MATLAB

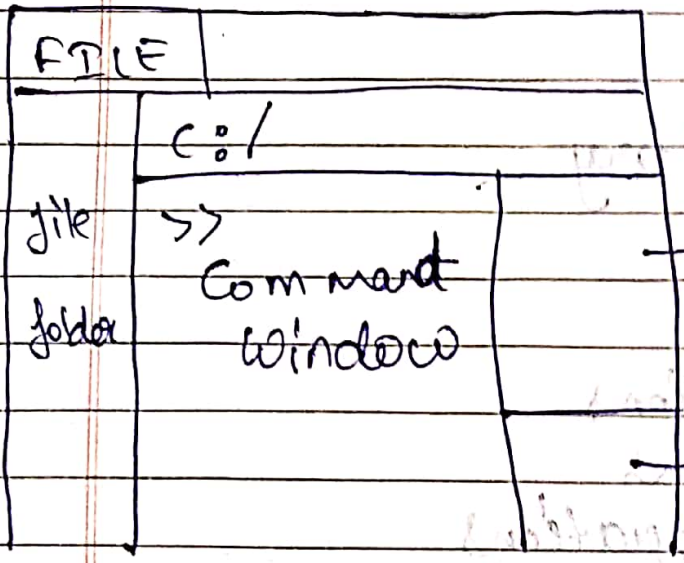
Matrix Laboratory

(i) Inbuit functions
(ii) User Defined functions
(iii) Complex Calculations
  - linear equations
  - non-linear eq.^{n} ($\tan x = x$)
(iv) Interpolation (retrieval of unknown data points)
(v) Curv fitting (defining nature of curve)
(vi) Sol.^{n} of differential eq.^{n}
(vii) Toolboxes (Simulink, Image processing)
(viii) Data Analysis
(ix) Integration

Files :-

(i) .m → script file (MATLAB EDITOR)
(ii) .mat → file to store output of Matlab
(iii) .fig → figure file
[(iv) .dat
(v) .txt]

Input & Output :-



Commands:

(i) clear
(ii) clear all
(iii) clc
(iv) who
(v) whos
(vi) what
(vii) date
(viii) clock
(ix) help
(x) look for
(xi) exit
(xii) quit
(xiii) ^C
(xiv) pwd

(i) clear → to delete particular variable → clear x
(ii) clear all → clear all the variables stored previously.
(iii) clc → clear all the screen
(iv) who → it'll give you information of variables
(v) whos → ~~dimen~~
(vi) what → give information about files & folders
(vii) pwd → for file location
(viii) date → gives date in the form of strings
(ix) clock → gives date, month, year, hours, mins, seconds in form of vector. eg-

1.0e+3 [2.019 . .]

I → 10^3

DATE: 20
PAGE No

(x) help → help, addition
(xi) lookfor → lookfor, complex
(xii) exit
(xiii) quit } → to exit MATLAB

(xiv) ^C → terminate the current window
(xv) >close → to close one figure window
(xvi) >close all → all

File name :-
(i) only first 31 characters are read.
(ii) first letter must be alphabet.
(iii) it is case sensitive.
(iv) file name should be relevant along with it should not match with existing commands.

Formatting the output :-

(i) format long → upto 15th places after decimal
(ii) format short → " 4th "
(iii) format long e → " 15th " followed by exponential
(iv) format short e → " 4th "
(v) format long g → total 15 digits
(vi) format short g → total 5 digits

↵ → enter

DATE: 20
PAGE No

eg-

x = pi → up to 15th decimal places
= 3.141592 ---- 793

>> format long
>> x = pi ↵
x = 3.1415 ---- 793
>> y = 10*x ↵
y = 31.415 ---- 7931

>> format short
>> x ↵
x = 3.1416
>> y ↵
y = 31.4159

>> format long g
>> x ↵
x = 3.1415 ---- 79
>> y ↵
y = 31.415 ---- 79

>> format short g
>> x ↵
x = 3.1415
>> y ↵
y = 31.415

DATE: 20
PAGE No

```
>> format long e
>> x ↵
x = 3.141592.....793 * 1.0 e +000
>> y ↵
y = 3.14 ---- 793 * 1.0 e +001
```

Matrices in MATLAB :-

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

syntax -

```
A = [a11 a12 a13 ; a21 a22 a23 ; ↵
     a31 a32 a33 ;]
```

Matrix Manipulations :-

```
A = [1 2 3 ; 4 5 6 ; 7 8 9];
>> A(2,3)          % to know the value
                   of a particular element
ans = 6

>> A(2,3) = 10;    % to replace
```

>> A(m:n, p:q) % selecting rows m to n & columns p to q

>> A(:, p:q) % all rows & p to q columns

>> A(m:n, :) % all columns & m to n rows

eg -

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 3 & 9 \\ 3 & 8 & 7 & 9 & 10 \\ 3 & 5 & 2 & 6 & 9 \\ 6 & 5 & 1 & 2 & 3 \end{bmatrix}$$

>> A(1:3, 2:3)

$$= \begin{bmatrix} 2 & 3 \\ 4 & 6 \\ 8 & 7 \end{bmatrix}$$

>> A(1, :) = [ ] % to delete 1st row

>> A(:, 3) = [ ] % to delete 3rd column

→ null matrix

• A particular element can not be deleted from a matrix.

>> B = A' % to calculate transpose of matrix A'.

>> reshape(A, m, n) % change the order of matrix to m×n.

eg - A = [1 2 3 4; 2 4 6 3; 3 8 7 10; 3 5 2 20]

>> B = reshape(A, 8, 2)

```
B =
     1     3
     2     6
     3     7
     3     2
     2     4
     4     3
     8    10
     5    20
```

>> B = reshape(A, 2, 8)

```
B =
     1     3     2     8     3     7     4    10
     2     3     4     5     6     2     3    20
```

>> size(A) % will give you the dimension of 'A' matrix

- null matrix :

$$A = [\ ]_{0\times 0}$$

uses- (i) to delete a row or column.

(ii) eg- j = [];

loop

x = i*i

j = [j; x]

→ to save the outputs.

- unity matrix :

```
j = [1 0; 0 1];
for i = 1:4
  si = [ :  ];
  j = j * si
end
```

- Adding a new row - no. of columns should be same

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$u = [10 \quad 20 \quad 30]$$

>> A = [A; u]

- Adding a new colum - no. of rows should be same

u = [10 ; 20]

>> A = [A u]

or

>> A = [A, u]

Go through -

eye, zeros, ones, rand, flipLR, flipud, tril, triu, creating vectors.

MATLAB

flipbe

$$A = \begin{bmatrix} 2 & 4 & 6 & 8 & 9 \\ 3 & 4 & 1 & 2 & 2 \\ 1 & 11 & 12 & 10 & 9 \end{bmatrix}$$

>> fliplr(A) % flips the matrix along column 3

$$A = \begin{bmatrix} 9 & 8 & 6 & 4 & 2 \\ 2 & 2 & 1 & 4 & 3 \\ 9 & 10 & 12 & 11 & 1 \end{bmatrix}$$

>> flipud(A) % flips matrices along R-2

$$A = \begin{bmatrix} 1 & 11 & 12 & 10 & 9 \\ 3 & 4 & 1 & 2 & 2 \\ 2 & 4 & 6 & 8 & 9 \end{bmatrix}$$

>> rot90(A) % rotate the matrix by 90° anti-clockwise

>> rot90(A, k)

k = 1 → by 90°

k = 2 → by 180°

k = 3 → by 270°

k = 4 → by 360°

>> diag( ) ← vector % it will give you a matrix with diag elements as the vector

DATE: 20
PAGE No

matrix

>> diag( ) % will give you the diagonal vectors of given matrix in column vector form.

eg- $A = \begin{bmatrix} 1 & 3 & 4 \\ 5 & 2 & 6 \\ 3 & 6 & 3 \end{bmatrix}$

>> diag (A)

$= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

>> diag (A, 1) % first upper of diagonal element

$= \begin{bmatrix} 3 \\ 6 \end{bmatrix}$

>> diag (A, -1) % first lower of diagonal element

$= \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

>> diag (A, 2) % second upper of diag. elem

= 4

>> diag (A, -2) % — lower ——

= 3

>> tril(A)

$$= \begin{bmatrix} 1 & 0 & 0 \\ 5 & 2 & 0 \\ 3 & 6 & 3 \end{bmatrix}$$

>> triu(A)

$$= \begin{bmatrix} 1 & 3 & 4 \\ 0 & 2 & 6 \\ 0 & 0 & 3 \end{bmatrix}$$

To make intervals within a specified limit —

[→ first element. → last element

>> x = a : inc : b

↳ spacing

>> x = 1 : 1 : 10

= [1 2 3 4 5 6 7 8 9 10]

>> x = 1 : 2 : 10

= [1 3 5 7 9]

Preference of spacing is higher than the limits.

>> x = linspace (a, b, n)
first ↵ ↓ No. of intervals
last

It will adjust the spacing accordingly. Preference of unity is higher.

$$inc = \frac{b-a}{n-1}$$

>> x = linspace (1, 10, 10)

= [1 2 3 4 5 6 7 8 9 10]

>> x = linspace (1, 10, 20)

$$inc = \frac{9}{19}$$

$$= \left[1 \quad \frac{28}{19} \quad \frac{37}{19} \quad \cdots \quad \frac{181}{19} \quad 10\right]$$

```
>> x = linspace (0, 2*pi, 1000);
   y = sin(x);
   plot (x,y)
```

MATRIX	ARRAY
+	+
−	−
*	.*
$AB^{-1} \leftarrow A/B$	./
$A^{-1}B \leftarrow \backslash$	.\

$m \times n$

>> ones(m, n) % to create a matrix of with all element as '1'

>> zeros(m, n) % all elements are 0

>> eye(m, n) % unity matrix of order m×n.

★ Relational Operators :-

Math	Matlab
$>$	$>$
$\geq$	$>=$
$<$	$<$
$\leq$	$<=$
$=$	$==$
$\neq$	$\sim=$

(i) you can't compare matrices of different order

(ii) you can compare a scalar with a matrix.

(iii)

Strings -

x = 'MATLAB'

→ 'a' < 'b'
= 1

highest → 'z' → small
lowest → 'A'

→ 'A' < 'X'
= 1

→ 'a' > 'A'
= 1

definition of spacing -

X = ['hi'; 'hello'; 'note']

X = char('hi', 'hello', 'note')

Trigonometric commands -

1) sin(x), cos(x)
↳ radian

2) sind(x), cosd(x)
↳ degree

3) sinh(x), cosh(x), tanh(x)

☆ Miscellaneous commands –

e^x → exp(x)

$\log_e x$ → log(x)

$\log_{10} x$ → log10(x)

$\sqrt{x}$ → $x^{\wedge}(0.5)$
sqrt(x)

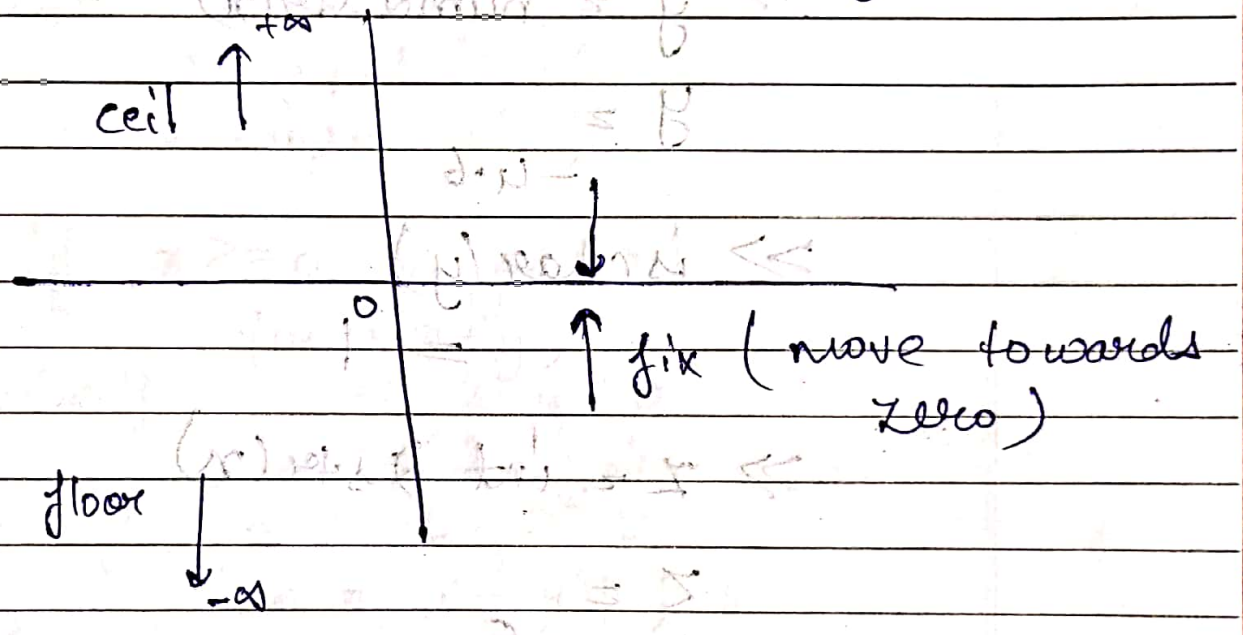
☆ Logical f.ⁿ –

- isnumeric ( ) → numbers
- ischar ( ) → strings
- isnan ( ) ← not a number
- isinf ( ) ← infinity
- isempty ( ) → empty

» isnan [inf]
= 0

Rounding commands –

- ceil (·)
- floor (·)
- fix (·)
- round (·) → nearest integer



eg- for » round (2.5)

= 3

↳ (argest integer

- for more than 1 element

» ceil ([2.4 , -2.4]) = 3 -2

```
>> x = -4.6
>> y = num2str(x)
y =
    '-4.6'
>> ischar(y)
ans = 1
>> z = int2str(x)
z =
    '-5'
>> ischar(z)
ans = 1
```

loops -

i) if - elseif - else - end

syntax -

```
if condition
    elseif condition
    else if
        else
    end
```

eg:-

$$f(x,y) = \begin{cases} x+y, & x \geq 0,\ y \geq 0 \\ x^2 e\, y, & x \geq 0,\ y < 0 \\ x - y^2, & x < 0,\ y \geq 0 \\ x^2 e\, y^2 & x < 0,\ y < 0 \end{cases}$$

code -

```
x = input('x');
y = input('y');

if x>=0 & y>=0
    fun = x+y;
elseif  x>=0 & y<0
    fun = x^2 +y;
elseif  x<0 & y>=0
    fun = x - y^2;
else
    fun = x^2 + y^2;
end
```

- (ii) for loop -

syntax - for i = start : inc : end

```
    |
    |
    |
    end
```

eg- Calculating cube of numbers from 1 to 20.

```
x[ ];
for i = 1:20
    x = [x; i^3];
end
x
```

% to see the inputs also

```
x = [x; i  i^3]
```

while loop -

Syntax -

```
while condition
end
```

eg=

```
x = 1;  X = [ ];
while x^3 < 2000
    X = [X; x  x^3];
    x = x + 1;
end
```

- Continue & break –

```
X = [];
for i = 1:1:10
    if i == 6
        continue/break
    end
    X = [X, i, i^2];
end
X
```

Plotting –

2-D –

plot (x, y, 'style option')

- line colour
  - b → blue
  - k – black
  - w – white
- marker style
  - *
  - o
  - square
- line style
  - —
  - --
  - -.

- xlabel (' ')
- ylabel (' ')
- title (' ')

- axis ([xmin xmax ymin ymax])
  ↳ to control axis

- to control only x-axis -
  axis([xmin xmax -inf inf])

- axis ('square')
  ↳ makes graph square in shape

- legend (' ', ' ')
  ↳ to insert legend in graph

- text (x, y, ' ')
  ↳ to write anything in graph

- gtext (' ')
  ↳ to write anything in graph by asking the user about position of text.

- grid on
  ↳ to show grid view

- plot($x_1, y_1, x_2, y_2, \ldots$)
  ↳ for ~~multiple graphs in same figure window~~

- hold on
  ↳ to plot multiple graphs in same figure

- subplot(m, n, p)
  - m → no. of rows
  - n → no. of column
  - p → location of fig.

  ↳ to plot different figure in same window.

eg -
```
subplot(2,2,1)
plot(--)
subplot(2,2,2)
plot(--)
subplot(2,2,3)
plot(--)
subplot(2,2,4)
plot(--)
```

3-D plotting :-

(i) Meshgrid
(ii) Mesh → to plot 3-D curves
(iii) surf → to ~~smathen~~ the boundaries
(iv) contour

$$\psi = A_0 e^{-r^2/\omega^2}$$

$$\text{gauss} = \psi = A_0 e^{-(x^2+y^2)/\omega^2}$$

```
A0 = 1.0;
w0 = 1.0;
x = -4 : 0.1 : 4;
y = x;
[X Y] = meshgrid (x,y)
Z = gauss = A0 * exp( - (X.^2 + Y.^2)./w0.^2);
figure;
mesh(X, Y, Z)
figure;
surf(X, Y, Z)
```

2-D surface obtained after being cut by Z-plane.

DATE: 20
PAGE No

contour(X, Y, Z, [0.3 0.5])

↳ contours with z = 0.3 & 0.5

contour3(X, Y, Z, 100)

↳ to show contours in 3-D

colorbar

↳ to make colourful figure at different levels

☆ function file :-

Syntax:

```
function [output variables]
= filename(input variable)
,
,
end
```

% to convert a script file cir.m to a f.n
% with input a & output x & y.

% to call a f.n in a new file

```
a = input(' ');
[x y] = circ(a);
plot(x,y)
```

% for multiple plots

```
for a = 1:10
[x y] = circ(a)
plot(x,y)
hold on
end
```

% Gaussian plot

```
function [gauss] = guassian(A0, r0, w0)
r = -10 : 0.01 : 10;
gauss = A0 * exp(-((r-r0)/w0).^2)
end
```

% to study the effect of A0

```
r0 = 1 ; w0 = 1 ;

for A0 = 1:10
    gauss = guassian (A0, r0, w0)
    plot ( r, gauss)
    hold on
end
```

Pause Command –

~~pause~~

~~for~~

pause (t)

↳ pauses the program for 't' seconds

```
for a = 1:10
    [x y] = circle(a);
    plot (x, y)

    pause (0.1)
    axis [-15 15 -15 15]
end
```

How to save inputs & outputs -

~~load ('data.dat')~~

Inputs -

Syntax - load ('filename.extn')

```
load ("data.dat")
x = data(:,1);
y = data(:,2);
plot (x,y)
```

Outputs -

(i) Syntax - save - ascii filename.extn variable

eg - save -ascii data.dat z
save -mat data.dat z

(ii) Syntax - fopen
fprintf
fclose

file identifier

eg - fid = fopen ('data.dat', 'w')
fprintf (fid, '%5.3f %4.2f', x, y);
fclose (fid)

format → [illegible]

% w.d f|d|e

- % → marker
- w → allowed width
- d → decimal precision
- f/d/e → float/decimal/exponential

\n → new line
\t → 1 tab (8 spaces)
\b → blank space

(two) syntax - disp(var)

eg- a = 3
disp(a)
disp('a')

Data Analysis :-

(i) Mean -

mean(A, 1) → 1 → column wise
mean(A, 2) → 2 → row wise

(ii) Median - same as mean (median)

(iii) standard deviation -

std(A, flag, 1/2)

↓

[0, 1]

(iv) maximum -

max (A, [], 1/2)

max (A, 2)

↳ it will replace all the values of A less than 2 by 2.

[maxval index] = max(A, [], 1)

↳ index at which maxvalue is encountered

(v) Minimum - same as max
(min)

(vi) Sum -

sum(A, 1)
~~cum~~sum(A, 1)

(vii) Product -

prod(A, 1)
cumprod(A, 1)

(viii) Difference -

diff(A, n, 1/2)
↳ no. of times you want to calculate the difference

diff(A, 2)
↳ 2 times diff. of column wise

(ix) Sort -

sort(A, 1)
sort(A, 2)

Curve fitting :-

eg 1> x = [1 2 3 4 5 6 7 8];

y = [1 2 3.5 4.5 5 5.7 6.8 8];

```
a = polyfit (x, y, 1);
xi = linspace (1, 8, 100);
yi = polyval (a, xi);
plot (x, y, 'o', xi, yi)
```

↳ symbol showing actual data points.

eg 2:

$$P(t) = P_0 e^{-t/\tau_0}$$

$$\log(P) = \log P_0 - t/\tau_0$$

Let $\log P$ = pbar

```
t = [                    ];
P = [                    ];
pbar = log(P);
tbar = t;
a = polyfit (tbar, pbar, 1);
tau = -1/a(1);
P0 = exp(a(2));
```

```
tnew = linspace(0, 20, 100);
pnew = p0.*exp(-tnew/tau);
plot(t, p, 'o', tnew, pnew)
```

eg 2:- $y = cx^d$

$\log y = \log c + d \log x$

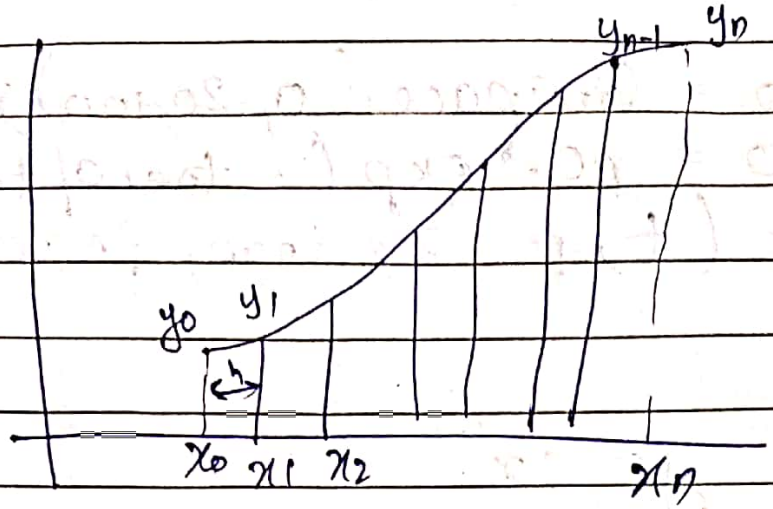
ybar, xbar

```
a = polyfit(xbar, ybar, 1)
a(1) = d
a(2) = logc
c = exp(a(2))
```

* Integration :- $I = \int_{x_1}^{x_2} y(x)\,dx$

```
trapz(x, y)
```

eg 1:- $I = \int_0^{\pi} \sin x\,dx$



$$A_1 = \frac{h}{2}(y_0 + y_1), \quad A_2 = \frac{h}{2}(y_1 + y_2)$$

$$I = A = \frac{h}{2}\left[x_0 + 2(y_1 + y_2 + \cdots + y_{n-1}) + y_n\right]$$

for $I = \int_0^{\pi} \sin x \, dx$

```
h = pi/100;
x = 0:h:pi;
y = sin(x);
sum = y(1) + y(length(x));
for i = 2: length(x)-1
  sum = sum + 2*y(i);
end
integ = (h*sum)/2;
```

⊙ eg 2:- quad ('function', xmin, xmax)
↳ if f^n is not defined earlier

quad (f, xmin, xmax)
↳ if 'f' is a defined f^n

How to define functions :-

Anonalous f^ns -

$f(x) = x^2$

f = @(x) x.^2

$f(x,y) = x \sin y + y \sin x$

f = @(x,y) x.*sin y + y.*sin x

Inline f^ns -

f = inline ('x.^2')
f = inline ('x.*sin y + y.*sin x')

→ int_dble = dblquad (f, 0, pi, 0, 2*pi)
↳ for double integration

→ 0 ≤ x ≤ π
→ 0 ≤ y ≤ 2π

Root finding :-

$$x^4 - 5x^2 + 3x - 5 = 0$$

(i) `a = [1, 0, -5, 3, -5];`

`root_poly = roots(a)`

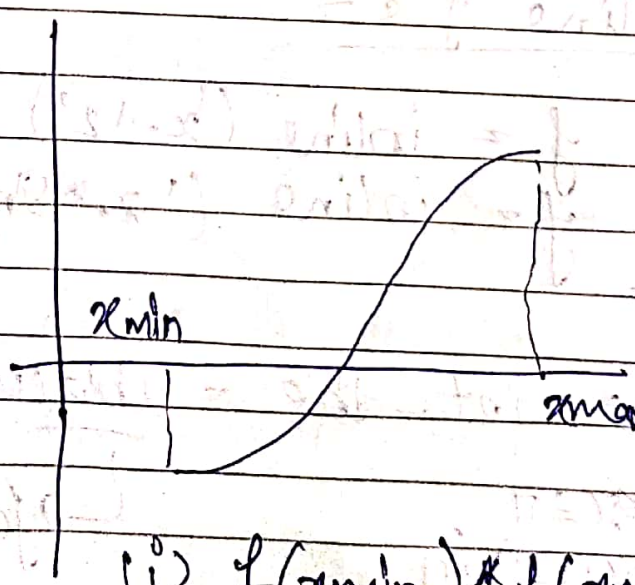
(ii) `fzero('function', guess value, acc)`

`f = @( )` — `acc` → accuracy

`root_fzero = fzero(f, x0)`

↓ It will calculate the root closest to x0

Bisection Method -



(i) $f(x_{min}) * f(x_{max}) < 0.0$

(ii) $x_{mid} = \frac{(x_{min} + x_{max})}{2}$

(iii) if $f(x_{mid}) * f(x_{max}) > 0$

$x_{max} = x_{mid}$

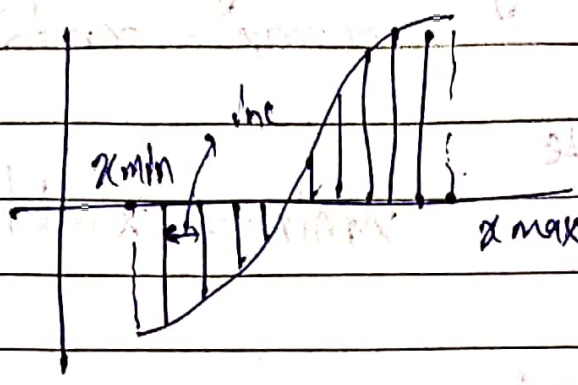
else

$x_{min} = x_{mid}$

① program:

```
f = @(x) x^2 - 4;
xmin = input(' ');
xmax = input(' ');
nstp = input(' ');
if f(xmin)*f(xmax) < 0.0
    for i = 1:nstp
        xmid = (xmax + xmin)/2;
        if f(xmid)*f(xmax) > 0
            xmax = xmid
        else
            xmin = xmid
        end
    end
else
    disp('Enter app. interval')
end
disp(['The root is', num2str(xmid)])
```

(2) program:



nstp = 100;

inc = (xmax - xmin) / 100

```
clear all;
f = @(x) x^4 - 4;
xmin = input(' ');
xmax = input(' ');
nstp = 1000;

inc = (xmax - xmin) / nstp;
ROOT = [ ];

for i = 1 : nstp
    if f(xmin) * f(xmin + inc) < 0
        root = (xmin + (xmin + inc)) / 2;
        ROOT = [ROOT ; root];
    else
    end
    xmin = xmin + inc;
end
```

if f(root)

else

Ordinary differential eq^n :-

ode 23 , ode 45

eg:

$$\frac{dx}{dt} = x + t \quad ; \quad x(t=0) = 1 \; ;$$

$x0$

```
y = @(x,t) x+t
[t  x] = ode23(y, tspan, x0)
```

ode45

t → value of 't' at which soln is needed

```
tspan = [0, 1, 2, 3, 5];
plot(t, x);
```

eg

$$u = \frac{du}{dt} = u' = -3u + 4v \quad , \quad u(t=0) = 0$$

$$v = \frac{dv}{dt} = v' = 3u - 5v \quad , \quad v(t=0) = 0.5$$

```
[t  x] = solver(function, tspan, x0)
```

solver → nothing

(ode23 or ode45)

2

program -

```
u0 = 0.00;
```

```
tspan = [0 1 2 3 ... 10];
[t z] = ode45(@func_siml, tspan, z0);
                  ↳ % calling f. file

plot(t, z(:,1), '-r', t, z(:,2), '--k')
          ↳ u              ↳ v
```

⇒ function file -

```
function [zdot] = fun_siml(t, z)
    zdot = [-3*z(1) + 4*z(2); 3*z(1) - 5*z(2)];
end
```

☆ Second order differential eq.ⁿ -

$$\frac{d^2\theta}{dt^2} + \omega^2 \sin\theta = 0, \quad \theta(t=0) = 0.0, \quad \dot{\theta}(t=0) = 0.5$$

$$\text{let } \frac{d\theta}{dt} = y, \quad \theta(0) = 0.0$$

$$\frac{dy}{dt} = -\omega^2 \sin\theta, \quad y(0) = \dot{\theta}(0) = 0.5$$

program -

```
th0 = 0.00;
th_dot0 = 0.50;

z0 = [th0; th_dot0];

tspan = [0  10];

[t  z] = ode45(@fun_ode2, tspan, z0);
figure;
plot(t, z(:,1), '-r', t, z(:,2), '--k')

legend('theta', 'thetadot');
figure;
plot(z(:,1), z(:,2))
```

⇒ function file -

```
function [zdot] = fun_ode2(t, z)
    zdot = [z(2); -wsqr*sin(z(1))];
end
```

wsqr = 1.5