1. INTRODUCTION IN MATLAB

Objectives of the paper:

- familiarity with the Matlab programming environment,
- recapitulation of some matrix computational elements,
- assimilating knowledge about solving matrix computing problems using the Matlab programming environment, by studying some examples and solving some problems.

It is recommended to go through Annex M1 before studying paragraphs 1.1 and 1.2.

1.1. Matlab programming environment elements

A brief description of the Matlab programming environment

Matlab (Matrix Laboratory) is a mathematical software, produced by The MathWorks, Inc. (http://www.mathworks.com), designed for numerical computing, numerical programming, modeling and simulation, data processing and graphic representations in science and engineering.

Matlab is composed of different program packages. One of the packages is the ***kernel***, which has its own interpreter, and contains general-purpose commands, basic mathematical functions, character string processing functions, control instructions, graphic representation functions, and so on. The other program packages, called ***toolboxes***, are collections of Matlab functions designed to solve problems in various areas, such as symbolic calculation (*Symbolic Math Toolbox*), optimization (*Optimization Toolbox*), modeling, simulation and analysis of dynamic systems (*Simulink*), statistics (*Statistics Toolbox*).

Launching the Matlab environment is done either by using the appropriate icon or by running the *Matlab.exe* program from the *bin* subdirectory of the *Matlab* directory. ***Leaving the Matlab environment*** is done either using the *Exit Matlab* command from the *File* menu or by typing one of the quit or exit commands in the command line**.**

The Matlab interface is made up of several ***windows.*** The user can choose which windows are visible at a time. Certain toolboxes have their own windows. The following will briefly describe some of the Matlab interface windows:

- *Command Window*: In this window, the instructions are entered in the *command line*, after the prompt, represented by the >> symbol. After pressing the ENTER key, each instruction is evaluated. If the instructions were correct, they are executed immediately, otherwise error messages are displayed. By default, the result of each executed instruction is displayed in the command line. If the result is not needed to be displayed, the ";" sign is added at the end of the instruction. Matlab is *case-sensitive*, so, a and A can represent two distinct variables;

- *Matlab file editor window (Editor)*: Matlab has a custom editor that opens either by using the *New* command in the *File* menu, followed by the *M-file* command, or by typing the edit command in the command line; a Matlab file, also called M-file, is an ASCII file that has the **.m** extension. An M-file can be edited in other text editors, such as *Notepad*;

- *the graphical representations window* (*Figure*): such a window opens automatically after executing a graphic representation command;

- *Help window*: provides detailed information and examples related to the Matlab environment, commands and functions; this window also includes information about installed toolboxes and various tutorials;

- *Workspace window*: contains a list, with some details, of the variables created in the command line or in script files; there is a special variable called ans (ANSwer), which is automatically created in situations where the evaluated expressions are not attributed to any variable;

- *Current Directory window*: displays the current directory files and subdirectories; it can be set in the command line by typing the cd path command, where *path* represents the relative or absolute path of the directory;

- *Command History window*: contains a list of the last commands executed; the commands in this window can be copied in the command line, or even executed.

Using command line help

The Matlab environment offers two ways to use the help:

- the Help window, which was briefly described in the previous paragraph;
- command line help, which is faster and provides concise information about the desired topic.

Command:

```
>>  help
```

causes the list of Matlab environment directories to appear. The first part of the list contains the kernel directories (subdirectories of the *matlab* directory). The second part of the list contains the directories of the installed toolboxes. The list consists of two columns: the first column contains the name of the directories, and the second column shows a short description of their contents.

Informations about a particular topic (command, Matlab function, directory) can be obtained with the command:

```
>> help subject
```

Several Matlab control functions

The main ***Matlab control functions*** are shown in Table 1.1:

Table1.1. *Matlab control functions.*

Function	Effect
cd	returns the current directory name or changes the current directory
clc	deletes the command window
clear	deletes variables and functions
disp	displays an array of numbers or characters without printing the name of the array
format	sets the display format of the data on the screen
realmin, realmax	is the smallest or the highest strictly positive floating point value that can be used in calculations
tic, toc	functions for starting and stopping a stopwatch
type	lists the contents of the mentioned M-file
which	returns the path where a Matlab file or function is located
who	lists the current variables in memory (Matlab workspace)
whos	lists current variables, their dimensions, their type

Matrices and arrays

The basic element with which Matlab operates is the ***matrix***.

A matrix is inserted into Matlab in the form of a string of values bounded by square brackets, "[" and "]", assigned to a variable, for example A. The string of values represents the matrix elements written line by line; the elements of a line are separated by a comma or space, the lines being separated from each other by semicolon (;).

The value of an element Ai,j of a matrix A is accessed through the construction: A(i,j).

Comments: 1. The Matlab assignment operator is the equal sign (=).
2. In Matlab, indices start at value 1.

Matrix operations are divided into two categories:

- operations performed according to matrix calculation rules, ie ***matrix operations*** (the appropriate operators are listed in parentheses): addition (+), subtraction (–), multiplication (*), right division (/) (for significance, see chapter 4), left division (\) (for significance, see chapter 4), exponentiation (^) (A^B, where A is a square matrix and B is a scalar or vice versa, but A and B can not be matrices simultaneously), transposition (') (A' = AT; respectively A'= $\overline{A}$T if A is a matrix of complex numbers);

- operations performed according to scalar calculation rules, between elements placed in the same position, i.e ***array operations*** (in this case the name ***array*** is used instead of *matrix*) (the appropriate operators are listed in parentheses): element-by-element addition (+), element-by-element subtraction (–), element-by-element multiplication (.*), element-by-element right division (./), element-by-element left division (.\), element-by-element exponentiation (.^), unconjugated array transpose (.') (with the exception of the transposition operation, for the other operations the operands must have the same size or one of the operands has to be a scalar).

In Matlab, ***special matrices*** that occur frequently in matrix calculations can be generated. Examples of such matrices are:

- matrix without any element (inserted to increase work speed):
  X=[];
- the matrix having all the elements equal to 1: *U=ones(dim);*
- the null matrix: *O=zeros(dim);*
- identity (unity) matrix: *I=eye(dim);*

where *dim* can have one of the following meanings:

- the matrix order, in the case of square matrices;
- the size of the matrix in the form: *number_lines*, *number_columns*, in the case of matrices that are not necessarily square;
- the size of another matrix A, in the form: *size(A)*, if it is desired to generate a matrix of the same size as the matrix A.

In Matlab, a ***vector*** is considered to be a matrix with one line or one column. The element of a vector is identified using a single index contained in parentheses.

A particular case of vectors is represented by so-called ***linearly spaced vectors***, for example, used to describe a finite interval, closed at both ends. These are vectors whose elements represent a finite sequence of consecutive terms of an arithmetic progression. Generating a linearly spaced vector v is done with the command:

```
v=a:step:b
```

where a represents the value of the first term and step the arithmetic progression ratio (called *calculation step*), v having as elements all the terms of the arithmetic progression in the interval [a,b], for a≤b and step≥0, respectively the interval [b,a], for b<a and step<0.

The vectors with step = 1 can also be generated with the command:

```
v= a:b
```

step = 1 being the only step implicitly calculated in Matlab. For any other step value, even if it is equal to -1, the step should be explicitly specified in the generating expression.

Manipulating elements of a matrix

The following types of ***matrix elements manipulation*** operations are more often used:

- Rearranging the elements of a matrix A in the form of a column vector v, using the command :

```
v=A(:)
```

The vector v contains the elements of the matrix in the order: the first column, the second column,... the last column. Because of this possibility of rearrangement, an element of a matrix can also be accessed by using a single index (as in the case of a vector), an index which actually specifies the position of the element in the column vector matrix arrangement.

- Extracting the submatrix S from a matrix A using the command:

```
S=A(v_lin,v_col)
```

where v_lin,v_col represent vectors (most often linearly spaced vectors) that specify the lines, repectively the columns used to extract the submatrix.

- Assembling large matrices from other matrices. For this operation, a command is used, whose syntax is similar to defining a matrix, the only difference being that the names of the matrix components are used instead of matrix values.

Matrix calculation

Table 1.2 summarizes the main functions used for ***matrix computation***.

Table 1.2. *Matlab functions for matrix computation.*

Function	**Sintax**	**Effect**
chol	R=chol(A)	performs the Cholesky factorization of the symmetrical and positively defined matrix A ($A=R^T*R$, where R is a upper triangular matrix)
conj	conj(A)	returns the conjugate of the matrix A with complex elements
det	det(A)	returns the determinant of the square matrix A
diag	diag(A,p) diag(A)	returns the pth diagonal of matrix A parallel with the main diagonal, that is above it (for p>0), bellow it (for p<0) or is identical with it (for p=0) returns the main diagonal of matrix A (identical with p=0)
inv	inv(A)	returns the inverse for the invertible square matrix A *Comment:* The inverse of a matrix can also be calculated with the command A^(-1)

Table 1.2. *Matlab functions for matrix computation - continuation*

Function	Sintax	Effect
lu	[L,U]=lu(A) [L,U,P]=lu(A)	performs the LR factorization of the matrix A ($P{\cdot}A=L{\cdot}U$, where L is a lower triangular matrix, U a upper triangular matrix and P is a matrix of permutations); if the first syntax is used, L will be a matrix containing a permutation of the lower triangular matrix lines so that $A=L{\cdot}U$
qr	[Q,R]=qr(A)	performs the QR factorization of the matrix A (A=Q·R, where Q is an orthogonal matrix and R a upper triangular matrix
rank	rank(A)	returns the rank of matrix A
size	[lin,col]=size(A)	returns the size of matrix A (number of lines, lin, and number of columns, col)

Comments: 1. In Matlab, the LR and QR factorizations can be performed for every matrix, not only for square ones. In this case, a matrix with any p lines and n columns is *lower triangular*, if for any i=1,2,...,p and j=1,2,..,n with i<j, the matrix element located on position (i,j) is null. Respectively, a matrix with any p lines and n columns is *upper triangular*, if for any i=1,2,...,p and j=1,2,..,n with i>j, the matrix element located on position (i,j) is null.

2. In Matlab by performing a LR factorization of a matrix of size p×n we obtain a lower triangular matrix of size p×min(p,n)and a upper triangular matrix of size min(p,n)×n.

3. In Matlab by performing a QR factorization of a matrix of size p×n we obtain a orthogonal matrix of size p×p and a upper triangular matrix of size p×n.

Strings

A string comprises one or more characters and is delimited by apostrophes, '. The apostrophes are not part of the string. A string can be assigned to a variable using the assignment operator =. Characters are internally stored via ASCII codes.

1.2. Examples

<u>Example 1.1</u>: Using command-line help, find the Matlab function for calculating arctangent and specify its call syntax.

<u>Solution</u>: Solving the problem consists of the following steps:

i) Search for a trigonometric function directory. For this following command is used:

```
>> help
```

From the list of directories shown it is noted that there is no directory intended only for trigonometric functions. However, the arctangent function is an elementary function, so it is probably found in the directory:

```
matlab\elfun                    -  Elementary math functions.
```

ii) View the contents of the directory *elfun*:

```
>> help elfun
```

In the list of Matlab functions in this directory is also the function you are looking for:

```
atan              - Inverse tangent.
```

iii) find the function call sintax:

```
>> help atan
```

The help of the *atan* function shows that the call syntax is:

```
atan(X)
```

which returns the arctangents, expressed in radians, for the vector *X* elements.

Comments: 1. The Matlab function names are written in small letters. They appear in *help* in large letters only to be highlighted.

2. If the name of the arctangent function in Matlab was assumed to be the same as that used in mathematics, *arctg*, and the help function with this name would have been called:

```
>> help arctg
```

Matlab would have displayed an error message telling the user that it did not find the M-file *arctg* (*arctg.m not found.*), which shows that the name of the Matlab function and the name of the file in which the function is implemented must be identical.

<u>Example 1.2</u>: Extract from the bellow defined matrix M: the element on line 1 and column 3, the first line, second column and the submatrix determined by the lines 1,2 and 4 and the columns 2, 3 and 4.

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ -1 & -2 & -3 & -4 \\ 0 & 5 & 0 & 7 \end{bmatrix}$$

Solution: Solving the problem consists of the following steps:

i) Inserting matrix M:

```
>> M=[1 2 3 4; 2 4 6 8; -1 -2 -3 -4; 0 5 0 7];
```

ii) Extracting the element on line 1 and column 3:

```
>> M(1,3)
ans =        3
```

iii) Extracting the first line :

```
>> M(1,:)
ans =
     1     2     3     4
```

iv) Extracting the second column:

```
>> M(:,2)
ans =
     2
     4
    -2
     5
```

v) Extracting the submatrix determined by the lines 1,2 and 4 and the columns 2, 3 and 4:

```
>> M([1,2,4],2:4)
 ans =
     2     3     4
     4     6     8
     5     0     7
```

Comments: 1. The extraction results were not attributed to any variables. In this case, Matlab has implicitly assigned the result of each extraction operation to the variable *ans*.

2. If it is desired to extract all the elements of a line, instead of the vector indicating the columns on which the extraction is made, use the abbreviated form „:”.A similar observation also applies to the extraction of a column.

3. The matrix M by not being explicitly reassigned, is not affected by any of the above operations. This can be seen by displaying the contents of the matrix at the end:

```
>> M
M =
     1     2     3     4
     2     4     6     8
    -1    -2    -3    -4
     0     5     0     7
```

1.3. Problems to solve

P1.1. Let $A = \begin{bmatrix} 2 & 3 & 0 \\ -5 & 0 & 8 \end{bmatrix}$ and $B = \begin{bmatrix} 4 & -3 & 1 \\ -1 & 0 & 5 \end{bmatrix}$. Write the instructions that display the results of all matrix and array operations, that can be performed with *A* and *B*.

P1.2. Let $Z = \begin{bmatrix} 2-3i & -i \\ 4+7i & 5 \end{bmatrix}$. Apply to Z the two transposition operators and specify the differences between the displayed results.

P1.3. Exemplify the use of command-line help to find the following Matlab functions:

- the function / functions that determine the ASCII codes of a string; illustrate the use of the function / functions found for the 'Matlab' string;
- the function / functions that compare/s two strings; compare the strings ‘test’ and ‘Test’ and display the result;
- the function / functions that determines the position of a string in another string; determine the position of the string S2 = 'test' in the string $S1$=‘This test is interesting.’.

P1.4. Consider the following matrix:

$$A = \begin{bmatrix} 3 & 1 & 0 & -2 & -9 \\ 0 & -5 & 4 & 8 & 10 \\ 6 & 6 & 1 & -5 & 7 \\ 11 & -2 & 6 & 9 & 4 \\ -8 & 0 & 0 & 0 & 1 \end{bmatrix}$$

It is required to extract:

- the third line;
- the last column;
- the last line;
- submatrix determined by the lines 2-4 and the columns 1-3.

P1.5. Determine the transposed, rank and determinant for the matrix from the previous problem. Is matrix *A* invertible? If yes, determine the inverse matrix of *A*.

P1.6. Make the LR and QR factorizations, for the matrix from problem P1.4. Check the orthogonality of the obtained Q matrix.

P1.7. Show the matrix obtained by successively concatenating the matrix from problem P1.4. with the vectors:

$$u = [14 \quad 9 \quad -7 \quad 0 \quad 1], \qquad w = \begin{bmatrix} 5 \\ 0 \\ -5 \\ 0 \\ 3 \\ 2 \end{bmatrix}.$$

ANNEX M1. MATRIX ALGEBRA ELEMENTS

M1.1. Matrix Algebra Elements

a. Different types of matrices

Let A be a matrix with p lines and n columns with elements from the set of real numbers, $\boldsymbol{R}$, or complex numbers, $\boldsymbol{C}$:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p,1} & A_{p,2} & \cdots & A_{p,n} \end{bmatrix}$$

We denote $A \in M_{p,n}(\boldsymbol{K})$, where $\boldsymbol{K} \in \{\boldsymbol{R}, \boldsymbol{C}\}$. We say that matrix A **has the dimension p × n**.

A square matrix of order n is a matrix A with n lines and n columns. In this case we use the notation $A \in M_n(\boldsymbol{K})$.

The transposed of matrixA $\in M_{p,n}(\boldsymbol{K})$ is the matrix noted as $A^T \in M_{n,p}(\boldsymbol{K})$, with n lines and p columns, obtained from the matrix A by changing the lines and columns between them:

$$A = \begin{bmatrix} A_{1,1} & A_{2,1} & \cdots & A_{\mathrm{p},1} \\ A_{1,2} & A_{2,2} & \cdots & A_{\mathrm{p},2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1,\mathrm{n}} & A_{2,\mathrm{n}} & \cdots & A_{p,n} \end{bmatrix}$$

The properties of the matrix transposition operation are:

1. $(A^T)^T = A$, $\forall\ A \in M_{p,n}(\boldsymbol{K})$;
2. $(\boldsymbol{a} \cdot A)^T = \boldsymbol{a} \cdot A^T$, $\forall\ A \in M_{p,n}(\boldsymbol{K})$ and $\forall\ \boldsymbol{a} \in \boldsymbol{K}$;
3. $(A + B)^T = A^T + B^T$, $\forall\ A, B \in M_{p,n}(\boldsymbol{K})$;
4. $(A \cdot B)^T = B^T \cdot A^T$, $\forall\ A \in M_{p,n}(\boldsymbol{K})$ and $\forall\ B \in M_{n,s}(\boldsymbol{K})$.

The conjugated matrix of A $\in M_{p,n}(\boldsymbol{C})$ is the matrix noted as $\overline{A} \in M_{p,n}(\boldsymbol{C})$ that contains the conjugated elements of matrix A.

A symmetric matrix is a square matrix A that has the property $A = A^T$.

A skew-symmetric matrix is a square matrix A that has the property $A = -A^T$.

A upper triangular matrix is a square matrix A that has the property $A_{i,j} = 0$, $\forall\ i > j$.

A lower triangular matrix is a square matrix A that has the property $A_{i,j} = 0$, $\forall\ i < j$.

A square matrix of order n in which the elements outside the main diagonal are all zero is called a ***diagonal matrix of order n.***

A diagonal matrix of order n with ones on the main diagonal, is called the **identity matrix of order *n***:

$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

An orthogonal matrix is a square matrix A that has the property $A \cdot A^T = I_n$.

b. Determinant. Inverse matrix

Let us consider a square matrix A of order n with real or complex numbers as elements. Let S_n be the group of all the element permutations of group $M=\{1,2,\dots,n\}$.

The number:

$$det\,A = \sum_{(k_1,k_2,\dots,k_n)\in S_n} (-1)^{I(k_1,k_2,\dots,k_n)} A_{1,k_1} A_{2,k_2} \dots A_{n,k_n}$$

where $I(k_1,k_2,\dots,k_n)$ is the number of all the inversions of the permutations $(k_1,k_2,\dots,k_n)$, is called ***the determinant of matrix A*** or ***determinant of order n***.

Observation. It is called an ***inversion of the permutation*** $(\boldsymbol{k_1},\boldsymbol{k_2},\dots,\boldsymbol{k_n}) \in S_n$**,** any pair $(i,j)\in M\times M$ that has the property $i<j$ and $k_i>k_j$.

A matrix $A\in M_n(\boldsymbol{K})$ that has the property $det\,A = 0$ is called a ***singular matrix***.

A nonsingular matrix is a matrix $A\in M_n(\boldsymbol{K})$ that has the property $det\,A \neq 0$.

A square matrix $A\in M_n(\boldsymbol{K})$ is ***an invertible matrix*** if a square matrix $B\in M_n(\boldsymbol{K})$ exists so that:

$A\cdot B= B\cdot A=I_n$.

Matrix B of the above relationship is called ***the inverse matrix of matrix A*** and is denoted by A^{-1}. The main properties of matrix inversion are:

1. A square matrix is ivertible if and only if it is a nonsingular matrix..
2. $\forall\ A\in M_n(\boldsymbol{K})$, A – invertible, it results that A^{-1} is invertible and $(A^{-1})^{-1}=A$;
3. $\forall\ A\in M_n(\boldsymbol{K})$, A – invertible, it results that A^T is invertible and $(A^T)^{-1}=(A^{-1})^T$;
4. $\det(A)^{-1} = \frac{1}{det\,A}$, $\forall\ A\in M_n(\boldsymbol{K})$;
5. $\forall\ A\in M_n(\boldsymbol{K})$ and $\forall\ B\in M_n(\boldsymbol{K})$, A,B – invertible, it results that $A \cdot B$ is an invertible matrix and $(A\cdot B)^{-1} = B^{-1}\cdot A^{-1}$.

Having matrix $A\in M_n(\boldsymbol{K})$, we call ***the minor of element $A_{i,j}$ from det A,*** the determinant of order n-1that can be obtained from $det\,A$ by eliminating line i and column j. The minor of element $A_{i,j}$ is noted with $M_{i,j}$.

It is called an ***algebraic complement*** or ***cofactor of element $A_{i,j}$ from det A***, the number:

$$\underline{A}_{i,j}=(-1)^{i+j}M_{i,j}$$

It is called the ***matrix A adjunct***, and is denoted by A^*, the matrix:

$$A^* = \begin{bmatrix} \underline{A}_{1,1} & \underline{A}_{2,1} & \cdots & \underline{A}_{n,1} \\ \underline{A}_{1,2} & \underline{A}_{2,2} & \cdots & \underline{A}_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{A}_{1,n} & \underline{A}_{2,n} & \cdots & \underline{A}_{n,n} \end{bmatrix}$$

Using the above notions, the inverse matrix of matrix A can be expressed in the following form:

$$A^{-1} = \frac{1}{det\,A} A^*$$

c. The rank of a matrix

Let us consider a square matrix A of order n with real or complex numbers as elements.

It is called ***a k order minor of matrix A*** any determinant of order k formed from k^2 elementa of A (k lines and k columns, keeping the elements order).

It is said that matrix A has **the rank *r***, if A has a minor of order r that is not null, and all minors of A of greater order than r, if such minors exist, are null. The fact that matrix A has the rank r is written in the form:

$rank\ A = r$

d. Matrix factorization

By factoring a square matrix A of order n, the matrix is decomposed into a product of two (or more) matrices of the same order as A. Typically, factors matrices are of certain types (triangular, orthogonal, etc.). Thus, several factorization methods are distinguished, among which we mention the following:

• ***LR factorization*** (also known as *LU* factorization)

The matrix $A \in M_n(\boldsymbol{K})$ is written as:

$$A = L \cdot R$$

where $L \in M_n(\boldsymbol{K})$ is a lower triangular matrix, and $R \in M_n(\boldsymbol{K})$ is a upper triangular matrix.

A particular case is the LR factorization of symmetric and positively defined matrices, called ***Cholesky factorization***. În acest caz, $L=R^T$.

Comment. Let A_k be the submatrix determined by the first k lines and first k columns of the square matrix A of order n. Matrix A is ***positively defined***, if and only if the determinants of all the submatrices A_k, $k=1,2,\ldots,n$ are strictly positive.

• ***QR factorization***

The matrix $A \in M_n(\boldsymbol{K})$ is written as:

$$A = Q \cdot R$$

where $Q \in M_n(\boldsymbol{K})$ is an orthogonal matrix, and $R \in M_n(\boldsymbol{K})$ is a upper triangular matrix.