Unit - II

chapter 3 :- Interactive computations

Matrices and vectors :-

Input :-

A matrix is entered row wise with consecutive elements of row separated by space or comma and the rows are separated by semicolon (or) carriage return (space bar). The entire matrix must be enclosed in a square bracket.

Example :-

Matrix

$$A = \begin{bmatrix} 1 & 2 & 5 \\ 3 & 9 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 2x & \ln x + \sin y \\ 5i & 3+2i \end{bmatrix}$$

Matlab Input :-

```
>> A = [1 2 5; 3 9 0]
        (or)
>> A = [1,2,5; 3,9,0]
>> B = [2*x log(x)+sin(y); 5i 3+2i]
>> B = [2*x log(x)+sin(y);
        5i   3+2i ]
```

Special cases :- Vectors and scalars

Vector :-

A vector is a special case of a matrix with just one row or one column. It is entered as the same way as the matrix

Example :-

```
u = [1 3 9]
```

it produces a single row (or) row vector

```
v = [1; 3; 9]
v =  1
     3
     9   ]
```

it produces a single column. (or) column vector

scalar :-

A scalar does not need brackets

Example :-

```
g = 9.81;
```

Null matrix :-

A square bracket with no elements between the

Create a null matrix

Example :- x = []

Continuation (Ellipsis)

If it is not possible to type the entire input on the same line, then use the three consecutive periods (...) to signal the continuation and continue the input on the next line. The three periods are called Ellipsis.

Example :-

```
A = [1/3   5.55*sin(x)   9.35   0.099;...
     3/(x+2*log(x))      3      0   6.555;...
     ((5*x)-23)/55   x-3   x*sin(x)   sqrt(3)];
```

A matrix can also be entered across multiple lines using carriage return at the end of each row. In the case of semicolon and ellipsis at the end of each row may be omitted. The following three commands are equivalent

```
>> A = [1 3 9; 5 10 15; 0 0 -5];
   A =
        1  3  9
        5 10 15
        0  0 -5

>> [1 3 9                    A =
    5 10 15                       1  3  9
    0  0 -5];                     5 10 15
                                  0  0 -5
>> A = [1 3 9; 5 10...
        15; 0 0 -5];
   A =
        1  3  9
        5 10 15
        0  0 -5
```

[Indexing] (or) Subscripting :- (5M|5)

[The elements of a matrix are accessed by specifying their rows and column indices. Thus A(i,j) in MATLAB refers to the element a_{ij} of a matrix A. (ie) the element in the ith row and jth column.

The indices allows a range of rows and columns to be specified at the same time.

Example :-

A(m:n, k:l) specifies rows m to n and column k to l of a matrix A. The colon can be used as the row (or) column index. Thus A(:, 5:20) refers to the elements in column 5 through 20 of all the rows of the matrix A.]

Dimension :-

Matrix dimensions are determined automatically by MATLAB. ie) No dimension declarations are required. The dimension of a existing matrix A can be obtained by

```
>> size(A) (or)
>> [m,n] = size(A);
```

which assigns the number of rows and columns of a matrix A

Example :- If B and C matrix do not exist already

```
>> B(2,3) = 5;
```

produces B =

```
0 0 0
0 0 5
```

```
>> C(3,1:3) = [1 2 3];
```

produces C =

```
0 0 0
0 0 0
1 2 3
```

20-01-2020 (Mon)

Logical (0 -1) in Matrix Index : (Binary Numbers)

~~The~~ only nonzero positive integers are legal index entries for a matrix. The only exception is logical zero (0). You can use the vector made up of zero's and one's in a matrix index if

i) The vectors are produced by logical (or) Relational operators.

ii) The 0-1 vectors created by converting (you is ed) into a logical array with a command logical.

Example :-

To get the 1st, 4th and 5th rows of Q in 0-1 vectors, is

```
v = [1 0 0 1 1]' ;
```

or `v = logical(v);`

Reshaping matrices

Matrices can be reshaped into vectors (or) any other ~~approximately~~ (priately) sized matrix

As a vector :-

All the elements of the matrix A can be strung into a single column vector b by the command

```
b = A(:)
```

(Matrix A is strucked into b column wise)

As a differently sized matrix :-

If the matrix A is a m x n matrix, it can be reshaped into a p x q matrix as long as m x n = p x q with the commands.

MATLAB command	Meaning
reshape (A, p, q)	For example A is a 6 x 6 matrix
reshape (A, 9, 4)	transforms into 9 x 4 matrix
reshape (A, 3, 12)	transforms into a 3 x 12 matrix

Transpose :-

Transpose of a matrix A is obtained by typing A' ie) the name of the matrix followed by single right quote.

Example :-

If $A = \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix}$ then $B = A'$ gives $B = \begin{bmatrix} 2 & 6 \\ 3 & 7 \end{bmatrix}$

If $C = \begin{bmatrix} 2 & 3+1i \\ 6i & 7i \end{bmatrix}$ then $C_t = C'$ gives

$C_t = \begin{bmatrix} 2 & -6i \\ 3-1i & -7i \end{bmatrix}$ (In C_t, t is for complex conjugate)

If $u = [0 \quad 1 \quad 2 \quad \ldots \quad 9]$

then $V = (3:6)'$

gives $V = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$

[0 1 2 3 4 5 6 7 8 9]
(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)
V.

(3:6)' means 3 - means start with 3rd element
6 - means ~~start~~ end with 6th number.

Matrix Initialisation :-

Initialisation of a matrix is not necessary in MATLAB
Large Matrices :- It is advisable in the following 2 cases:

To generate (or) manipulate a ~~given~~ large matrix initialize it the matrix to zero matrix of the required dimension. An mxn matrix can be initialised by the command

A = zeros(m, n)

A = zeros(5, 5)

Initialization reserves the number of memory location (or) block memory in the computer memory.

Dynamic matrices :-

If the rows or columns of a matrix are computed in ~~group~~ loop (for example while

and appended (added) to the matrix in each of the execution of the loop then we want to initialize the matrix to a null matrix before the loop starts:

A null matrix 'A' is created by the command A = [ ]. Once created, a row or column of any size may be appended to A ~~as described~~.

21-01-2020 Tue

Appending a row or column :-

(Adding / including row or column in a matrix)

A row/column can be easily appended to a existing matrix provided the row or column has the same length as the length of row or column of the existing matrix.

The command A = [A; u] add a row vector u to the row of A. Similarly A = [A v] appends a column vector v to the column of A

Example :-

If $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $u = [5 \ 6 \ 7]$ and $v = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$

Then.

$A_1 = [A; u]$ produces $A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 5 & 6 & 7 \end{bmatrix}$, a 4x3 matrix

$A_2 = [A \ v]$ produces $A_2 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 4 \end{bmatrix}$, a 3x4 matrix

$A_3 = [A \ u']$ produces $A_3 = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 7 \end{bmatrix}$, a 3x4 matrix

3.2 Matrix and Array operations :-

(1) 10M(S) (i) [Arithmetic operations :-]

MATLAB allows ~~All the~~ mathematical operations namely

+, -, addition, subtraction; *, multiplication; /, division; ^ (caret) exponentiation to be ~~are~~ carried out on matrices

in straight forward ways. For example, if A and B are any two matrices of same size, the following operations are compatible

A+B or A-B is valid if A and B are same size

A*B if A's no. of column same as B's no. of row

A/B is valid and equals A·B⁻¹ for same size square matrix

A^2 A*A makes sense only if A is square matrix

The left Division :-

The left division (\) in MATLAB is used to solve the matrix equation Ax = b

ie) x = A\b = inv(A) * b

5\3 = 3/5 = 3×5⁻¹ = 0.6

Array operations :-

The Array operations are done on an element by element basis. ie) element by element multiplication, division and exponentiation between two matrices or vectors of same size are done by preceding the corresponding arithmetic operators by a period. ie).

- .* element by element multiplication
- .\ element by element left division
- ./ element by element right division
- .^ element by element exponentiation
- .' non conjugated transpose

Example :-

u.*v produces $[u_1v_1 \; u_2v_2 \; u_3v_3 \ldots u_nv_n]$

u./v produces $[u_1/v_1 \; u_2/v_2 \; u_3/v_3 \ldots u_n/v_n]$

u.^v produces $[u_1^{v_1} \; u_2^{v_2} \; u_3^{v_3} \ldots u_n^{v_n}]$

Relational operations :-

There are six relational operators used in MATLAB. These operations result in a vector

matrix of same size as the operand with 1 where the relation is true and 0 where it is false. The relational operators are

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal
~=	not equal

5M(S) [Example :-

If $x = [1\ 5\ 3\ 7]$ and $y = [0\ 2\ 8\ 7]$ then $K = x < y$ results in $K = [0\ 0\ 1\ 0]$ because $x_i < y_i$ for $i = 3$

$K = x <= y$ results in $K = [0\ 0\ 1\ 1]$ because $x_i <= y_i$ for $i = 3, 4$

$K = x > y$ result in $K = [1\ 1\ 0\ 0]$ because $x_i > y_i$ for $i = 1, 2$

$K = x >= y$ results in $K = [1\ 1\ 0\ 1]$ because $x_i >= y_i$ for $i = 1, 2, 4$

$K = x == y$ results in $K = [0\ 0\ 0\ 1]$ because $x_4 = y_4$

$K = x \neq y$ results in $K = [1\ 1\ 1\ 0]$ because $x_i \neq y_i$ for $i = 1, 2, 3$] 5M(S)

(3)

23-01-2020 (Thurs) 10M(S) (iii)

[Logical operations :-]

The Logical operators are

&	logical AND
|	logical OR
~	logical complement (NOT)
xor	exclusive OR

These operators are worked in a similar way as the relational operators and produces vector or matrices of the same size as the operand with one 1 when the condition is true

and 0 where the condition is false.

Example :- vectors $x = [0\ 5\ 3\ 7]$ and $y = [0\ 2\ 8\ 7]$ then $m = (x>y) \& (x>4)$ result in $m = [0\ 1\ 0\ 0]$ because the condition is true for x_2

$n = x/y$ result in $n = [0\ 1\ 1\ 1]$ because either x_i or y_i is non-zero for $i = 2, 3, 4$

$m \cong (x/y)$ result in $m = [1\ 0\ 0\ 0]$ because logical complement of x/y

$p = xor(x, y)$ result in $p = [0\ 0\ 0\ 0]$ because there is no such index i for which x_i or y_i, but not both, is non-zero] 5M (S)

In addition to the logical operators there are many useful built-in logical functions, such as

all → true (= 1) if all the conditions (elements) of a vector are true

Example :- all(x<0) returns 1 if all the elements of x

any - true (= 1) if any elements of the vectors are true.

Example :- any(x) returns 1 if any elements of x is non-zero

exist - true (= 1) if the arguments exist.

is empty - true (= 1) for an empty matrix

Elementary math functions :-

The following built-in math functions take matrix inputs and perform array operations (element by element) on them. Thus they produces output matrix of the same size as the input matrix

Trigonometric functions :-

sin	sind - sine	sinh - hyperbolic sine
cos	cosd - cosine	cosh - hyperbolic cosine
asin	asind - inverse sine	asinh - inverse hyperbolic sine
acos	acosd - inverse cosine	acosh - inverse hyperbolic cosine
tan	tand - tangent	tanh - hyperbolic tangent.

etc.

Example :-

If q = [0 pi/2 pi], x = [1 -1 -1 1] and y = [1 1 -1 -1]

then sin(q) gives [0 1 0]

sinh(q) gives [0 2.3013 11.5489]

atan(y./x) gives [0.7854 -0.7854 0.7854 -0.7854]

atan2(y/x) gives [0.7854 2.3562 -2.3562 -0.7854]

Exponential function :-

exp - exponential

log - natural logarithm

log_{10} - base 10 logarithm

sqrt - square root

nth root - real nth root of real numbers

Complex functions :-

abs - absolute value, {(abs(A) produces absolute value of matrix A)}

angle - phase ~~value~~ angle, {(angle(A) gives the phase angle of complex A)}

complex - construct complex numbers with Re (real) and Img (imaginary) parts
(complex(A,B) produces A+iB)

conj - complex conjugate {(conj(A) produces an element of matrix A in conjugate form i.e. $\overline{A}_{ij}$)}

imag - imaginary part of complex number
(imag(A) produces imaginary part of A)

real - real part of complex number
(real(A) produces real part of A)

Round-off functions :-

fix - round towards zero
Eg :- fix ([-2·33, 2·66]) = [-2, 2]
floor - round towards $-\infty$
Eg :- floor ([2·33 2·66]) = [-3 2]
ceil - round towards $+\infty$
Eg :- ceil ([-2·33, 2·66]) = [-2, 3]
mod - modulus after division
Eg :- mod (26, 5) = 1 and mod (-26, 5) = 4
round - round ~~to~~ towards nearest integer
Eg :- round [(-2·33, 2·66)] = [-2, 3]
rem - remainder after division
sign - signum function
Eg :- sign ([-2·33, 2·66]) = [-1, 1]
Eg :- If a = [-1·5, 7] b = [2, 3] then
rem (a, b) = [-1·5 1]

Matrix functions :-

The matrix functions are
expm(A) - finds the exponential of matrix A
logm(A) - finds log(A) such that $A = e^{\log(A)}$
sqrtm(A) - finds $\sqrt{A}$

3·3 Character strings :-

The character strings are entered within two single right quote characters & MATLAB treat every string as a row vector with one element per character.

For example
i) message = 'Arruna'
Create a vector named message of size 1x6 [Spacing a string is counted as a character]
ii) String create a column vector of text object if they have exactly the same

names = ['John'; 'Ravi'; 'Mary'; 'Xiao']

Output :-

names =

John

Ravi

Mary

Xiao

Note :-

The command

howdy = ['Hi'; Hellow'; 'Namaste']

This will give an error because the row has different length. The correct format of howdy is

howdy = ['Hi␣␣␣␣␣'; 'Hellow␣'; 'Namaste']

Each string being 7 characters long, here ␣ denotes a space.

Example program :-

```
>> A = [1 2; 3 4]
   A =
        1  2
        3  4
>> asqrt = sqrt(A)      (square root of each element of A)
   asqrt =
             1.0000   1.4142
             1.7321   2.0000
>> Asqrt = sqrtm(A)     /sqrtm, computes √A
                        i.e) [Asqrt]*[Asqrt]
 Asqrt =
   0.5532+0.4644i   0.8070+0.2124i
   1.2104-0.3186i   1.7641+0.1458i
>> exp_aij = exp(A)
   exp_aij =   2.7183   7.3891
              20.0855  54.5982
```

```
>> exp_A = expm(A)
exp_A =
    51.9690   74.7366
   112.1048  164.0738
```

Manipulating character strings :-

The character strings can be manipulated like matrices. Thus

```
names : ['John'; 'Ravi'; 'Mary'; 'Xiao']
howdy : ['Hi'; 'Hellow'; 'Namaste']
```

Then

```
C = [howdy(2,:) names(3,:)]
C = [Hellow Mary]
```

C produces Hellow Mary as the output for C. This feature can be used along with number-to-string conversion function such as num2str and int2str which create text object containing dynamic values of variables.

For example

```
title (['variance study with sample size n="int2str
```

Output :-

Variance study with sample size n = 25

There are several built-in function for character string manipulation

char - creates character array using automatic padding also convert ASCII number values to the character array.

abs - converts character to their ASCII numeric values

blanks(n) - create n blank characters

deblank - remove the blanks from a string

eval - executes the string as a command

findstr - find the specified substring in a given string

int2str - converts integer to string
upper - convert the lower case letter into the string upper case letter
lower - convert the upper case letter into the string lower case letter
ischar - true (=1) for character array
isletter - true (=1) for alphabetical character

※ 5M(S) + 5M(S) [The eval function :-

~~The~~ MATLAB provides a powerful function 'eval' ~~is used~~ to evaluate text string and execute them if they contain valid MATLAB commands.

For example

eval('x = 5 * sin(pi/3)')

This compute x = 5sin(pi/3) and is equivalent to x = 5 * sin(pi/3)

Write a program to run a command 10 times using 'eval' function :-

```
% Example of using eval function
t = [0:0.1:1000]; % t contains 1001 elements
for k = 1:10
outputfile = ['result', int2str(k)];
theta = k*pi*t;
x = sin(theta);
y = cos(theta);
z = x.^2+y.^2;
eval(['save ', outputfile, ' x y z'])
end ] ※ 5M(S)
```

θ = kπt
x = sinθ
y = cosθ
z = x²+y²
x, y, z

3.8 2M + 5M [Plotting simple graph :-

The plots in MATLAB appear in the Graphic window. MATLAB provides good facilities for both 2D and 3D graphics by using proper MATLAB commands

10/MIS)

fplot

The 'fplot' takes the function of single variable and limits of the axis as the input and produces a plot of the function.

The syntax of fplot is

fplot('function', [xmin xmax])

Example :-

1. Plot the function $f(x) = e^{-x/10} \sin(x)$ for x between 0 and 20 using fplot function.

```
% program using fplot function
fplot ('exp(-.1*x).* sin(x)', [0 20])
xlabel('x')
ylabel('f(x)=e^(x/10) sin(x)')
title('A function plotted with fplot')
```

(or)

2. Write

```
% simple 2D graph
>> x = 0:0.1:20;
>> y = exp(-0.1*x).*sin(x);
>> plot(x,y)
>> xlabel('x')
>> ylabel('f(x)=e^(x/10) sin(x)')
>> title('A simple 2D graph')
```

ezplot :-

The 'ezplot' takes the function of a single variable and limit of the axis as the input and produces a plot of the function.

The syntax

ezplot('function', [xmin, xmax])

where the specification of the domain $x_{min} < x < x_{max}$ is optional. The default domain is $-2\pi < x < 2\pi$

10M(S)

'ezsurfc' is another form of the combination of surface plot with contour plot.

Example :-

$z = -5/(1+x^2+y^2)$ over the domain $|x|<3$ and $|y|<3$

```
z = inline ('-5./(1+x.^2+y.^2)');
ezsurfc(z,[-3,3,-3,3])
```

3.5 command line function :-

Inline function :-

A mathematical function such as F(x) (or) F(x,y) usually requires the values of the independent variables for computing the values of the function. we frequently need to evaluate such functions in scientific calculations. The syntax of the inline function is

F = inline ('function formula')

Example :-

1. Compute $F(x) = x^2 \sin(x)$ using inline function.

Solution :-

```
F = inline ('x^2* sin(x)')
```

To get the value of F(x) different output for x, we modify the program as

```
F=inline ('x.^2. *sin(x)')
```

Anonymous functions :-

Anonymous functions are functions without name created and referred by their handles. A functi handle, created internally by MATLAB and stored in a user defined variable, is basically the identity of the funct

The general format/syntax of the anonymous function is

f = @ (input list) mathematical expression

where f is the function handle.

The input list can contain a single variable or several variables separated by commas.

Example :-

```
>>fx = @(x) x^2-sin(x);        create a function f(x) = x^2 - sin x
                                 evaluate at x = 5
>> fx(5)
>> fxy = @(x,y) x^2+y^2;      evaluates f(x,y) at x=2, y=3
>> fxy(2,3)
>> fx =@(x) x.^2 -sin(x)
 >> x = [0:0.1:pi/2];
>> plot(x,fx(x))              /plot f(x) over 0 to π/2
```

Using Built-in functions and on-line help :-

MATLAB provides hundreds of built-in functions for numerical linear algebra, data analysis, Fourier transformation, data interpolation and curve fitting, root finding, Numerical solution of ODE, Numerical quadrature, sparse matrix calculations and graphics. There is online help for all built-in functions. We can get online help in several ways, they are

[help :- (the most direct online help)

By typing 'help', function name on the command line, it gives the definition of the typing function name. If you know the exact name of a function, you can get help on it.

For example typing 'help' help provides help on the function help itself.

look for :-

[The keyword search function]

If you are looking for a function, use "lookfor keyword" to get a list of functions with the string keyword in their description.

For example, typing

lists ~~or~~ functions that create identity matrices ~~will be displayed~~.

helpwin :- (the click and navigate help)

To get the online help by clicking and navigating ~~through~~ use window based help, helpwin. To ~~create~~ activate a help window, type helpwin ~~type help~~ ~~in~~ at the command prompt ~~from~~ or select the Help Window from the Help menu on the command window menu bar.

helpdesk :- (the web browser-based help)

MATLAB provides an extensive online documentation in both HTML and pdf format. If you like to read online documentation and get detailed help by clicking hyper linked text, use the web browser based help facility 'helpdesk'. To activate the help window click on the 'help' icon on the menu bar] 10M(S)

MATLAB gives the following online help

>> help - help by itself list the name of the categories in which online help files are organised.

>> matlab\general - General purpose command.

>> matlab\ops - Operators and special characters.

>> matlab\lang - programming language constructs.

>> matlab\elmat - Matrix fun Element wise matrix and manipulation.

>> matlab\elfun - Matrix function - numerical linear algebra

>> matlab\graph2d - 2D graphs

>> matlab\graph3D - 3D graphs

>> matlab\demos - examples and demonstration.

Program :-

i) Write the syntax of ezplot and plot the parametric curve $x(t) = t$, $y(t) = e^{-t/2}$ $0 < t < \pi/2$ using ezplot