1 Introduction to MATLAB

This chapter is intended to get the user started using MATLAB through simple exercises in numerical calculations. The chapter starts by describing how to download and install MATLAB in a Windows environment. Installation of the software in other operating systems is very similar and is explained in detail in the MATLAB website.

What is MATLAB?
MATLAB is a numerical, programming and graphics environment with a variety of applications in science and engineering. MATLAB is a self-contained package including a large number of intrinsic numeric, programming and graphics functions.

Getting started with MATLAB

To *get started*, launch the MATLAB application.
To *load a value into a variable* use an assignment statement (one that includes the equal sign), e.g., a = 3.2. Try the following exercises for simple arithmetic operations. Type the commands shown in front of the MATLAB prompt. In this chapter, the prompt `EDU»` is used. This prompt belongs to a student version of MATLAB.

```
a = 3.2 <return>
b = 6.4 <return>
a+b <return>
a-b <return>
a*b <return>
a/b <return>
a^b <return>
who <return>
```

The last command will return a list of the active variables in your worksheet. Many of them are system variables that cannot be modified by the user.

MATLAB has a number of *special constants*, such as *i* and *pi*, corresponding to the unit imaginary number, and π = ratio of circumference to diameter, respectively. The base of the natural logarithms, *e* can be obtained by evaluating *exp(1)*. The value *eps* is another special constant corresponding to the maximum value for which *1 + exp(1)ps = 1*. Other important constants are *inf* = infinity, and *nan* = not-a-number. Boolean (i.e., logical) constants are *1 (true)* and *0 (false)*.

Try the following exercises to see what values are returned by MATLAB:

```
exp(1) <return>
i <return>
j <return>
pi <return>
eps <return>
inf <return>
nan <return>
3>2 <return>
3<2 <return>
```

Comments in MATLAB are represented by the percentage sign (%). Anything in front of %/ is taken as a comment. For example, enter:
`a = 4.5 % redefining a` <return>

MATLAB will return the value of 4.5 for a and ignore the comment after the % character.

Simple operations with MATLAB

Simple scalar operations: the following exercises will get you acquainted with a few of MATLAB's abilities for operating with scalar values.

`a = 2` <return>
`b = 3` <return>
`Save('a')` <return>
`clear a` <return>
`a` <return>
`b` <return>
`load('a')` <return>
`a` <return>
`exp(a) + exp(b)` <return>
`sin(a*pi/b)` <return>

(***Note***: the clear command is used to eliminate variables, as in *clear a*, as shown above. By itself, *clear* deletes all variables recently defined. Therefore, be very careful when using this command).

Vectors:

- To enter vectors use the square brackets and separate the elements with commas or blanks for a row vector, e.g.: v = [-1. , 2., pi].

- The transpose of a vector (or matrix) is expressed using the apostrophe, for example, type: v'

- To enter a column vector, use any of the following procedures: w = [3; -2; 5] <return>

Or, use

r = [6 <return>
-2 <return>
10] <return>

- You can create a row vector by indicating a starting value, an increment (or decrement), and an ending value, all separated by the colon (:) operator as follows:

vector_name = starting_value : increment : ending value

for example: x = -10.0 : 0.5 : 10.0 <return>

- If you prefer to store it as a column vector, try the following: `xt = x'` <return>

- Let's apply a function to the vector x, try: `y = sin(x*pi/10)` <return>

- We can plot the vectors ***x,y*** using: `plot(x,y)` <return>

. [Type `help plot` <return> for more information]

- Let's clear the MATLAB variables and operate with other vectors: Type `clear` <enter> followed by `clc` <enter>. The command *clc* clears the Matlab interface.

- Enter the row vectors, u = [-1. 2. 3.] and v = [6. -10. 0.]

- Perform the following operations:

u + v <return>
u - v <return>
u*v <return>
u*v' <return>
u'*v <return>

- To extract elements of the vectors, try:

u(3) <return>
u(2) + v(1) <return>

- Try the following exercise: `a = 1; b = 2; c = 3; r = [a, b, c]` <return>

- To suppress MATLAB responses use the semi-colon after entering a command. For example, try: `s = [-1., 2.];` <return>

- Vectors can also contain characters as their elements, e.g.,
`letters = ['a', 'b', 'c', 'd']` <return>

Note: Expressions such as 'a', 'b', etc., are referred to as strings. Therefore, only those string operations such as *concatenation*, *part*, etc. are allowed for vectors with character elements. MATLAB strings and string operations are presented in a subsequent chapter.

Matrices:

- Here are several ways to enter matrices: (use `clear` and `clc` to clear the memory and the screen)

A = [1. 2. 3.; 4. 5. 6.; 1. -1. 0.] <return>
B = [1. 1. 1. <return>
2. 3. -1. <return>
5. 7. -2.] <return>
u = [1. 3. -5.]; v = [4. 2. 3.]; w = [-1. 0. 1.]; <return>
C = [u; v; w] <return>
r = [u, v, w] <return>
D = [u' v' w'] <return>

- *Matrix operations*: try the following operations:
-

A + B <return>
C - D <return>
A*B <return>
B*A <return>
C*u <return>
D*v' <return>
rank (A) <return>
inv(A) <return>

cond(B) <return>
det(C) <return>
A*inv(A) <return>
inv(B)*B <return>
eig(A) <return> (calculates eigenvalues)
trace(C) <return>

(**Note**: to find out more about any of the commands listed here type *help* followed by the command name, e.g., *help spec*).

Solution of linear systems: some possibilities are (use `clear` and `clc` to restart memory and screen):

```
A = [1. 3. 2.; 2. 1. -1.; 5. 2. 1.]; b = [2; 3; 4]; <return>
xa = inv(A)*b <return>
xb = linsolve(A,b)<return>
xc = A\b <return>
```

(Note: type *help linsolve* to learn more about this command).

Simple MATLAB Input and Output

Output: To get a list of your current session use the function *diary*. The format is as follows: *diary (output_filename)*, where the filename is written within quotes. To end collecting output in the current diary output file use *diary off*. For example, try the following session:

```
clear; clc; <return>
diary ('session1.txt') <return>
A = [1. 2. 3.; 2. 3. 1.; 3. 2. 1.];b=[5; 4; -1.];<return>
A <return>
b <return>
x = linsolve(A,b) <return>
diary(0) <return>
```

Next, use NOTEPAD, or other text editor, to open the file *session1.txt* to see the contents of the file. The default directory for storing a *diary* file is the current working directory, i.e., typically the *work* subdirectory within the MATLAB directory.

Cutting and pasting: You can also copy statements or results from the Matlab interface into a text file by cutting and pasting.

Command Input: you can read MATLAB commands from a script file, which is basically a text file listing all the commands you want to use. As an example, create the following text file in the *work* subdirectory of the MATLAB directory using NOTEPAD, and call it *session2.m*:

```
clear
A = [1. 2. -3. % entering
3. 4. 5. % elements of
7. 8. 9.] % matrix A
b = [1.; 2.; 3.] % enter vector b
xa = inv(A)*b % calculate x using matrix inverse
xb = linsolve(A,b) % calculate x using MATLAB's own linsolve function
```

Then, press *clc* in MATLAB, and type: *session2*
You will see MATLAB execute all the commands in your file, stopping at the end of the file.

MATLAB command history

All commands entered in a given MATLAB session get stored into a MATLAB command history buffer. The commands are thus accessible for re-use or edition. All the command history functions are available through the option *History* under the *File* menu in the MATLAB worksheet. The most useful commands are *cntl-P* and *cntl-N*, which lets you access the previous command or the next command, respectively, in the command history buffer. (As an alternative, you could use the up and down arrow keys in your keyboard to move about the command history buffer in Matlab). Once a command is recalled from the command history buffer it can be edited by using the backspace or delete keys, or by inserting new characters by simply typing at the proper location.
For example, try the following MATLAB session:

1 - Use `clear; clc;` to clear memory and interface in Matlab.

2 - Enter the following commands (you don't need to enter the comments):

```
EDU>> x = [0:pi/20:2*pi];
EDU>> y = sin(x) + sin(2*x);
```

3 - Use *cntl-P* and edit the previous command (`y = sin(x) + sin(2*x);`) to read:

```
EDU>> z = sin(x) + cos(x);
```

4 - Use *cntl-P* once more to edit the previous command (`z = sin(x) + cos(x); `) to read:

```
EDU>> p = cos(x) + cos(2*x);
```

5 - So far you have created vectors ***x***, ***y***, ***z***, and ***p***. Next, we use the following commands to produce a plot of ***y***-vs.-***x***:

```
EDU>> figure(1); plot(x,y)
```

6 - Use *cntl-P* to edit the previous command to read:

```
EDU>> xset('window',1); plot(x,z)
```

7 - Continue using *cntl-P* to edit the last commands and produce the following plots:

```
EDU>> figure(2); plot(x,p)
EDU>> figure(3); plot(y,z)
EDU>> figure(4); plot(y,p)
EDU>> figure(5); plot(z,p)
```

Selective worksheet output

Suppose you have been working on a lengthy MATLAB session whose command history buffer contains some results that produced errors as well as some intermediary results that are not of interest to you for output. For your output, you can select only those commands relevant to your final results by using *Cntl-P* and *Cntl-N*, or the up and down arrow keys. Try the following MATLAB session that explores some basic vector operations using vectors x and y:

1 - Use `clear; clc;` to clear memory and interface in Matlab.

2 - Enter the following MATLAB commands:

```
EDU>> x = [1, 2, 5, -4]

EDU>> y = [0, 2, 3,-5]

EDU>> x*y
```

```
EDU>> x.*y

EDU>> sum(ans)

EDU>> sum(x.*y)

EDU>> x*y'

EDU>> x'*y

EDU>> y'*x

EDU>> x*y' + y'*x

EDU>> b = x*y' + y'*x
```

Note: These commands and their corresponding results represent an exploratory session for vector operations.

3 - Suppose that you are only interested in the commands defining vectors x and y, in the operations that produce the dot product of the vectors (i.e., `sum(x.*y)` and `x*y'`), and in the very last command (`b = x*y' + y'*x`). Using the *diary* command create the file *c:\myVectors.txt* and collect only the commands of interest out of the command history buffer by using *cntl-P* and *cntl-N* as needed. The resulting MATLAB session should look like this:

```
EDU» diary('c:\myVectors')
EDU» x = [1, 2, 5, -4]

x =

     1     2     5    -4

EDU» y = [0, 2, 3, -5]

y =

     0     2     3    -5

EDU» sum(x.*y)

ans =

    39

EDU» x*y'

ans =

    39

EDU» b = x*y' + y'*x

b =

    39    39    39    39
    41    43    49    31
    42    45    54    27
    34    29    14    59

EDU» diary off
```

The session, except for the very first command (`diary('c:\myVectors')`) is stored in file *c:\myVectors.txt*. This file can be edited or printed from a text editor such as NOTEPAD, the editor available with Matlab.

Current directory / creating a work directory

MATLAB uses a current directory where files are saved by default, for example when using the function *diary*. To see the current directory use:

```
EDU>> pwd
```

Under a Windows operating system, the default current directory is typically the *work* sub-directory within the Matlab installation. The command *pwd* stands for *p*rint *w*orking *d*irectory.

A preview of MATLAB functions

Here are some *useful functions* in MATLAB that we will explore in more details in subsequent chapters:

- Elementary functions: `sum, prod, sqrt, diag, cos, max, round, sign, fft`
- Sorting: `sort, find`
- Specific matrices: `zeros, eye, ones`
- Linear algebra: `det, inv, qr, svd, eig, schur, trace`
- Random numbers: `rand`
- Programming: `function, for, if, end, while, warning, error, break, return`
- Comparison symbols: `==, >=, >, <=, <, =, &` (and), `|` (or), ~ (not)
- Debugging: `pause, return`

To find out more about these functions use the help command. For example, try:

```
EDU>> help roots
EDU>> help eye
EDU>> help trace
```

Exercises

Determine the result of the following calculations using MATLAB if $a = 2.3$, $b = -2.3$, $c = \pi/2$, $x = 2/\pi$, and $y = \sqrt{3}$. Produce a diary file, e.g., *c:\Exercises1.txt*, showing the calculations in problems [1] through [5].

[1]. $(a^2 + bc + x)$
[2]. $\sin(c) + y/c$
[3]. $(a+c)/(x+y)$
[4]. $1/(\cos(c) + \ln(x))$
[5]. $(a+c)^3/b$

Enter the matrices `A = [1 -1 3; 2 5 -6; 0 7 1]`, and `B = [0, 2, 3; -1, 1, 2; 1, 2, 3]`. Then calculate the following:

[6]. sum(A)	[7]. sum(B')'	[8]. prod(A)	[9]. prod(B')'
[10]. A', transpose	[11]. diag(A)	[12]. rank(B)	[13]. inv(B)
[14]. eig(A')	[15]. C = A + B	[16]. D = A-B	[18]. P = A*B
[19]. Q = B*A	[20]. R = B*A'		

Enter the following vectors: `u = [2, -1, 3, 5], v = [1:5:16], w = [10:-2:4]`. Then calculate the following:

[21]. u'	[22]. z = u + v	[23]. y = u - v	[24]. S = u'*v
[25]. q = u*v'	[26]. r = u.*v	[27]. k = S*v'	[28]. m = V*S

Using `A = [1 -1 3; 2 5 -6; 0 7 1]` and `b = [-14, 46, 9]'`, solve the system of linear equations represented by the matrix equation $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$, where $\mathbf{x} = [x_1, x_2, x_3]'$, using:

[29]. The inverse of matrix **A**, i.e., $\mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}$
[30]. MATLAB function *linsolve*
[31]. MATLAB left-division, i.e., x = A\b

Enter a vector `x = [0:0.1:20];` then create the vectors, `y = sin(x); z = sin(x/2); w = y + x; r = y-x;` and

[32]. Plot y vs. x [33]. Plot z vs. x [34]. Plot w vs. x [35]. Plot r vs. x