LAB 2

Spring 2011, BESE- 15 A&B

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.

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 includet 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.

Exercise A

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 B

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

```
      2 2 2 0 5 5 5
      2 2 2 0 5 5 5
      3 3 3 0 5 5 5
A=    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
```

Exercise 1

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 2

Create a 5 x 7 matrix in which the first row are the numbers 1 2 3 4 5 6 7, the second row are the numbers 8 9 10 11 12 13 14, the third row are the numbers 15 to 21 and so on. From this matrix create a new 3x4 matrix that is made from row 2 to 4 and columns 3 to 6 of the first matrix.