LAB 2

Fundamental Concepts

Objective

The aim of this introductory lab is to introduce you to the basic functions in the Matlab and Numerical Methods with Matlab toolbox. By the end of today's lab, you should be able to understand the Arrays Creation, Arrays Access, the Arrays operations, Element by element on Arrays.

Submission Requirements

You are expected to complete the assigned tasks within the lab session and show them to the lab engineer/instructor. Some of these tasks are for practice purposes only while others (marked as '*Exercise*' or '*Question*') have to be answered in the form of a lab report that you need to prepare. Following guidelines will be helpful to you in carrying out the tasks and preparing the lab report.

Guidelines

- In the exercises, you have to put the output in your Lab report. You may add screen print to the report by using the 'Print Screen' command on your keyboard to get a snapshot of the displayed output. This point will become clear to you once you actually carry out the assigned tasks**.**

- Name your reports using the following convention:

  Lab#_Rank_YourFullName

  - '#' replaces the lab number
  - '*Rank*' replaces Maj/Capt/TC/NC/PC
  - '*YourFullName*' replaces your complete name.
- You need to submit the report even if you have demonstrated the exercises to the lab engineer/instructor or shown them the lab report during the lab session.

Creating Matrix/Arrays

The Matrix/Array is a fundamental form that Matlab uses to store and manipulate data. It is a list of numbers arranged in rows and/or columns. Simplest way to create matrix is to use the constructor operator [ ].

One Dimensional

It is a list of numbers that is placed in a row or a column.

Variable_name=[type vector elements **]**

Row Vector: To create a row vector type the element with a space or a comma between the elements inside square brackets.

$$row = [E_1,\ E_2,\ \ldots\ E_m] \quad or \quad row = [E_1\ E_2\ \ldots\ E_m]$$

```
>>A=[12, 62, 93, -8, 22]
```

Column Vector: To create a column vector type the element with a semicolon between the element or press the enter key after each element inside square brackets.

$$col = [E_1;\ E_2;\ \ldots\ E_m] \quad or \quad row = [E_1$$
$$E_2$$
$$\ldots\ E_m]$$

```
>>B=[12; 62; 93; -8; 22]
```

Example Date

Year	2007	2008	2009	2010
Population (Millions	136	158	178	211

Practice 1

```
>> yr=[2007 2008 2009 2010]                %Year is assigned to a row vector named yr.
yr =   2007      2008      2009      2010

>> pop=[136; 158; 178; 211]                %Population data is assigned to column
                                           vector named pop.
pop =
  136
  158
  178
  211
```

Matrix creation using : Operator (colon)

Colon operator is used to create a vector with constant spacing the difference between the elements is the same.

Variable_name=[start : step : stop**]** or start : step : stop

Where start is first term, stop is last term and step is spacing between elements. The default value of step is 1.

Practice 2

```
>> yr=[2007 : 2010]                          %Year is assigned to a row vector named yr.
yr =  2007    2008    2009    2010

>> x=[1: 2 : 13]                             %First element is 1, spacing 2 and last 13
x =   1   3   5   7   9   11   13

>> y=[1.5: 0.1 : 2.1]
y =   1.5000   1.6000   1.7000   1.8000   1.9000   2.0000   2.1000

>>  x = -pi/2:2*pi/60:pi/2;
```

Note: If stop value can not be obtained by add step's to stop then the last element is the vector with be last number that does not exceed stop.

Matrix creation using linspace function

The linspace function is used to generate a row vector of n linearly equally spaced elements.

Variable_name=linspace(start, stop, n**)**

Where start is first element, stop is last element and n is the total number of elements are created. The default value of n is 100.

Practice 3

```
>> x=linspace(0,1);                          %100 equally spaced elements are created

>> va=linspace(0,8,6)                        %6 elements, first element 0, last element 8
va =  0   1.6000   3.2000   4.8000   6.4000   8.0000

>> vb=linspace(30,10,11);

>> vc=linspace(49.5,0.5);                    %100 elements will be displayed
```

Two Dimensional

A two dimensional array/matrix has numbers in rows and columns. Matrix can be used to store information like in a table.

Variable_name=[1st row elements; 2nd row elements; ..... ;last row elements **]**

To create a matrix type the element with a semicolon between the rows or press the enter key after each row inside square brackets.

$$row = [RE_1;\ RE_2;\ \ldots\ RE_m] \quad \text{or} \quad row = [RE_1\ RE_2\ \ldots\ RE_m]$$

Practice 4

```
>> a=[5 35 43; 4 76 81; 21 32 40]          %A semicolon is typed before a new line
>> b= [ 7  2  76  33  8                    %Press enter before a new row
1  98  6  25  6
5  54  68  9  0]

>> cd=6;e=3;h=4;                           %Three variable are defined
>> Mat=[e,cd*h,cos(pi/3);h^2,sqrt(h*h/cd),14]
Mat =                                      %Elements are defined by expressions
   3.0000   24.0000    0.5000
  16.0000    1.6330   14.0000
```

Special Matrix

zeros(n) **n × n**-matrix will be created and all elements are zero.

zeros(m,n) **m × n**-matrix will be created and all elements are zero.

ones(n) **n × n**-matrix will be created and all elements are one.

ones(m,n) **m × n**-matrix will be created and all elements are one.

eye(n) **n × n**-matrix will be created and diagonal elements are one (identity matrix).

diag(V) Creates a square matrix with the elements of vector V in the diagonal

diag(M) Creates a column vector with the diagonal elements of matrix M

triu(M) Extract upper triangular part of matrix M

triul(M) Extract lower triangular part of matrix M

Practice 5

```
>> zr=zeros(3,4)
zr =
    0    0    0    0
    0    0    0    0
    0    0    0    0

>> on=ones(3,4)
on =
    1    1    1    1
    1    1    1    1
    1    1    1    1

>> idn=eye(4)
idn =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
```

```
>> A=[1 2 3;4 5 6;7 8 9]
A =     1    2    3
        4    5    6
        7    8    9

>> diag(A)
ans =   1
        5
        9
>> a=[4          5          6]
>> diag(a)
ans =   4    0    0
        0    5    0
        0    0    6
>> triu(A)
ans =   1    2    3
        0    5    6
        0    0    9

>> tril(A)
ans =   1    0    0
        4    5    0
        7    8    9
```

The Transpose Operator (‘)

It is use to switch a row or column vector to a column or row respectively. When it is applied on matrix, it swithes the rows or columns to columns or rows respectively. It is applied by typing a single quote (‘) following the variable.

Practice 6

```
>> aa=[3  8  1];                          %Define a row vector aa.
>> bb=aa'                                 %Define vector bb as the transpose of aa
bb=     3
        8
        1
>>C=[2  55  14  8;  21  5  32  11;  41 64  9   1];     %Define a matrix C with order 3x4
>> D=C'                                   %Define matrix D as transpose of C.
D=      2       21      41
        55      5       64
        14      32      9
        8       11      1
```

Array Addressing

Elements in an array (either vector or matrix) can be addressed individually or in subgroups.

Vector Addressing

- The address of an element in a vector is its position in the row or column.
- A vector named ve, ve(k) refers to the element in position k.
- The first position is 1.
- You can change the value of any element by address like this v(k)=new value.

Practice 7

```
>> VCT=[35  46  78  23  5  14  81  3  55];      %Define a vector
>> VCT(4)                                       %Display the fourth element
ans=    23

>> VCT(6)=273                                   %Assign a new value to the sixth element.
VCT=    35  46  78  23  5  273  81  3  55

>> VCT(2)+VCT(8)                                %Use of vector element in expressions
ans=    49

>> VCT(5)^VCT(8)+sqrt(VCT(7))                   %Use of vector element in expressions
ans=    134
```

Matrix Addressing

- The address of an element in a matrix is its position defined by the row and column.
- A matrix named ma, ma(k,p) refers to the element in row k and column p.
- The first row and column position is 1.
- You can change the value of any element by address like this ma(k,p)=new value.

Practice 8

```
>> MAT=[3 11 6 5; 4 7 10 2; 13 9 0 8];      %Create a 3x4 matrix

>> MAT(3,1)                                  %Display the first element in third row
ans=    13

>> MAT(3,1)=20                               %Assign a new value to (3,1) element.
MAT=  3      11      6      5
      4      7       10     2
      20     9       0      8

>> MAT(2,4)- MAT(1,2)                        %Use of vector element in expressions
ans=    -9
```

Use of : (colon) in vector addressing

A colon can be used to address the range of elements in vector

ve(:) Refers to all the element of the row or column vector ve

ve(m:n) Refers to elements m through n of the vector ve

Practice 9

```
>> ve=[4 15 8 12 34 2 50 23 11];             %A vector ve is created

>> u=ve(3:7)                                 %A vector u is created from ve(3 through 7)
u=      8 12 34 2 50
```

Use of : (colon) in matrix addressing

A colon can be used to address the range of elements in matrix

ma(:,n) Refers to the elements in all the rows of column n of the matrix ma

ma(m,:) Refers to the elements in all the columns of row m of the matrix ma

ma(:,p:q) Refers to the elements in all the rows between column p and q of matrix ma

ma(m:n,:) Refers to the elements in all the columns between row m and n of matrix ma

ma(m:n,p:q) Refers to the elements in rows m to n and columns p to q of matrix ma

Practice 10

```
>> ma=[1 3 5 7 9 11; 2 4 6 8 10 12; 3 6 9 12 15 18; 4 8 12 16 20 24; 5 10 15 20 25 30];

>> B=ma(:,3)                    %Define a column vector B from elements of 3 column
B=      5
        6
        9
        12
        15

>> C=ma(2,:)                    %Define a row vector C from elements of 2 row.
C=      2  4  6  8  10  12

>> E=ma(2:4,:)                  %Define a matrix E from the element of row 2 to 4
E=      2  4  6   8   10  12
        3  6  9   12  15  18
        4  8  12  16  20  24

>> F=ma(1:3,2:4)                %F from the element of row 1 to 3 and column 2 to 4
E=      3  5  7
        4  6  8
        6  9  12
```

Important

```
>> v=4:3:34;

>>u=v([3,  5,  7:10])           % u is created with 3rd, the 5th and 7th to 10th element of v
u=      10  16  22  25  28  31
```

Adding element to existing array

A variable can be changed by adding elements to it .

A vector can be changed to have more elements or it can be changed to be a tow-dimensional

The addition of elements can be done by simply assigning values to the additional elements.

Adding elements to vector

- Elements can be added by assigning values to the non-existing positions of vector
- If a vector has n elements and a new value is assigned to an element with address of n+2 or larger, Matlab assigns zeros to the elements that are between the last orignal elements and new element
- Elements can also be added to a vector by appending existing vectors.

Practice 11

```
>> DF=1:4;

>>DF(5:10)=10:5:35                    %Adding 6 elements starting with the 5th
DF=    1  2  3  4  10  15  20  25  30  35

>> AD=[5  7  2];

>> AD(8)=4                            %Matlab assigns zeros to the 4th through 7 elements.
AD=    5  7  2  0  0  0  0  4

>> AR(5)=24                           %{Assign a value to the 5th element of new vector and
                                       Matlab assigns zeros to 1st to 4th elements}%
AR=    0  0  0  0  24

>> RE=[3  8  1  24];
>>GT=4:3:16;
>>KNH=[RE  GT]                        %Define a new vector KNH by appending RE and GT.
KNH=  3  8  1  24  4  7  10  13  16
```

Adding elements to matrix

- Rows and/or columns can be added to matrix by assigning values to new rows or columns
- Can be done by assigning new values or by appending existing variables.
- If a matrix has a size of m x n, and a new value is assigned to an element with new address beyond the size of the matrix, Matlab increases the size of the matrix to include the new element. Zeros are assigned to the other elements that are added.

Note: The size of the added rows or columns must fit the existing matrix.

Practice 12

```
>> E=[1  2  3  4;  5  6  7  8];
>>E(3,:)=[10:4:22]                    %Add the vector 10 14 18 22 as the third row of E
E=      1   2   3   4
        5   6   7   8
        10  14  18  22

>>K=eye(3);                           %Identity matrix of 3x3 is created

>>G=[E  K]
E=      1   2   3   4   1 0 0
        5   6   7   8   0 1 0
        10  14  18  22  0 0 1

>>AW=[3  6  9;  8  5  11];
>>AW(4,5)=17;          %Matlab changes the matrix size to 4x5 and assigns zeros to new elements
>>BG(3,4)=15;          %Matlab creates 3x4 matrix and assigns zeros to elements except BG(3,4)
```

Deleting Elements

An element or a rang of elements, of an existing variable can be deleted by reassigning nothing to these element.

This can be done by using square brackets with nothing typed in between them.

Practice 13

```
>> kt=[2 8 40 65 3 55 23  15  75  80];

>> kt(6)=[]                                  %Eliminate the sixth element.
Kt=      2 8 40 65 3 23  15  75  80

>> kt(3:6)=[]                                %Eliminate elements 3 to 6
Kt=      2 8 15  75  80

>> mtr=[5  78  4  24  9;  4  0  36  60  12;  56  13  5  89  3];

>> mtr(:,2:4)=[]                             %Eliminate all the rows of columns 2 to 4
mtr=    5        9
        4        12
        56       3
```

Arrays Built in Function

Function	**Description**	**Example**
length(A)	Returns the number of elements in the vector A.	>>A=[5 9 2 4]; >>length(A) ans=4
size(A)	Returns a row vector [m,n], where m and n are the size m x n of array A	>>A=[6 1 4 0 12; 5 19 6 8 2]; >>size(A) ans=2 5
reshape(A,m,n)	Rearrange a matrix A that has r rows and s columns to have m rows and n columns. r times s must be equal to m times n.	>>A=[5 1 6; 8 0 2]; >>B=reshape(A,3,2) B= 5 0 8 6 1 2

Strings and Strings as variable

- A string is an array of characters. It is created by typing the characters within single quotes
- String can include letters, digits, other symbols and spaces.
- A string that contains a single quote is created by typing two single quotes within string.
- Matlab has built in function named char that creates an array with rows that have the same number of characters, length all the rows equal to the longest row by adding spaces at the end of short lines.

Variable_name= char('string1','string2','string3', ...)

Practice 14

```
>> B='My name is Shahid'
B= My name is Shahid

>> B(4)
ans=n

>> B(12)
ans=S

>> B(12:17)='Khalid'
B= My name is Khalid

>>Info=char('Student Name:','Shahid','Grade:','A+');     %Variable Info is of different strings
```

Note: Function char creates an array with four rows with the same lenght as the longes row by adding empty spaces to the shorter lines.

Addition and Subtraction

- The operations addition and subtraction can be used with array of identical (Same order) size.
- The sum or difference of two array is obtained by adding or subtracting their corresponding elements.
- When a scalar is added to or subtracted from an array the number is add or subtracted from all the elements of the array.

Practice 15

```
>> VectA=[8  5  4];

>> VectB=[10  2  7];

>> VectC= VectA+VectB
VectC= 18  7  11
>> A=[5  -3  8; 9  2  10];

>> B=[10  7  4;  -11  15  1];

>> C=A+B
C=      15      4      12
        -2      17     11

>> D=A-B
D=      -5      -10    4
        20      -13    9


>> C-8
ans=    7       -4     4
        -10     9      3
```

Array Multiplication

- Multiplication operation is executed by Matlab according to the rules of linear algebra.
- If A and B are two matrices, the operation A*B can be carried out only if the number of columns in matrix A is equal to the number of rows in matrix B.
- If A and B are both nxn then A*B not equal to B*A
- Power operation can only be executed with a square matrix
- Two vectors can be multiplied if both have the same number of elements, and one is a row vector and the other is column vector.
- An array is multiplied by a number then each element in the array is multiplied by the number

Practice 16

```
>> A=[1 4 2; 5 7 3; 9 1 6; 4 2 8];

>> B=[6 1; 2 5; 7 3];

>> C= A*B
C=      28  27
        65  49
        98  32
        84  38

>> D=B*A;
???Error using ➔*
Inner matrix dimensions must agree.

>> F=[1 3; 5 7];

>> G=[4 2; 1 6];

>> F*G

ans=    7       20
        27      52


>> G*F

ans=    14      26
        31      45

>> A=[2 5 7 0; 10 1 3 4; 6 2 11 5];

>>b=3;

>>C=b*A
C=      6       15      21      0
        30      3       9       12
        18      6       33      15
```

Array Division

- The division operation can be done with the help of the identity matrix and the inverse operation.
- eye() is used to create Identity matrix, if a matrix A is square, it can be multiplied by the identity matrix I, from the left or from the right. (A*I = I*A = I)
- inv() is used to find inverse of matrix. The matrix B is the inverse of the matrix A if when the two matrices are multiplied the product is the identity matrix.($B*A = A*B = I, B=A^{-1}$)
- det() is used to find determinant of matrix. Not every matrix has an inverse. A matrix has an inverse only if it is square and its determinant is not equal to zero.

Practice 17

```
>> A=[2 1 4; 4 1 8; 2 -1 3];

>>det(A);                          %|A| is not equal to zero

>> B=inv(A);                        %Use the inv() to find the inverse of A.

>> A*B                             %Multiplication of A and B gives the identity matrix.
ans=    1       0       0
        0       1       0
        0       0       1

>> A*A^-1                          %Use of power of -1 to find the inverse of A
ans=    1       0       0
        0       1       0
        0       0       1
```

Element by element operation

- When operation applied on each element is said to be element by element. Addition and subtraction are by definition already element by element operations but multiplication, division and exponential are done by using . (dot) operator.
- This operation is applied on two arrays then must be the same size of both arrays but one of them can be a scalar.
- If two vectors a and b are: a=[a_1 a_2 a_3] and b=[b_1 b_2 b_3] then element by element of the two vectors as follows

a.*b = [a_1*b_1 a_2*b_2 a_3*b_3] a./b = [a_1/b_1 a_2/b_2 a_3/b_3] a.^b = [a_1^b_1 a_2^b_2 a_3^b_3]

Symbol	**Description**	**Symbol**	**Description**
.*	Multiplication	./	Right division
.^	Exponential	.\	Left division

Practice 18

```
>> A=[2 6 3; 5 8 4];

>>B=[1 4 10; 3 2 7];

>> A.*B                            %Element by element multiplication of array.
ans=    2       24      30
        15      16      28

>> C=A./B;                         % Element by element division of array

>> B.^3;                           %Element by element exponentional of array B.
```

Practice 19

Function $y=(x^2+1)^3 x^3$, calculate the value of y for the following values of x

-2.5 -2 -1.5 -1 -0.5 1 1.5 2 2.5 3

Built in function for analyzing Arrays

Function	**Description**	**Example**
mean(A)	If A is a vector, returns the mean value of the elements of the vector	>>A=[5 9 2 4]; >>mean(A) ans=5
C=max(A)	If A is a vector, C is the largest element in A. If A is a matrix, C is a row vector containg the largest element of each column of A.	>>A=[5 9 2 4 11 6 7 11 0 1]; >>C=max(A) ans=11
[d,n]=max(A)	If A is a vector, d is the largest element in A, n is the position of the element	>>[d,n]=max(A) d=11 n=5
C=min(A)	The same as max(A), but for the smallest element.	>>A=[5 9 2 4]; >>C=min(A) C=2
[d,n]=min(A)	The same as [d,n]=max(A), but for the smallest element.	
C=sum(A)	If A is a vector, returns the sum of the element of the vector	>>A=[5 9 2 4]; >>C=sum(A) C=2
sort(A)	If A is a vector, arranges the element of the vector in ascending order	>>A=[5 9 2 4]; >>sort(A) ans=2 4 5 9
C=median(A)	If A is a vector, returns the median value of the elements of the vector	>>A=[5 9 2 4]; >>C=median(A) C=4.5000
C=std(A)	If A is a vector, returns the standard deviation of the elements of the vector.	>>A=[5 9 2 4]; >>C=std(A) C=2.9439
rand	Generates a single random number between 0 and 1	>>rand Ans=0.2311
rand(1,n)	Generates an n elements row vector of random numbers between 0 and 1	>>a=rand(1,4);
rand(n)	Generates an n x n matrix with random numbers between 0 and 1	>>b=rand(3);
rand(m,n)	Generates an m x n matrix with random numbers between 0 and 1	>>c=rand(2,4)
Randperm(n)	Genereates a row vector with n elements that are random permutation of integers 1 to n	>>randperm(8) Ans= 8 2 7 4 3 6 5 1

Exercise 1

Use Matlab to show that the sum of the infinite series $sum(1/n^2)$ converges to $pi^2/6$. Do it by computing the sum for

a) n=100
b) n=1000
c) n=10000
d) n=100000

Exercise 2

Solve the following system of foure linear equations

$$5x+4y-2z+6w=4$$

$$3x+6y+6z+4.5w=13.5$$

$$6x+12y-2z+16w=20$$

$$4x-2y+2z-4w=6$$

Graphics

MATLAB produce two and three dimensional plots of curves and surface

Two Dimensional Plots

- Plots are very useful tool for presenting information.
- Plot title, Legend, X axis label, Y axis label, Text label and Markers.
- Graph is shown in figure window.

Plot Command

The plot command is used to create two dim plots.

The simplest from of the command is plot(x,y), Where x (horizental axis) and y (vertical axis) are vector.

Both vectors must have the same number of elements.

The curve is constructed of straight line segments that connect the points.

The figure that is created has axes with linear scale and default range.

Practice 20

```
>> x=[1 2 3 5 7 7.5 8 10];
>>y=[2 6.5 7 7 5.5 4 6 8];
>>plot(x,y)
```

plot(x,y,'line specifiers','PropertyName', PropertyValue)

Line Specifiers:

- Line specifiers are optional and can be used to define the style and coloar of the line and the type of markers.
- Within the string the specifiers can be typed in any order.
- Specifier is optional i.e. None, one, two, or all the three can be included in command.

Line Style	Specifier	Line Color	Specifier	Marker Type	Specifier
solid(default)	-	red	r	plus sign	+
dashed	--	green	g	circle	o
dotted	:	blue	b	asterisk	*
dash-dot	-.	cyan	c	point	.
		megenta	m	square	s
		yellow	y	diamond	d
		black	k		
		white	w		

Property Name and Property Value:
Properties are optional and can be used to specify the thickness of the line, the size of marker, and color of marker's edge line and fill.
The property name is typed as a string, followed by comma and value for property.

Property Name	Description	Possible Property Values
LineWidth or linewidth	Specifies the width of the line	A number in units of points (default 0.5).
MarkerSize or markersize	Specifies the size of marker	A number in units of points.
MarkerEdgeColor or markeredgecolor	Specifies the color of the marker, or the color of the edge line for filled markers.	Color specifiers from the table above, typed as a string
MarkerFaceColor or markerfacecolor	Specifies the color of the filling for filled markers.	Color specifiers from the table above, typed as a string

Practice 21

```
>>plot(x,y,'r');
>>plot(x,y,'--y');
>>plot(x,y,'*');
>>plot(x,y,'g:d');
>>plot(x,y,'-mo','LineWidth',2,'markersize',12,'MarkerEdgeColor','g','markerfacecolor','y');
```

```
>>yr=[1988:1:1994];                    %Sales Years
>>sle=[8  12  20  22  18  24  27];     %Sales in millions
>>plot(yr,sle,'—r*','linewidth',2,'markersize',12);
```

Plot of function
Plot of given functgion can be done in Matlab by using the plot or the fplot commands.

Using plot: Plot a function y=f(x) with plot command, user first needs to create a vector of values of x for the domain that function will be plotted.
Then a vector y is created with the corresponding values of f(x) by using element by element calculation
Plot the x and y vectors by using plot command.

Using fplot: Plot a function y=f(x) with fplot command between specified limits.

fplot('function',limits, n, line specifiers)

function can be typed directly as a string and The function can not inclue prefiouly defined variables.
limits is a vector with two element that specify the domain of x [xmin,xmax] or [xmin,xmax,ymin,ymax].
n is the number of points between interval.
Line specifier are the same as in plot command.

Practice 22

Examples: $y=3.5^{-0.5x}\cos(6x)$ for -2<=x<=4 and y=cos(x), z=cos(x)2 for 0<=x<=pi

```
>>x=[-2:0.01:4];                      %Create vector x with domain of function
>>y=3.5.^(-0.5*x).*cos(6*x);          %Create vector y with funtion value at each x
>>plot(x,y);
%Two or more graphs can be created in the same plot by typing pairs of vectros.
>>x=0:0.1:pi;                         % specifies the domain
>>y=cos(x);                           % define 1st function
>>z=cos(x).^2;                        % define 2nd function
>>plot(x,y,x,z,'o')                   % display graph. (x, y 1st fun , x, y 2nd fun )
```

Examples: plot of function y=3x3-26x+10, and its first and second derivatives for -2<=x<=4

```
>>x=[-2:0.01:4];
>>y=3*x.^3-26*x+6;
>>yd=9*x.^2-26;
>>ydd=18*x;
>>plot(x,y,'-b',x,yd,'—r',x,ydd,':k');

>>fplot('x^2+4*sin(2*x)-1',[-3 3]);
>>fplot('tanh',[-2,2])                 % plots y=tanh(x) over [-2,2]
```

Using the hold on, hold off commands

Hold on command keeps the figure window with first plot open, including the axis properties and formatting if any was done.
Additional graphs can be added with plot command that are typed next.
Hold off command stops this process.

Practice 23

Examples: plot of function y=3x3-26x+10, and its first and second derivatives for -2<=x<=4

```
>>x=[-2:0.01:4];
>>y=3*x.^3-26*x+6;
>>yd=9*x.^2-26;
>>ydd=18*x;
>>plot(x,y,'-b');
>>hold on
>>plot(x,yd,'—r');
>>plot(x,ydd,':k');
>>hold off
```

Using the line command

With the line command additional graphs (lines) can be added to a plot that already exists.

line(x,y,'PropertyName',PropertyValue)

The line is almost same as plot except the line specifier, but the line style, color and marker can be specified as property name and property value features.

Practice 24

Examples: plot of function y=3x3-26x+10, and its first and second derivatives for -2<=x<=4

```
>>x=[-2:0.01:4];
>>y=3*x.^3-26*x+6;
>>yd=9*x.^2-26;
>>ydd=18*x;
>>plot(x,y,'LineStyle','-','color','b');
>>line(x,yd,'LineStyle','--','color','r');
>>line(x,ydd,'LineStyle',':','color','k');
%Example  Plot x and y on graph using linspace
>>x=linspace(0,1);                    % Row vector with 100 points
>>y=x.^n.*exp(x);                     % Gives row vector
>>plot(x,y,'o')                       %or     >>plot(x, x.^n.*exp(x))
%Example Plot ellipse c(t)=(2cos(t),3sin(t)), where 0<=t<=2pi
>>t=0:0.2:2*pi;
>>plot(2*cos(t),3*sin(t));
```

Formatting a Plot

The plot needs to be formatted to have a specific look and to display information in addition to the graph itself i.e. axis labels, plot title, legend, grid, range of custom axis, and text labels. Formatting can be done by using Commands and Editior

Formatting plot using Commands
It is useful when a plot command is a part of a computer program.
Formatting commands are entered after the plot or fplot commands.

xlabel('text as string')	%placed next to the x-axis
ylabel('text as string')	%placed next to the y-axis
title('text as string')	%placed at the top of figure as title
text(x,y,'text as string')	%placed in the figure such that first character is at point x,y
gtext('text as string')	%placed in figure at position specified by the user with mouse

The legend command
The leged shows a sample of line type of each graph that is plotted, and place a label, specified by the user.

legend('string1','string2', ... , pos)

where pos is an optional number that specifies where the figure placed.

pos=-1	places the ouside the axes boundries on the right side
pos=0	places the inside the axes boundries in a location that less interfers with graph
pos=1	places the upper-right corner of the plot(default)
pos=2	places the upper-left corner of the plot
pos=3	places the lower-left corner of the plot
pos=4	places the lower-right corner of the plot.

The axis command
Matlab creates axes with min and max limits values, but axis command can be used to change the range and the appearance of the axes.

axis([xmin,xmax])	%Sets the limits of the x axis
axis([xmin,xmax,ymin,ymax])	%Sets the limits of both x and y axes
axis equal	%Sets the same scale of both axes

The grid command

grid on	%adds grid lines to the plot
grid off	%removes grid lines from the plot

Exercise 3

Create a 6 x6 matrix in which the middle two rows and the middle two columns are 1's and the rest are 0's

Exercise 4

Using the eye command create the array A of order 7 x 7. Then using the colon to address the elements in the array change the array as show bellow

$$A=\begin{matrix} 2 & 2 & 2 & 0 & 5 & 5 & 5 \\ 2 & 2 & 2 & 0 & 5 & 5 & 5 \\ 3 & 3 & 3 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 4 & 7 & 0 & 9 & 9 & 9 \\ 4 & 4 & 7 & 0 & 9 & 9 & 9 \\ 4 & 4 & 7 & 0 & 9 & 9 & 9 \end{matrix}$$

Exercise 5

Create a vector, name it Afirst, that has 16 equally spaced elements. First element is 4 and last element is 49. Create a new vector, call it Asecond that has eight elements. The first four elements are the first four elements of the vector Afirst, and the last four are the last four elements of the vector Afirst.

Exercise 6

Show that $\lim_{n \to \infty} \left(1+\frac{1}{n}\right)^n = e$

Do this by first creating a vector n that has the elements:1 10 100 500 1000 2000 4000 and 8000. Then create a new vector y in which each element is determined from the elements of n by $\left(1+\frac{1}{n}\right)^n$

Compare the elements of y with the value of e (type exp(1) to obtain the value of e).