Institute of Informaiton Technology
Jahangirnagar University

Experiment 1 & 2: Introduction to MATLAB

# MATLAB works with three types of windows on your computer screen. These are the Command window, the Figure window and the Editor window. The Figure window only pops up whenever you plot something. The Editor window is used for writing and editing MATLAB programs (called M-files) and can be invoked in Windows from the pull-down menu after selecting File | New | M-file.

# Create a folder of your group name in C drive.
# Open MATLAB 6.5. In "current directory" select your folder.

# From the "file" menu start a new m-file. Always write your program in m-file & save it. The file/function/variable name must start with a letter and cannot contain space. The name can be a mix of letters, digits, and underscores. (*e.g.*, vector_A, but not vector-A (since "-" is a reserved char). must *not* be longer than 31 characters.

# You can also write any command or program in "command window".
# Function "clear all" removes all variables, globals, functions and MEX links. Write clear all at the beginning of your m-file.
# Write "clc" in command window to clear the command window, "clg" to clear graphics window. # MATLAB is case sensitive. *e.g.*, NAME, Name, name are 3 distinct variables.
# Write "help function_name" in command window after the ">>" sign to see the description of the function. For example, type "help sin" to know about sin functions in MATLAB. You can also use help in the menubar.
Explore MATLAB’s lookfor and help capability by trying the following:

```
>> lookfor keywords
>> help
>> help plot
>> help ops
>> help arith
```

Special Characters:

There are a number of special **reserved** characters used in *MATLAB* for various purposes. Some of these are used as arithmetic operators, namely, +, -, *, / and \. While others perform a multitude of purposes:

- % -- anything after % (and until end of line) is treated as comments, *e.g.*,

```
>> x = 1:2:9; % x = [1 3 5 7 9];
```

- ; -- delimits statements; suppresses screen output, *e.g.,*

```
>> x = 1:2:9; y = 2:10; % two statements on the same line
```

- ... -- statement continuation, *e.g.,*

```
>> x = [ 1 3 5 ...
7 9]; % x = [1 3 5 7 9] splitted into 2 lines
```

- : -- range delimiter, *e.g.,*

```
>> x = [1:2:9]; % x=[1,3,5,7,9]
```

- ' -- matrix transposition, *e.g.,*

```
>> x = [1:2:9]'; % x changed from row vector to column vector
```

If the vector/matrix is complex, “ ’ ” results in complex conjugation ***and*** *matrix transposition.*

- , -- command delimiter, *e.g.,*

```
>> x = [1:2:9], y = [1:9] % two statements on the same line
```

- . -- precede an arithmetic operator to perform an elemental operation, instead of matrix operation, *e.g.,*

```
>> x = 3:3:9

x =

    3     6     9

>> y = 3*ones(3,1)'

y =

    3     3     3

>> z =x./y

z =

    1     2     3
```

- * -- "wild card", *e.g.*,

```
>> clear A* % clears all variables that start with A.
```

Note that many of these characters have multiple functionalities (function overloading) depending on the context, *e.g.*, "*" is used for scalar multiply, matrix multiply and "wild card" as seen above.

<u>Arithmetic Operations & Built in functions:</u>

#<u>Example1</u>. Find $y = \exp(5^2/3pi)$.
Solution: In m-file write the following command

```
clear all;
a=5^2;
b=3*pi;
y=exp(a/b);
disp(y)
```

#Save the file and "run" the program from "Debug" menu. Type "y" in command window and press "enter".
# Remove ";" from all the lines and run the program.
# Write each line in command window and press "enter" after each line.
# Variable names are assigned to expressions by using equal sign. For example, a=5^2; here "a" is the variable that store the value of 5^2 or 25.
# See the list of built in functions from "help" menu. Some built in functions are
abs() cos() sin() exp() log() real() sqrt() floor() ceil()

#<u>Exercise 1.</u> Find the value of $y = \ln(\sinh(\exp(54^3/6*pi)))$

<u>Matrices:</u>

# Write A= [1 2 3; 4 5 6; 7 6 3] in command window and press "enter". It is a 3×3 matrix.
# Write A(1,3) in command window to view the 3rd element in 1st row. The first parameter within bracket denotes row and the second parameter denotes column.
# Z=zeros(2,3) creates a 2×3 matrix of zeros. Similarly ones(), eye() create special types of matrices.
# Write A=0:0.3:3 in command window. 0 is the starting value, 3 is the end value and 0.3 is the step value.
# Write "help size", "help prod" and "help length" in command window.

# <u>Exercise 2</u>: Find the size, and length of following matrices
A=[1 2 3; 4 5 6;7 6 54; 65 23 45]
B=7:1:13.5
# Write A(1:2,2:3) in command window. Write A([1 2],[2 3]). These are different ways to select a submatrix of a matrix.

#A(1,1)=sin(5); assign a new value to an element of A.

Matrix Operations:

# All arithmetic operations can be performed on matrices.
# Operations can be performed on each element or on whole matrix. For example,

```
>> x = 3:3:9

>> y = 3*ones(3,1)'

>> z =x./y
```

# Some operations are performed on square matrices only.

```
#   +   -   *   /   ^  (algebraic/matrix definitions)
   .+   .-   .*   ./   .^  (element by element operation)
```

Additionally,

" ' " performs matrix transposition; when applied to a complex matrix, it includes elemental conjugations followed by a matrix transposition
\ and .\ perform matrix and elemental left division

#Exercise 3. A=[2 3; 4 5]; B=[3 4; 6 7];
Find A+B, A*B, A.*B,A/B,A\B, A.^2,A./B

Graphics:

# MATLAB can produce 2 and 3 dimensional plots.

MATLAB is an interactive environment in which you can program as well as visualize your computations. It includes a set of high-level graphical functions for:

- Line plots(plot, plot3, polar)
- Bar graphs(bar, barh, bar3, bar3h, hist, rose, pie, pie3)
- Surface plots(surf, surfc)
- Mesh plots(mesh, meshc, meshgrid)
- Contour plots(contour, contourc, contourf)
- Animation(moviein, movie)

#Example 2.

```
  x=0:0.1:pi;
      y=cos(x);
   plot(y);
     plot(x,cos(x),’r’);
         plot(x,y,x,y.^2);
```

#Example 3.

```
   x=0:0.1:pi;
     y=cos(x);
      plot(y);
        hold on
          plot(x,cos(x),'r');
```

#Example 4.

```
   x=linspace(0,7);
    y=exp(x);
     subplot(2,1,1), plot(x,y);
       subplot(2,1,2), semilogy(x,y);
```

# Example 5.

```
  x = magic(3);
    bar(x);
       grid
```

#Exercise 4. Plot the following functions in the same window y1=sin x, y2=sin 2x, y3=sin 3x, y4=sin 4x where x varies from 0 to pi.

Loops & Conditionals:

#MATLAB has the following flow control constructs:
• if statements
• switch statements
• for loops
• while loops
• break statements

The if, for, switch and while statements need to terminate with an end statement.
Example:

#Example 6. IF:

```
     x=-3;
if x>0
  a=10;
    elseif x<0
       a=11;
    elseif x= = 0
     a=12;
      else
        a=14;
```

```
end
```

What is the value of ‘a’ after execution of the above code?

#Example 7. WHILE:

```
  x=-10;
while x<0
   x=x+1;
end
```

What is the value of x after execution of the above loop?

#Example 8. FOR loop:

```
  x=0;
for i=1:10
   x=x+1;
end
```

What is the value of x after execution of the above loop?

Defining matrices via the vector technique

Using the for loop in MATLAB is relatively expensive. It is much more efficient to perform the same task using the vector method. For example, the following task

```
  for j=1:n

    for i=1:m

      A(i,j) = B(i,j) + C(i,j);

    end

  end
```

can be more compactly and efficiently represented (and computed) by the vector method as follows:

```
  A(1:m,1:n) = B(1:m,1:n) + C(1:m,1:n);
```

If the matrices are all of the same size (as is the case here), then the above can be more succinctly written as

```
  A = B + C;
```

For sufficiently large matrix operations, this latter method is vastly superior in performance.

#Example 9. BREAK:

The break statement lets you exit early from a for or a while loop:

```
   x=-10;
while x<0
    x=x+2;
  if x = = -2
    break;
  end
end
```

What is the value of x after execution of the above loop?

Relational Operators

Symbol	Meaning
<=	Lessthanequal
<	Less than
>=	Greater than equal
>	Greater than
==	Equal
˜ =	Not equal

Logical Operators

Symbol	Meaning
&	AND
|	OR
˜	NOT

Defining functions:

# In MATLAB there is scope for user-defined functions.

Suggestion: Since MATLAB distinguishes one function from the next by their file names, name files the same as function names to avoid confusion. Use only lowercase letter to be consistent with MATLAB's convention.

# To define a function, start a new M-file
The first line of M-file should be

```
  function  variable_name=function_name(parameters);
```

variable_name is the name of variable whose value will be returned.
function_name is user defined according to the rules stated previously.

#Example 10:

```
function y=cal_pow(x);

y=1+x^2;

end
```

# Save this function as cal_pow.
# Start another new M-file .This will be our main file.
# Write the following commands and run the file:

```
 clear all;
     x=0:1:3;
        t=length(x);

 for i=1:t
      val(i)=cal_pow(x(i));
end

plot(x,val);
```

Do the following:

# Exercise 5. Write a program to compute the variance of an array $\mathbf{x}$. The variance σ is defined to be:

$$\sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2$$

where $\overline{x}$ is the average of the array $\mathbf{x}$.

(a) For $\mathbf{x}$, use all the integers from 1 to 1000.
(b) Create $\mathbf{x}$ by built in function rand.

# Exercise 6 . Define the matrices

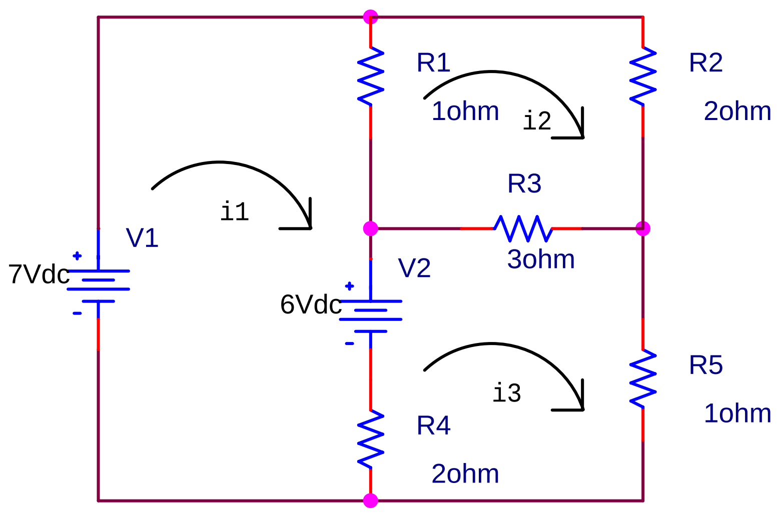
```
A=[17 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
B=[ 2 3 4 5 ; 6 7 8 9 ; 10 11 12 13 ; 14 15 16 17 ]
C=[ 1 2 3 ; 4 5 6 ; 7 8 9 ]
y=[ 4 3 2 1 ]'
```

Note the transpose ' on the y-vector which makes y a column vector.
a) Compute *AB* and *BA*. Is matrix multiplication commutative?
b) Compute *AC*. Why do you get an error message?
c) Solve the following system of equations:

$$17x_1+2x_2+3x_3+4x_4 = 4$$
$$5x_1+6x_2+7x_3+8x_4 = 3$$

$9x_1+10x_2+11x_3+12x_4 = 2$
$13x_1+14x_2+15x_3+16x_4 = 1$

# Exercise 7 . Solve the following circuit to find i1, i2, and i3.



Ans: i1= 3 amp, i2= 2 amp, i3= 3 amp.