One Session Introduction to Matlab

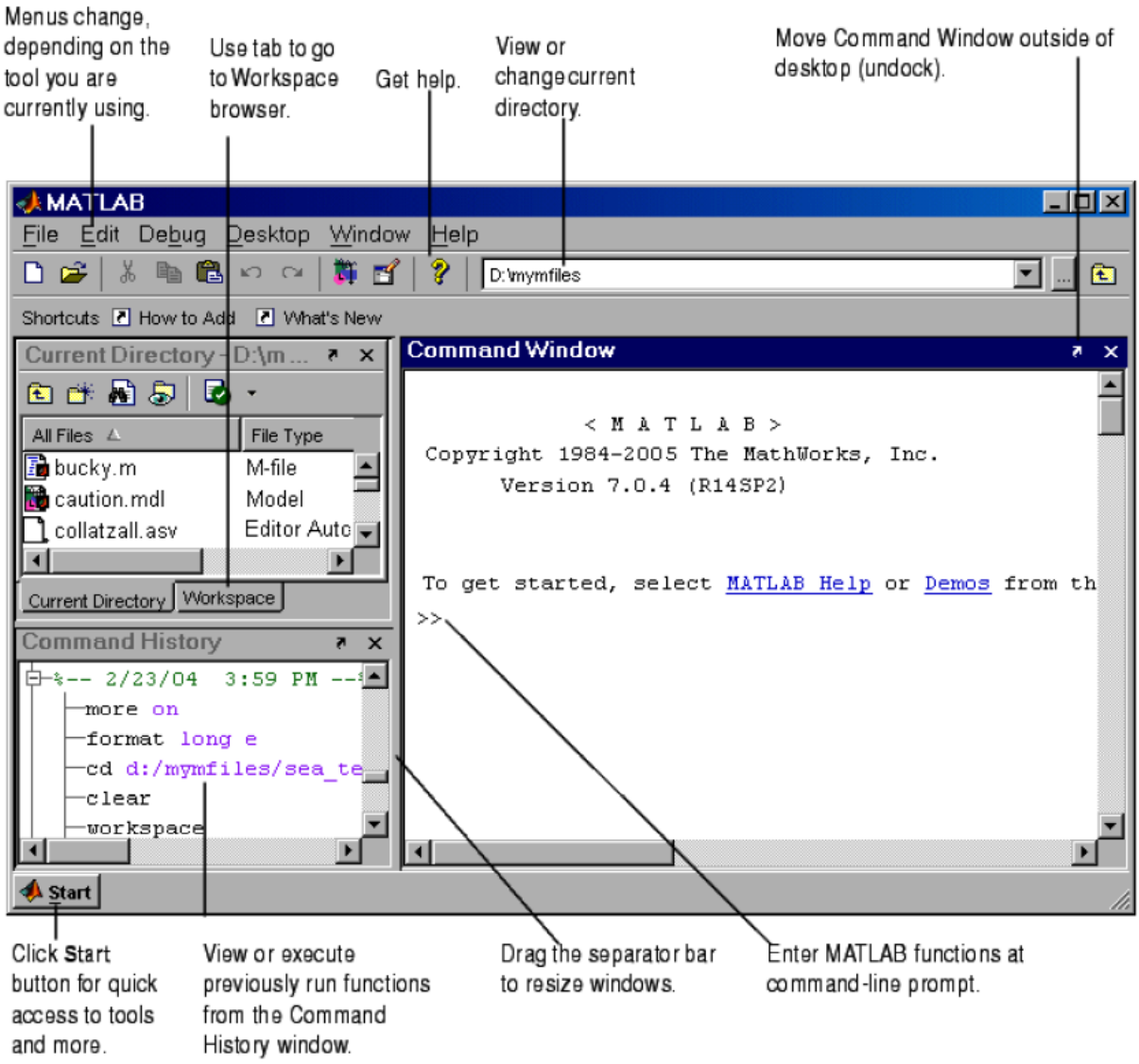
By Yasser F. O. Mohammad

1. Introduction

Matlab is a tool developed originally to handle Matrix operations but became the most commonly used tool for prototyping and fast numerical calculations in the engineering field.

This document introduces you to a subset of Matlab's functionality that you will require in our course about "Engineering Analysis".

2. Matlab Screen



3. Most Important Commands in Matlab

>>help *something*

Gives you brief information about a function in matlab.

>>help exp

>>lookfor *something*

Gives briefer information about ALL functions related to what you typed

>>lookfor exponential

>>doc *something*

Opens the HTML help with more information about the function you typed

>>doc exp

4. General Info

- Matlab is an interpreted language so the computer processes each line of code as it reads it. For this reason if you write a long Matlab program with an error in its end, this error will not be discovered except after all previous lines are executed.
- Matlab is case sensitive so the variable *x* is not the same as *X*.
- Matlab has a large number of built in functions as will be seen later that can be very useful and you can add your functions as you need. Sets of functions related to some field are usually combined in what is called a *toolbox*. You can find *toolboxes* for nearly any thing you need.
- If you start a line in Matlab with % it will be considered a comment and will not be processed (like // in C++)
- When you type any command in Matlab, it prints the answer for you except it you add a semicolon (;) after the line.

  >>2+3

  >>2+3;
- You can write multiple commands in a single line if you separate them by a semicolon (;)
- You can clear all the defined global variables in the workspace using the command (clear).
- You can make Matlab save a copy of every thing you type using (diary fileName) and can stop this using (diary off)
- You can save current variables to a .mat file using (save 'filename') and then load them using (load 'filename'). You can even save specific variables using (save 'filename' v1,v2) then load them using a similar format.
- To stop a lengthy operation press <ctrl+c>
- To exit matlab type *exit*.
- Matlab is easy.

5. Operators

The most important operators in Matlab are:

=,+,-,*,/,<,>,<=,>=,==,|,& (as you expect)

% (mod) ^(power of) .*,./(element by element matrix operations) '(Transpose)

~(not) ~=(not equal locgical) &,| (and, or)

6. Variables

There are many types of variables in Matlab but we are only interested in few of them in this course. There is no need for forward declaration of variables in Matlab. Variables are defined when first used and variable names are case sensitive

A. Numeric Variables

>>a=2.5

This statement defines a variable called *a* and initializes it to the value 2.5. You can see this variable now in the workspace window.

>>b=a^2+3

This statement is OK as all the variables in the right hand side are now defined

>>b=A^2+3

This statement causes an error as there is no variable called *A* (remember Matlab is case sensitive)

>>a=[1,1.1,3]

This statement defines a variable called *a* that contains a row vector (1XN) of three values. Notice that now the previous *a* variable (that contained 2.5) no longer exists.

>>b=[1;1.1;3]

This statement defines a variable called *a* that contains a column vector(NX1) of three values.

>>c=[1,2;3,4;5,6]

This statement defines a variable called *c* that contains a 2D matrix(NXM).

Try the following statements to understand what they do (some of them will not work):

>>a=a.*2
>>a=a+b
>>a=a+b'
>>a(1)
>>a(4)
>>a(0)
>>a(1:2)
>>c(3,1)
>>c(1,3)
>>c'
>>c(1,3)
>>c=c'
>>c(1,3)
*>>d=c*3*
>>d=c'.*3
>>d(1,:)=a(2:3)
>>d=[a(1:2);b(2:3)']
>>d=[a(1:2);b (2:3)']
>>e=c(:,1:2).*d
>>f=c(:,1:2)*d

B. Strings

>>c=' some words are fatal'

This statement defines a variable called *c* which contains an array of characters and initializes it to the value 'some words are fatal'. You can see this variable now in the workspace window.

>>d='and some can heal'
>>e=strcat(c,d)

What does *strcat* do in this case?

7. M-Files

Rather than typing your commands one by one in the command window you can combine multiple commands in a file with .m extension and then run them all by typing its name. Try this out

>>edit test1.m

Now an editor is opened with the file test.m opened in it. Type the following:

g=f/2
f=g/4

Now save and go back to the command window and type:

>>test1

Notice that the variable *f* is now changed and that you have a new variable *g* in the workspace

This means that all variables in an m file are global and that all global variables in the workspace are accessible in m files.

8. Built-in Functions

Matlab has a large set of built in functions that implements most commonly used mathematical operations.

cos(x) Cosine	abs(x) Absolute value
sin(x) Sine sign(x)	Signum function
tan(x) Tangent	max(x) Maximum value
acos(x) Arc cosine	min(x) Minimum value
asin(x) Arc sine	ceil(x) Round towards +1
atan(x) Arc tangent	floor(x) Round towards -1
exp(x) Exponential	round(x) Round to nearest integer
sqrt(x) Square root	rem(x) Remainder after division
log(x) Natural logarithm	angle(x) Phase angle
log10(x) Common logarithm	conj(x) Complex conjugate

New functions are added to matlab in the form of toolboxes all the time. You can find a Matlab function somewhere in the web to do whatever you dream of usually (though I doubt you can find a function to find you a job or make this class end any faster!!)

9. Functions

You can add you own functions very easily. There are many ways to define functions in matlab.

A. *A function in a file*

>>edit myAdd.m

This opens a new m file for you. We will use this file to define a function. The function name MUST be myAdd. Function declaration in matlab has the following format:

```
function [output1, output2, …., outputm]=FunctionName(input1,input2,…..inputn)
.
.
.
End
```

FunctionName can be any name but if you want to be able to use the function outside the m file (which you usually do!!) then it must be the same as the name of the m file. Now try to define your myAdd function. In the opened m file type

```
function [r]=myAdd(a,b)
r=a+b;
end
```

Now save and go back to the command window then try:

```
>>a=myAdd(a,b')
```

Of course functions are usually longer and more useful than this toy example!

You can define variables inside functions but these (except outputs) are local and will not be seen outside it. You can have multiple functions inside a file but only the one with the same name as the file is accessible outside this files. Nevertheless, functions inside the same file can access each other.

Functions can and usually contain flow control statements like if-then, for and while. The 'end' at the end of the function is not strictly necessary but is recommended just in case you decide to add new internal functions inside the file.

B. *Inline functions*

If your function is very simple (one line of code or so) you need not create a .m file for it. You can define it in matlab directly using:

```
>>myAdd2=inline('a+b','a','b');
```

Now you can try

```
>>a=myAdd2(a,-b')
```

C. *Using @*

You can do the same using:

```
>> myAdd4=@(a,b) a+b
>>h=myAdd4(b,b)
```

D. *Evaluating using feval*

You can evaluate a function using the feval evaluator as:

```
>>f=feval('myAdd','a','a')
>>f=feval(myAdd2,'b','b')
>>f=feval(myAdd3,'b'','b'')
```

10. If-Then

The if statement in matlab looks like (italic parts are optional):

```
if condition
.
elseif condition
.
else
.
end;
```

```
>>edit test.m
```

Write the following test2.m file:

```
a = 4;b = 4;
if (a<b)
   j = -1;
elseif (a>b)
   j = 2;
else
   j = 3
end
```

Write an if statement to add a and b' if a(1) was positive and to subtract them if it was negative.

11. For

The FOR loop in matlab is used to run something a known number of times (it is not as complex as C++'s for). Its general format is

```
for variable=start:step:final
.
```

```
        end
```

Try:

```
>>for i=1:0.1:2;
        i
         end;
```

12. While

The while loop is used to execute something based on a condition:

```
        while condition

                .

        end
```

Try the following:

```
>>t=1; while t<10;
t=t+1
end
```

13. Very Very Basic Plotting

The simplest plotting function is explot that takes a function and a range and then plots it.

```
>> f=@(x) 1./x-(x-1)
>>ezplot(f,0,4)
```

What can you see? Where is the root of *f* (nearly)?

The most commonly used plotting function in matlab is plot:

```
>>x=0:0.01:1;
>>y=sin(3*pi*x);
>>plot(x,y);
```

Now add these one by one and look at the graph

```
>>title('Graph of y=sin(x)');
>>xlabel('x');
>>ylabel('y');
```

Now try the following and watch the graph after each line:

```
>>z= cos(3*pi*x);
>>plot(x,y,'r-',x,z,'b--');
>>legend('Sin curve','Cos curve');
>> hold on
>> plot(x,z./2,'g-.');
>>hold off
>> plot(x,z./3,'g-');
```

Use *doc plot* to get more information on plot and its options.

14. Assignment

- Write a function that implements matrix multiplication element by element using for loops. Your function should check that the dimensions of the input matrices are acceptable for the multiplication. The function should also return the sum of squares of the result where for a matrix A with elements a_{ij}, the sum of squares is defined as $\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}^2$.
- Use your function to multiply A and B and find the sum of squares of the result where:

$$A = \begin{bmatrix} 1 & -2.1 & 2.4 \\ 1.5 & 1 & 2.5 \\ 0 & -3 & 1 \end{bmatrix} \; and \; B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & .5 \\ 0 & 1 & -.5 \end{bmatrix}$$