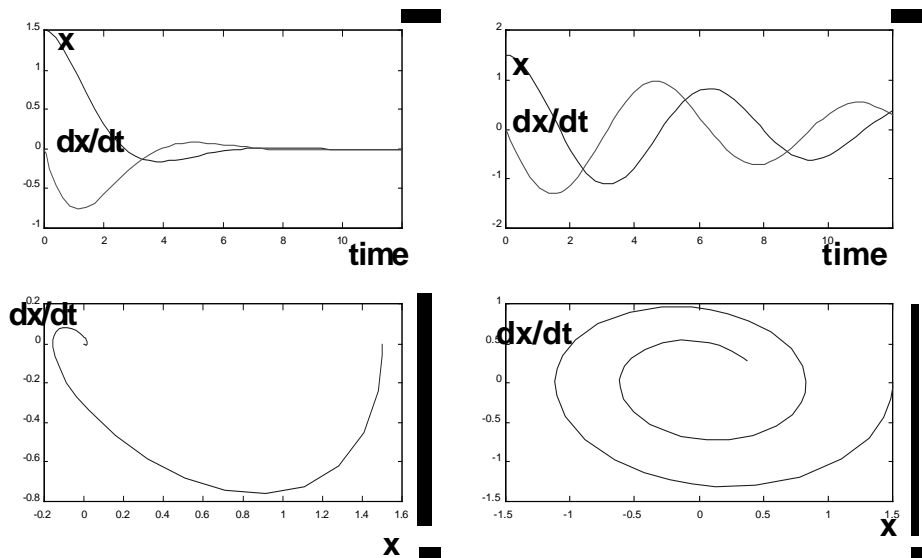


Fig. 3 Oscillation in underdamped system: high (left) and low (right) value of  $\beta$ .

Student can easily see the difference, when the equation coefficient *beta* is changed on-line. This can be done using mouse to move the slider. The figures on right show less damped system ( $\beta=0.15, k=1$ ). Source code of the above example can be found in [Mrozek, 1996].

#### 3.2 Analysis and synthesis of control systems

MATLAB gives remarkable support for teaching of control. Many classical control problems are solved easily using matrix algebra and differential equation solvers. Student can get even more assistance using SIMULINK, *Control System toolbox* or other toolboxes.

*Control System toolbox* has several functions that are useful for designing and analysing linear systems. These functions can be used to study the response of open and closed-loop systems in both the continuous and discrete time domain and in frequency domain.

For example, the Bode plot for the second-order system transfer function (confirm chapter 3.1)

$$\frac{x(s)}{u(s)} = \frac{1}{s^2 + s + 3}$$

can be generated with these statements:

```
> numerator=1; denominator=[1 1 3]; bode(numerator,denominator)
```

The plot generated is shown below

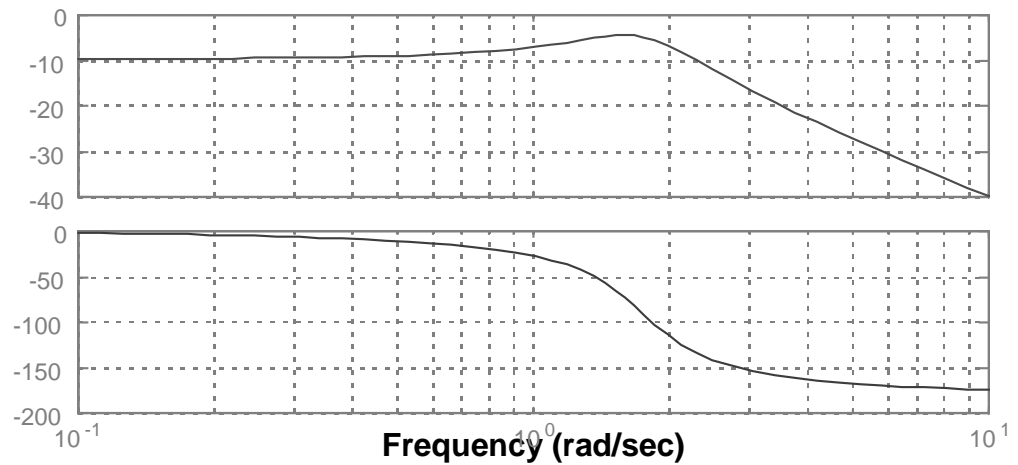


Fig. 4: Bode diagram

### 3.3 Using SIMULINK to design, test and rapid prototyping of control systems

SIMULINK (a companion program to MATLAB) is an interactive system for simulating non-linear dynamic systems. It is a graphical mouse-driven program that allows to model a system by drawing a block diagram on the screen and manipulating it dynamically. Student can open chosen block library (sources, sinks, linear, non-linear and non-linear) and import blocks into his model with mouse (drag and drop). Linear, non-linear, continuous-time, discrete-time, multivariable, and multirate system can be build and simulated.

An example of electric DC drive system is presented. Student may change any block and value of any parameter (e.g. motor and controller parameters) and observe the simulation results. Also, he may build his own model using blocks imported from library, from optional toolkits (DSP, fixed point arithmetic, non-linear control and communications) or from his own library.

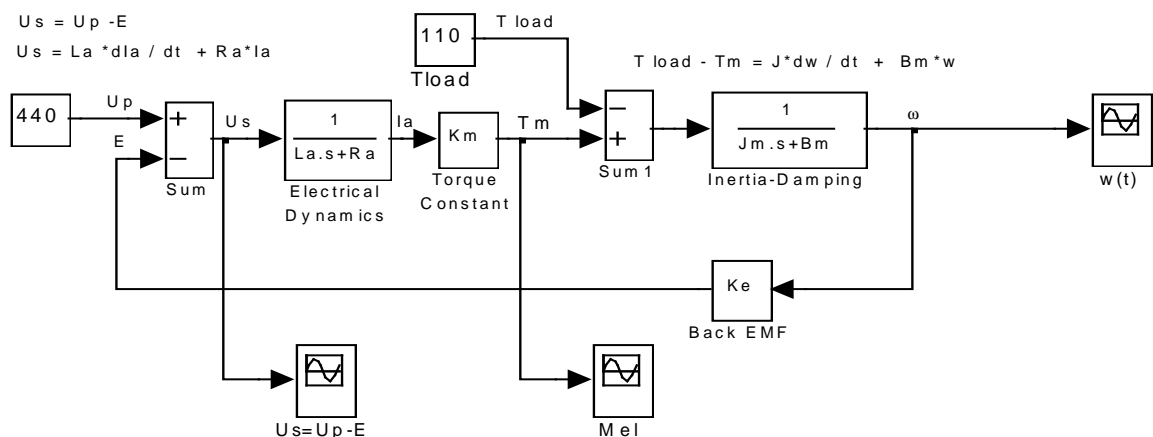


Fig. 5: SIMULINK model of DC electric motor.

SIMULINK is very easy to understand and use. Most of students can build their first simple model during first half an hour of laboratory project.

## **4. Conclusion**

Teaching how to use computer effectively should be an important objective of study program on university. This includes computation and visualisation of results as well as using of editors, data-bases (including INTERNET resources) or CAE and DTP facilities to support projects, laboratories and any other work.

MATLAB is very efficient CAL tool and can be used to support and attract educational process. It may help students to solve many computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran. Also students of non-technical faculties will find MATLAB very useful to do statistical and financial computing with high quality visualisation of results. Later, students may use MATLAB in industry, for high-productivity research, development, and analysis.

## **5. References**

1. Mrozek B., Mrozek Z. "MATLAB, Uniwersalne środowisko do obliczeń naukowo-technicznych", 3-rd edition (in Polish), PLJ Warszawa, 1996, ISBN 83-7101-325-6.
2. Getting Started with MATLAB, version V, The MathWorks Inc.,1996
3. MATLAB Language Reference Manual, version 5, The MathWorks Inc.,1996
4. Using MATLAB, The MathWorks Inc.,1996
5. Using MATLAB Graphics, The MathWorks Inc.,1996
6. The MATLAB 5.0 New Features Guide, The MathWorks Inc.,1996