* **Books** →

[① Chapra → Applied Numerical Methods
② Chapra & Canale → Numerical methods for Eng.

[③ Chapman → Matlab programming
④ R. Pratap → Getting started with MATLAB

• **Reference** →

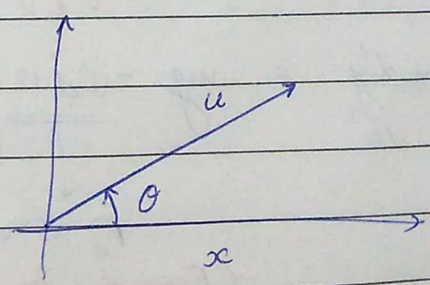
① Devries & Hasbun
- A first Course in Computational Physics

② Giordano & Nakanishi
- Computational Physics.

• Mid Term }
• End Term } → **Lab**
• Quiz →

Q. Why Computational Physics ?

Eg.1 $\vec{F} = m\vec{a}$



$$\frac{d^2y}{dx^2} = -g$$

$$\frac{dy}{dt} = -gt + k_1$$

$$\frac{d^2x}{dt^2} = 0$$

$$y = -\frac{1}{2}gt^2 + k_1 t + k_2$$

$$x = u_x t$$

- At $t=0$, $x=0$ $\quad u_x = u\cos\theta$
- $y=0$ $\quad u_y = u\sin\theta$

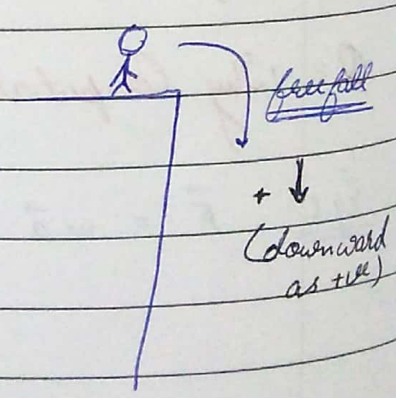
$$x = u_x t$$
$$y = u_y t - \frac{1}{2}gt^2$$

* **Restrictions on ideal condⁿ:**

1. Air drag
2. Wind velocity
3. Magnus Effect
4. Coriolis Effect

- You can see the eg. of Coriolis effect, one is due to earth's rotation the target misses its projectile.

Eg 2 **Bungee Jumping** ⇒



Coef. of air drag

$$m\frac{d^2y}{dt^2} = mg - \underline{C_d v^2}$$

$F_A \propto v^2$

(Force due to air drag)

- $$\frac{dv_y}{dt} = g - \left(\frac{C_d}{m}\right)v^2 \quad [1^{st} \text{ order diff. eq}^n]$$

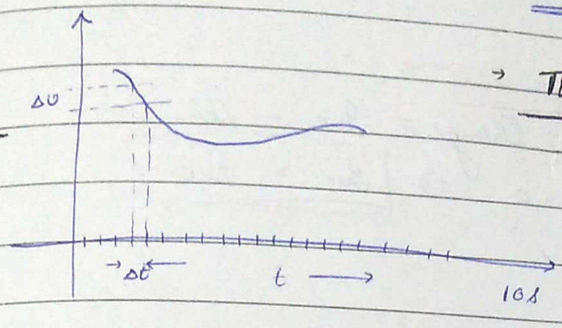
Solⁿ →

$$v(t) = \sqrt{\frac{gm}{C_d}} \tanh\left(\sqrt{\frac{gC_d}{m}}\, t\right)$$

Scanned by CamScanner

$$\left[\tanh(x) = \frac{e^{+x} - e^{-x}}{e^{x} + e^{-x}} \qquad \underline{\frac{dv}{dt} \simeq \frac{\Delta v}{\Delta t}} \right]$$

• One way to solve 1st order DE



→ <u>This is Euler Method.</u>

t	v
$t_1 = 0$	$v_1 = 0$
$t_2 = t_1 + \Delta t$	$v_2 = ?$
$t_3 = t_1 + 2\Delta t$	$v_3 = ?$
⋮	⋮
t_n	v_n

$$\frac{v(i+1) - v(i)}{t(i+1) - t(i)} = g - \left(\frac{C_d}{m}\right) v^2(i)$$

$$\Rightarrow v(i+1) = v(i) + \Delta(t) \left\{ g - \left(\frac{C_d}{m}\right) v^2(i) \right\}$$

• For $i = 1 \Rightarrow$

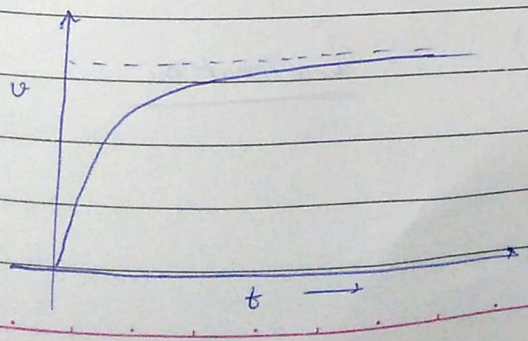
$$v(2) = v(1) + \Delta t \left\{ g - \left(\frac{C_d}{m}\right) v^2(1) \right\}$$

$\to v(1) = v_1$

$\to v(2) = v_2$ & so on.

$$\Rightarrow v_2 = v_1 + \Delta t \left\{ g - \left(\frac{C_d}{m}\right) v_1^2 \right\}$$

•



→ Free fall ⟶ <u>Terminal velocity</u>

$$\boxed{v = \sqrt{\frac{gm}{C_d}}}$$

Coding → When you have a Problem & its code, run the code for a test case where you already knows the solⁿ.

* Conservation law →

Change = Inc. − Dec.

When, change = 0 ⇒ $\boxed{\text{Inc.} = \text{Dec.}}$

* ~~Ass~~

pi; } → More precision upto 15 digits
format long; (Double precision)
pi;

format short;] → 4 digit precision
pi;

* Colⁿ Matrix → • b = [2; 4; 6; 8; 10]

• b = [2
4
6
8
10]

• b = [2 4 6 8 10]'

• who } → To know abt the variable.
whos

• MATLAB has no zero index.

x = zeros(1,11) % Preallocation

- linspace(x1, x2, n) $\longrightarrow$ To generate a row vector
- logspace() # See it
- n = 100 # by default.

x = linspace(0, 1, 6)

x = [0, 0.2, 0.4, 0.6, 0.8, 1]

$$\boxed{h = \frac{x_n - x_1}{(n-1)}}$$

$$\boxed{h = \frac{x_n - x_1}{(n-1)}} \rightarrow \text{Step size (bin width)}$$

- logspace(x1, xn, n)

x = logspace(-1, 2, 4) $\rightarrow$ $[10^{-1}\ 10^{0}\ 10^{1}\ 10^{2}]$

$x = [10^{x_1}, 10^{x_2}, 10^{x_3} \ldots]$

$x_1 = x_1$

$x_2 = x_1 + h$

$x_3 = x_2 + 2h$

If n is omitted then by default it will be 50.

* Operators $\rightarrow$ / $\rightarrow$ Division

\ $\rightarrow$ left division

^ $\rightarrow$ Power

* System of linear Eq^n $\rightarrow$

$$A = \begin{bmatrix} 3 & 2 & 5 \\ 4 & 8 & 7 \\ 5 & 3 & 8 \end{bmatrix} \qquad b = \begin{bmatrix} 4 \\ 9 \\ 11 \end{bmatrix}$$

$Ax = b$ → To find x

→ $x = A \backslash b$ % $A^{-1}b$

[In MATLAB]

[$A^{-1}Ax = A^{-1}b$

→ $x = A^{-1}b$]

* Matrix Division is undefined, btw we can do elementwise division.

• Matrix Multiplication → $\boxed{B = A*b}$

No. of ~~rows~~ col^n in 1^st matrix = No. of rows in 2^nd one.

* Element by element operation →

① Elementwise squaring → $\boxed{B = A.\hat{}2}$

② Elementwise divide → $\boxed{C = X./Y}$

* To get help → Command Prompt

① >> help log ② doc log

* Rounding → $E = [-1.6, -1.5, -1.4, 1.5, 1.6]$

① round(E)

[-2, -2, -1, 2, 2]

② ceil(E) # Push towards +ve ∞

[-1 -1 -1 2 2]

⑤ floor (E)

→ [-2, -2, -2, 1, 1]

---

① min(E) ② max(E) ③ mean (E)

④ prod(E) ⑤ sum (E) ⑥ sort (E)

<u>Note</u> → Write units in <u>comment</u>.

Eg. $v = \sqrt{\frac{gm}{cd}} \tanh\left(\sqrt{\frac{g\,cd}{m}}\; t\right)$

```
g = 9.81 ;  % m s^-2
m = 68.1 ;  % kg
```

→ Always use Std SI units while doing a ques. in MATLAB.

```
cd = 0.25 ;  % kg/m
t = 0:2:20;
v = sqrt[(g*m)/cd] * tanh( sqrt((g*cd)/m)*t);
```

% To continue a expression in next line ...

```
plot(v, t);
xlabel('velocity');
ylabel('time');
title('v vs t');
grid();
legend('velocity');
```

Camlin Page
Date / /

* Colour Specifiers →

① Blue → b
Green → g
Red → r
Cyan → c
Magenta → m
Yellow → y
Black → k
White → w

② Line →
Solid → -
Dashed → :
Dotted → -.
Dashed dot → --

③ Symbols → point .
circle o
X-mark x
Plus +
Star *
Square s
diamond d

* Array wise op →

① .* → term by term multiplication
② ./ → ——"—— division
③ .^ → ——"—— exponentiation

$h = \frac{x_n - x_1}{(n-1)}$

⇒

④ .\ → Element wise inverse multiply

* MATLAB ⇒

clear variables;
close all;

*.m , *.mat

Script files

Function files

* Multiple plot on different windows →

figure();
plot(x, y1);
figure();
plot(x, y2);

• multiplot() %. Works like Subplot

↳ Multiple plots on Same window

• Function files →

function [out1, out2, ...] = funcname (in1, in2, ...)

%. help comments

out1 = ____

out2 = ____

end

Name of the f^n file will be this

It will tell what the f^n will do, what are the i/ps & what are the o/ps.

Why Required ??

\>> help funcname

→ It will show what is written as comment written after the first line of the f^n. [declaration]

→ Path → Wherever MATLAB searches for a f^n or program, it searches

- **H1 line** → 1st line of the help.

>> lookfor bunjee
% Search all the f^n having word bunjee in first line of the help comment.

* **Eg. of f^n files** →

I function v = freefall (t, m, cd)
  % freefall : bunjee velocity with drag
  % v =

* To take value from the **user** →

>> m = input ('Mass (kg) : ')
   % This will take a value m in it.

- **O/p** →
  ① m % in Command Prompt.
  ② disp (m) ↵ % finer way
  ③ fprintf ('The velocity is %8.4f m/s \n', velocity);

8 → 3 digits on left hand side [before decimal]
4 → 4 decimal pt. after decimal

- **Format** — Description

%d	integer
%e	Scientific format
%E	"
%f	Decimal format
%g	Compact of %e or %f
\n	Newline

\t ⟶ Tab.

* To save the values →

```
>> g = 9.81;      t = 4;  d = 0.25
   v = ______ ;
save veldrag  v  d
```

% v & d will be saved in veldrag as veldrag.mat

. load veldrag   % Load the variable.

→ If we don't specify v & d then by default all the variable will be saved present in workspace.

```
>> A = [5 7 8 3 4 2];
>> save Mymtx.txt ;  ⟶ -ascii
>> clear variables;
~~>> load Mymtx.txt~~
>> A = load('Mymtx.txt');
```

* if

```
if condition
    statements
end
```

if
~~elsef~~ elseif

Eg. →

$$y = \begin{cases} -1 & ; \text{ if } x<0 \\ 0 & ; \text{ if } x=0 \\ 1 & ; \text{ if } x>0 \end{cases}$$

```
function sgn = mysign(x)
    if x<0
        sgn = -1;
    elseif x==0
        sgn=0;
    else
        sgn = 1;
    end
end
```

- **for loop →**

```
for index = start : step : finish
    ⋮
end
```

Eg. →

```
for i = 1:2:10
    disp(i)
end
```

```
for i = 10:-1:0
    disp(i)
end
```

* **Vectorization →**

```
i=0
for t = 0:0.02:50
    i = i+1;
    y(i)= cos(t^2);
end
```

Same Code

```
t= 0:0.02:50;
y = cos(t.^2);
```

This is vectorization.

→ Vectorization makes code run faster. So, do vectorization whenever possible.

Camlin Page
Date / /

* while loop →

```
while (1)                         while (condition )
    statements                        Statement
    if cond^n                         end
        break;
    end
end
```

Eg. →

```
x = 19                           |  x = 19
while (1)                        |  while (1)
    if (x<0) break; end          |      x = x-5
    x = x - 5;                   |      if x<0 , break;
    disp (x)                     |           end
end                              |      disp (x)
                                 |  end
Pre test loop                    |        Post test loop
```

• pause (0.5)   % 0.5 ms or sec. ⟶ ©

15/01/19

* Anonymous f^n →

① Script file   ② Function file

→ One more way to define f^n is Anonymous f^n.

Syntax →
→ fhandle = @(arglist) expression ;

% $f1(x,y) = x^2 + y^2$ — It can be written as.

→ `f1 = @(x,y) x.^2 + y.^2;`

Camlin Page
Date / /

```
>> a = 4
>> b = 2
```

$f(x) = a x^b$

```
f2 = @(x) a*x.^b ;
f2(3) = 36

a = 3
```

Run this again, then only value of a & b will change.

f2(3) = 27

* Plotting a func. →

```
→ fplot (func, limit) ;
```

$$v(t) = \sqrt{\frac{g*m}{d}} \tanh\left(\sqrt{\frac{gd}{m}} * t\right)$$

```
g = 9.81 ; m = 68 ; d = 0.25;

% Define f^n
vel = @(t) sqrt( ______ )      % fixing the values

% To plot
fplot (vel, [0 12]) ;   % See the step size
```

Q. Write a script to find the average vel from t=0 to 12 s

```
t = linspace (0, 12) ;
v = sqrt (g*m/d) * tanh(sqrt(g*d/m) * t);

→ vmean = mean(v)
```

→ We use the expression as Anonymous f^n, if we have to use the expression again & again otherwise not.

<u>script file</u> →

tmin = 0 ; tmax = 12 ; n = 20 ;

vel = @ (t) sqrt ( ) * tanh ( )

Lower limit   Upper limit

vmean = findmean (vel, tmin, tmax, n)

f^n ↳ data pts.

[If we have to pass the Anonymous f^n to another f^n we don't have to do vel (t) we pass only vel]

<u>Function file</u> →

```
function vmean = findmean (f, a, b, n) ;

    x = linspace (a, b, n) ;
    y = f(x) ;
    vmean = mean(y) ;
```

<u>Saved</u> → findmean.m

```
end
```

* We will always get approx answer and not exact one. So, we calculate accuracy & precision

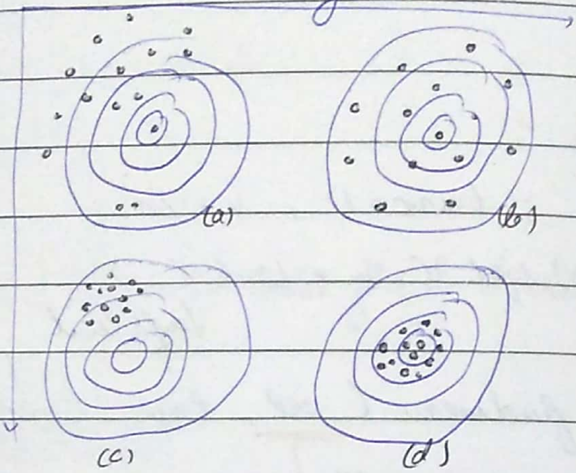
<u>Accuracy</u> - How closely we computed or measured value agrees with true value.

Camlin Page
Date / /

- **Precision** → How closely individual values agrees with one another.

Accuracy →

Precision ↓



(a) → Biased & imprecise
(b) Unbiased —— " ——
(c) Biased & precise
(d) Unbiased & —— " ——

- Error = Biasness + Impreciseness.

* **Error** → True value − Approx value.

Eg → True value of bridge = 1 km.
True —— " —— chalk = 10 cm.

Here, both have 1 cm. Error ← [Measured value of bridge = 99,999 cm.
—— " —— chalk = 9 cm]

Prob. → The value that we are measuring should also come into the def. of Error.

- **Fractional Error** = $\frac{\text{True value} - \text{Approx Exp. Value}}{\text{True value}}$

So, in above eg. → $f_b = 10^{-5}$, $f_c = 10^{-1}$
[frac. error of bridge]

. Further most of the time we don't have True value, so:

→ $$\boxed{\text{Fractional Error} = \text{abs}\left(\frac{\text{Present approx} - \text{Prev. approx}}{\text{Present approx.}}\right)}$$

* Root finding → Finding the zeros of expression.

① Graphical Method

② Bracket Method → Bisection, → False Position

③ Open end method → Newton Raphson, → Secant Method, → Successive approx.

. Lets visit Bungee Jumping Prob. → $v = \sqrt{\frac{gm}{d}} \tanh\left(\sqrt{\frac{gd}{m}}\, t\right)$

$v = 36\,ms^{-1}$, $t = 4s$, $d = 0.25\,kg/m$, $g = 9.81\,ms^{-2}$

$m = ?$

⇒ Max. mass a person can have so that it didn't cross vel. $36\,ms^{-1}$ in $4s$.

* It is very difficult to put 'm' on left hand side & everything on right because $v \to$ Explicit f^n

$m \to$ Implicit f^n of exp.

Approach →

$$f(m) = \sqrt{\frac{gm}{d}} \tanh\left(\sqrt{\frac{gd}{m}} \times t\right) - v$$

→ There will be one value of m for which exp. becomes zero.

Camlin Page
Date / /

I Graphical Method →

d= , g= , t , v =

(1)

n = linspace (40, 250)

f1 = sqr( ) —— %. Expression

Method1 plot (m, f) ;

↳ Looking at Graph we see that at apx 150 the f becomes 0.

→ By Expression

(2) g, d, v, t

f2 = @(m,d) sqrt ( ) — ;

%. Expression

Method2 fplot (f2, [40, 250]) ;

→ By Anonymous func "

Expression →

f1 = sqrt(g*m/d) .* tanh(sqrt(g*d./m) *t - v ;

[∵ we are multiplying an array with another array]

[./ as dividing by m as array will not make sense.]

Eg. f= g./m ;

→ f2 will be same as f1.

③ Modification of Method 2.

g, d, v, t

```
f2 = @(tird) ______
m = linspace(40, 250);
fval = f2(m);        → It is a 1D array
✓ plot(m, fval);
✗ plot(m, f2);       → Error as f2 is a fn & has no value
```

As a seperate fn file →

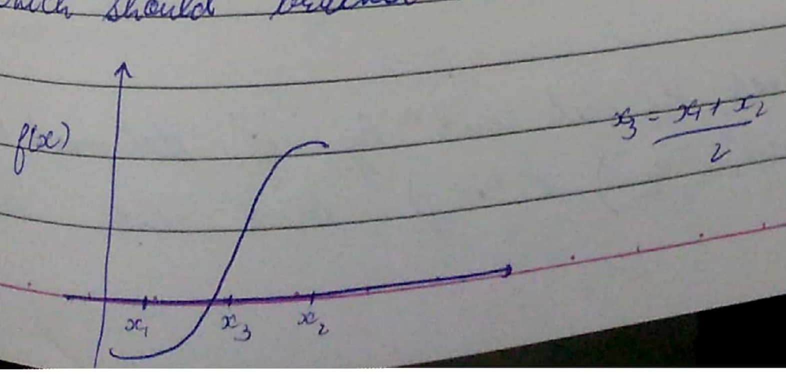
```
m, d, v, t
fval = fun4(m, d, v, t)
plot(m, fval); grid();
```

→ Function →

```
function fv = fun4(m, d, v, t)

    ~~fv =~~ g = 9.81;
    fv = sqrt(g*m/d)*tanh(sqrt(g*d./m)*t)
         - v;

end
```

II Bracket Method →

① Bisection method → ① Requires 2 guesses which should bracket the root



$$x_3 = \frac{x_1 + x_2}{2}$$

① 2 initial guesses which bracket the root.

② $x_3 = \dfrac{x_1 + x_2}{2}$

③ Check which bin encloses the root.

If (x_1, x_3) bracket the root then
$x_2 = x_3$
else if (x_3, x_2) bracket the root →
$x_1 = x_3$

④ Keep doing above until satisfied.

* **Algorithm** →

i/p → $f(x)$, x_1, x_2, tol
o/p → root

(Write the f^n for this) Check if $f(x_1) * f(x_2) < 0$ otherwise exit program.

while (∞ loop) $\qquad abs\left(\dfrac{x_1 - x_2}{x_1}\right) > tol$
$\qquad\qquad\qquad\qquad\qquad\qquad\Downarrow$
$\qquad\qquad\qquad\qquad$ → Some +ve number

$x_3 = (x_1 + x_2) / 2$;
if ~~$x_3 \circ$~~ $f(x_3) * f(x_1) < 0$
$x_2 = x_3$;
else
$x_1 = x_3$;
~~end~~ end
(Cond^n for exit)
→ while loop ends.

- You can check the exit cond^n by 'if' statement also:
if ($abs\left(\dfrac{x_1 - x_2}{x_1}\right) < tol$) break; end

* Cond^n of failure →

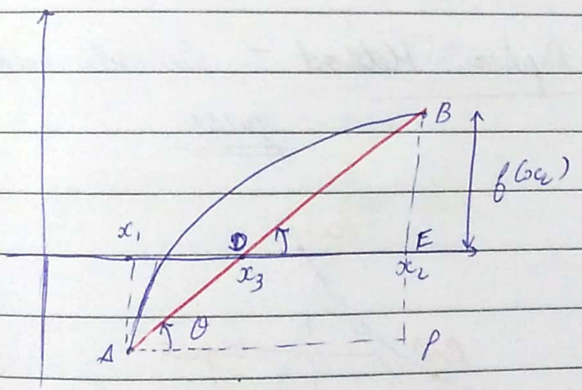
① $f(x_1) * f(x_2) < 0$
$\longrightarrow$ Odd no. of roots

② $f(x_1) * f(x_2) > 0$
$\longrightarrow$ Even no. of roots

③ $f(x_1) * f(x_2) < 0$
$\longrightarrow$ Even roots if f^n at root is tangential.

④ $f(x_1) * f(x_2) < 0$
$\longrightarrow$ Even root if f^n is discontinuous.

② Method of false Position →



In $\triangle BDE$ → $\dfrac{f(x_2)}{(x_2 - x_3)} = \tan\theta$

In $\triangle BAP$ → $\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} = \tan\theta$

So, $x_2 - x_3 = \dfrac{f(x_2)\, (x_2 - x_1)}{f(x_2) - f(x_1)}$

$$\Rightarrow \boxed{x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}}$$

→ But here sometime $f(x_2) - f(x_1)$ will be too small and dividing by a very small no. will lead to large errors.

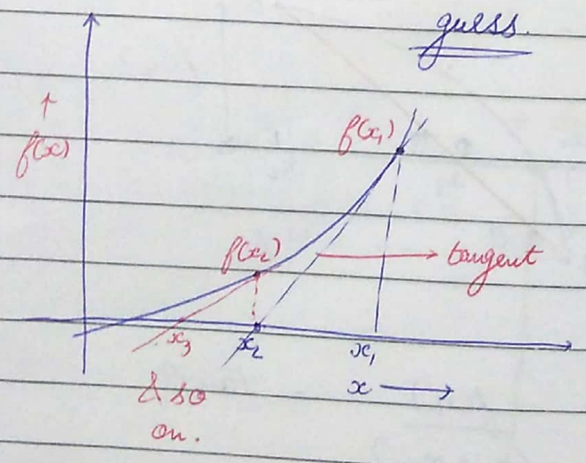
So,

if abs($f(x_2) - f(x_1)$) < tol
exit program with proper message.

// If we have to check this cond^n too. Otherwise algo. wise it will be similar as the above one

III ★ Open End Method ⇒

★① Newton - Raphson Method → We will need an initial guess.



→ The advantage here is it will converge too fast.

△ACD ⇒

$$\frac{f(x_1)}{(x_1 - x_2)} = \tan\theta = \underbrace{f'(x_1)}_{\text{derivative of } f_1 \text{ at } x_1}$$

→ $$\boxed{x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}}$$

Algo. →

i/p → $f(x)$, $f'(x)$, x_1, tol, n

$\underbrace{f(x)}_{f^n}$, $\underbrace{x_1}_{\text{Real}}$, $\underbrace{tol}_{\text{0.001, 0.0001}}$, $\underbrace{n}_{\text{Max iteration}}$

→ There can be case when this method will diverge, so we have to limit the no. of iteration (n).

o/p → root.

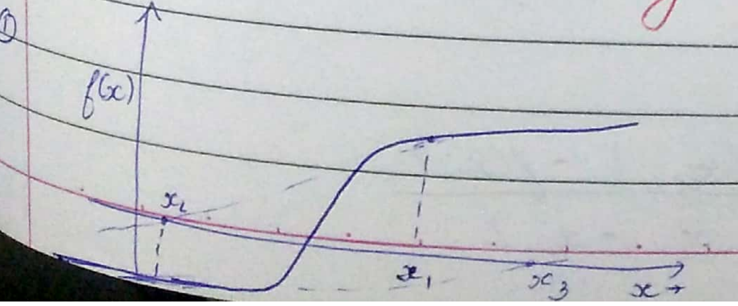
Algo. →

```
while (n > 0)
        if(abs(f'(x1)) < tol )
              print msg & exit ;
        end
    x2 = x1 - (f(x2) / f'(x1) ) ;

 ① if ( f(x2) < tol )        abs((x2 - x1) / x1) < tol
          Mine                        sir's
        break; end

 ** x1 = x2 ;      // Imp. line .
    n = n - 1 ;
end
root = x2 ;
```

→ We need to find the derivative of the f^n on paper ~~before~~ if we are using this method.

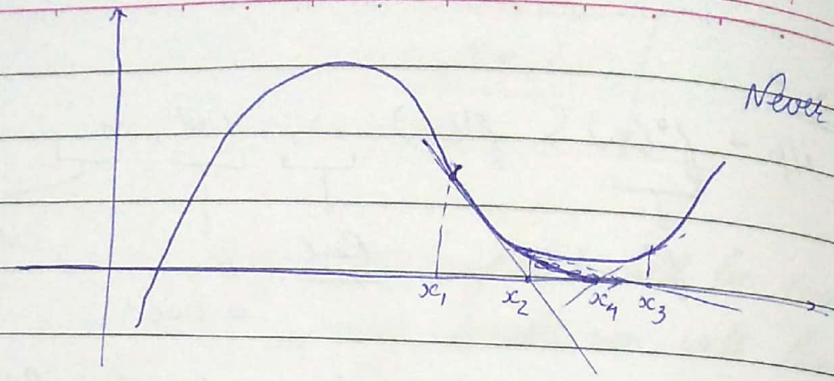
* Cond^n when it will not converge →

①



It will never converge

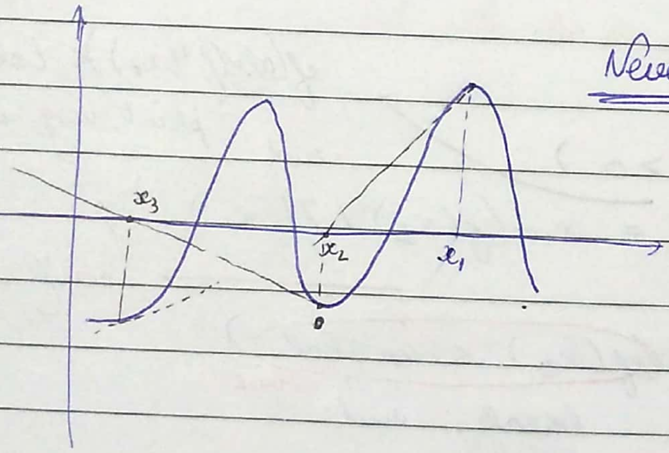
(2)



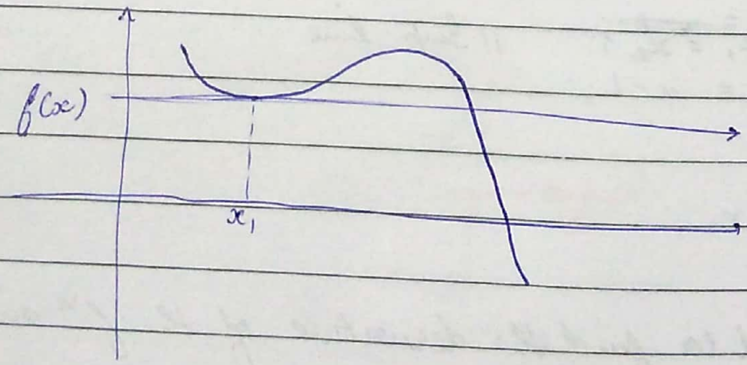
Never C.

(3)

Never C.



(4)



(2) Secant Method →

→ We don't want to find the derivatives, use this method.

$$f'(x_i) \simeq \frac{f(x_{i-1}) - f(x_i)}{(x_{i-1} - x_i)}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$\boxed{x_{i+1} = \frac{x_{i-1}\, f(x_i) - x_i\, f(x_{i-1})}{f(x_i) - f(x_{i-1})}}$$ ⇒ Secant Method

→ This is similar to false position method but the two initial guess need not to bracket the root.

Algo. →

i/p → $f(x)$, x_1, x_2, tol, n

o/p → Root

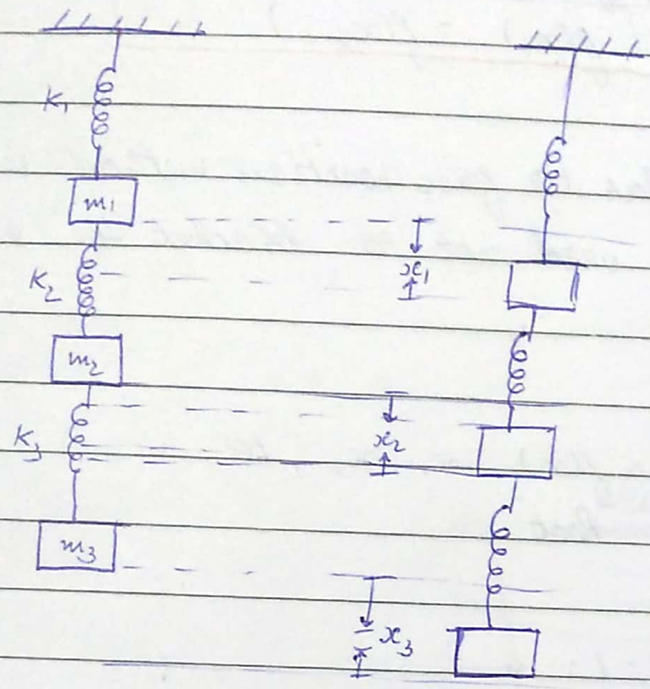
```
for  i = 1:1: n
     if (abs(f(x2) - f(x1)) < tol) Msg; break; end
     x3 = (x1 * f(x1) -  x2 f(x2)) /(f(x2) - f(x1));

     if ( abs ((x2 - x1)/x1 ) < tol )
          Msg; break;
     end

     ~~temp = x2;~~
     x1 = x2 ;
     x2 = x3 ;
end
root = x3 ;
```

22/01/19

* Sys. of linear Eq^n =>

• Problem =>

Initial



$$m_1 \frac{d^2x_1}{dt^2} = m_1 g + k_2(x_2 - x_1) - k_1 x_1$$

$$m_2 \frac{d^2x_2}{dt^2} = m_2 g + k_3(x_3 - x_2) - k_2(x_2 - x_1)$$

$$m_3 \frac{d^2x_3}{dt^2} = m_3 g - k_3(x_3 - x_2)$$

• Steady State Solution (Eqb. sol^n) =>

$$(k_1 + k_2)x_1 \quad - k_2 x_2 \quad = m_1 g$$

$$-k_2 x_1 + (k_2 + k_3)x_2 - k_3 x_3 = m_2 g$$

$$-k_3 x_2 + k_3 x_3 = m_3 g$$

- **Mtx solⁿ ⇒**

$$C = \begin{bmatrix} k_1+k_2 & -k_2 & 0 \\ -k_2 & k_2+k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix}$$

$$b = \begin{bmatrix} m_1 g \\ m_2 g \\ m_3 g \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$Cx = b$$

$$\Rightarrow \boxed{x = b/C} \text{ or, } C \backslash b$$

- **Symmetric Mtx.** ⇒ $a_{ij} = a_{ji}$

Eg → $A = \begin{bmatrix} 5 & 1 & \boxed{2} \\ 1 & 3 & \{7\} \\ \boxed{2} & \{7\} & 8 \end{bmatrix}$

- **Diagonal Matrix** ⇒ $\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$

- **Upper triangular Mtx.** ⇒ $\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$

- **Lower ∆lar Mtx.** ⇒ $\begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$

Scanned by CamScanner

- **Banded Mtx** ⇒ $\begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix}$

→ Elements above & below the diagonal Mtx are non-zero, and the rest are zero.

- Methods to solve linear eq^n ⇒

① Elimination of unknown ⇒

$a_{11}x_1 + a_{12}x_2 = b_1$ ——— ①

$a_{21}x_1 + a_{22}x_2 = b_2$ ——— ②

① × a_{21} ② × a_{11}

$$\Rightarrow x_2 = \left(\frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}}\right)$$

②
$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$ ——— ①

$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$ ——— ②

$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$ ——— ③

$C = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ $b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$

$a = \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{array}\right]$ → Augmented Mtx

② − ① $\Big($ a

a

$(a_{21} -$ a_{1}

a_{11}

0

- ③ − ① $\Big($ a_{3}

a_{11}

$(a_{31} - u_3$ a

→ Now, x_1 is

$a_{11}x_1 +$

* Pseudo Code

for i = 2

u =

a_{i1} =

② − ① $\left(\frac{a_{21}}{a_{11}}\right)$

$$\underbrace{\left(a_{21} - \frac{a_{12}}{a_{11}} \times a_{11}\right)}_{0} x_1 + \overbrace{\left(a_{22} - \left(\frac{a_{12}}{a_{11}}\right) a_{12}\right)}^{a_{22}} x_2 + \overbrace{\left(a_{23} - \left(\frac{a_{12}}{a_{11}}\right) a_{13}\right)}^{a_{23}} x_3 = \underbrace{\left(a_{24} - \left(\frac{a_{21}}{a_{11}}\right) a_{14}\right)}_{a_{24}}.$$

③ − ① $\underbrace{\left(\frac{a_{31}}{a_{11}}\right)}_{u_{3}}$ ⟹ ③ − ① u_{3}

$$(a_{31} - u_3 a_{11}) x_1 + (a_{32} - u_3 a_{12}) x_2 + (a_{33} - u_3 a_{13}) x_3 = (a_{34} - u_3 a_{14})$$

→ Now, x_1 is eliminated from ② & ③ eqⁿ ⟹

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \quad \text{--- ①}$$
$$a'_{22}x_2 + a'_{23}x_3 = a'_{24} \quad \text{--- ⑥}$$
$$a'_{32}x_2 + a'_{33}x_3 = a'_{34} \quad \text{--- ⑦}$$

* Pseudo Code ⟹

for $i = 2$ to 3 in step of 1.

$u = a_{i1} / a_{11}$

$a_{i1} = 0$.

```
    for j = 2 to 4 in step of 1
            a_ij = a_ij - u a_1j
    end
end
```

→ Eliminate x_1 from all eq^n below eq^n ①.

• Now, we will eliminate x_2 below 2^{nd} eq^n.

Then,
$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1$$
$$a_{22}' x_2 + a_{23}' x_3 = b_2'$$
$$a_{33}' x_3 = b_3''$$

→ In general we will eliminate x_k from all eq^n below the k^{th} eq^n.

```
for   i=2  to  n
      u = a_i1 / a_11
      a_i1 = 0
      for j = 2 to (n+1)
              a_ij = a_ij - u a_1j
      end
end
```

• Now we have to eliminate x_k from all eq^n below the k^{th} eq^n →

```
for  k = 1  to n-1
     for  i = (k+1) to n
          u = a_ik / a_kk
          a_ik = 0
```

The above method is Naive - Gauss Elimination Process.

Imp → In this method the diagonal element can't be zero.

for $j = (k+1)$ to $(n+1)$

$a_{ij} = a_{ij} - u\, a_{kj}$

end

end

end

→ This is forward Elimination.

Now, to find the value of x_1 to x_n we will do back - substitution.

$x_3 = a_{34} / a_{33}$

$x_2 = (a_{24} - a_{23} x_3) / a_{22}$

$x_1 = (a_{14} - a_{12} x_2 - a_{13} x_3) / a_{11}$

Algo **

→ $x_n = a_{n(n+1)} / a_{nn}$

for $i = (n-1)$ to 1 in step of (-1)

$x_i = \left(a_{i(n+1)} - \sum_{j=i+1}^{n} a_{ij} x_j \right) / a_{ii}$;

end

```
xn = a n(n+1) / a nn

for i = (n-1) : (-1) : 1
    temp = 0;
    for j = i+1 : 1 : n
        temp = temp + (a(i)(j) x(j));
    end
    x(i) = (a(i)(n+1) - temp) / a(i)(i);
end
```

Scanned by CamScanner

Q. $$A = \begin{bmatrix} k_1+k_2 & -k_2 & 0 \\ -k_2 & k_2+k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix};$$

(In laptop)

$$b = g\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix};$$

$k_1 = \overset{3}{10}$ N/m | $m_1 = 0.07$
$k_2 = \overset{2}{15}$ N/m | $m_2 = 0.065$
$k_3 = \overset{5}{20}$ N/m | $m_3 = 0.075$

2 files $\Rightarrow$ script file, f^n file

$g = 9.8\, m\, s^{-2}$

Ans. $x_1 = 0.6860$ $\quad x_2 = 1.37720$
$x_3 = 1.5190$

24/01/19

- In Forward Eli: →
- The loop can be run in a single line using vectorization.

```
a(i, (k+1):(n+1)) = a(i, (k+1):(n+1)) -
                    u * a(k, (k+1):(n+1));
```

```
for j = (k+1):(n+1)
    a(i,j) = a(i,j) - u*a(k,j);
end
```

- Backward Subs.

```
x(n) = a(n, n+1) / a(n,n);
for i = (n-1):(-1):1
    x(i) = (a(i,n+1) - a(i, (i+1):n) .* x(i+1:n
           / a(i,i));
end
```

.* → not reqd because we use :

Camlin | Page | Date / /

OR,

→ temp = 0 // It is also not needed.

temp = ~~temp +~~ a(i, i+1:n) * x(i+1:n);

x(i) = (a(i, n+1) - temp) / a(i, i)

Original loop ⇒

```
temp = 0
for j = (i+1):n
    temp = temp + a(i,j) * x(j)
end
```

Prob. →

In above method diagonal element can't be zero or some small digit.

So, we will see the modification of the above method ⇒

* Pivot Gauss Elimination Process.

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

$$A = \begin{bmatrix} \boxed{0 \quad 2 \quad 3} \\ \boxed{4 \quad 6 \quad 7} \\ 2 \quad -3 \quad 6 \end{bmatrix} \rightarrow \text{Swap} \leftarrow B = \begin{bmatrix} 8 \\ -3 \\ 5 \end{bmatrix}$$

Whichever is highest element in 1st colⁿ swap that row with 1st row (here)

Note → We are choosing the highest value as we need not to divide by small no.

I

$$a = \left[\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{array}\right]$$

→ Augmented Mtx ~~with~~ of 4 Eqⁿ

$k = 1$

→ Consider the sub colⁿ

→ Find the highest absolute value (let a_{21} here).

→ Interchange the 1st & 2nd rows (here).

→ Proceed to use forward elimination for $k = 1$.

II

$$a = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22} & a_{23} & a_{24} & a_{25} \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} \end{pmatrix}$$

$k = 2$

Start here and proceed as above.

III $k = 3$

IV $k = 4$

% Partial pivoting

① Take the k^{th} column and find the highest value in the k^{th} colⁿ starting from k^{th} row upto the n^{th} row

② Let the biggest value have index p.

③ If p is not equal to k

Swap the two rows ($p \leftrightarrow k$).

for $i = (k+1)$ to n & so on.... (previous code).

Scanned by CamScanner

$$a = \left(\begin{array}{cccc|c} 9 & 4 & 3 & 6 & 7 \\ 0 & 3 & 9 & 5 & 2 \\ 0 & -8 & 6 & 10 & 1 \\ 0 & 2 & 5 & 9 & 8 \end{array}\right)$$

$n = 4, \quad k = 2$

$a(2:4, 2) \longleftrightarrow a(k:n, k)$

$\Downarrow$

→ We have to choose max of abs. values of colⁿ.

$\max(\text{abs}(a(k:n, k))) = 8$

$\Downarrow$ to get index of 8

[bigvalue, p] = max (abs (a(k:n,k))) ;

$\downarrow$

Index → 2 ⇐ Because we consider the sub mtx as red dotted.

**

• q = p+(k-1); → Index of the original Mtx.

• Partial Pivoting Code ⇒

```
for k = 1:(n-1)
    [bigvalue, p] = max(abs(a(k:n, k)));
    q = p+(k-1);
    if (q ~= k)
        a([k,q],:) = a([q,k],:);
end
```

Swapping is done

(Rest Similar to the code before)

H.W.

Q. $C = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 6 & 7 \\ 2 & -3 & 6 \end{bmatrix}$, $b = \begin{bmatrix} 8 \\ -3 \\ 5 \end{bmatrix}$

→ Apply Pivotal Gauss Eli. method to do this

* Tridiagonal Mtx. →

$$\begin{bmatrix} f_1 & g_1 & 0 & 0 \\ e_2 & f_2 & g_2 & 0 \\ 0 & e_3 & f_3 & g_3 \\ 0 & 0 & e_4 & f_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

⇓

Banded Mtx.
or Tridiagonal
Mtx. prob

Putting this in upper triangular Mtx format

$$\begin{bmatrix} f_1 & g_1 & & \\ & f_2 & g_2 & \\ & & f_3 & g_3 \\ & & & f_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

• Now, to do this we will proceed in this way:

(2) - (1) (e_2/f_1)

$e_2 = e_2 - f_1 \times (e_2/f_1) = 0$

$f_2 = f_2 - g_1 (e_2/f_1)$

$r_2 = r_2 - r_1 (e_2 / f_1)$ → Forward Elimination

③ − ② × (e_3/f_2)

$f_3 = f_3 - g_2 (e_3/f_2)$

$r_3 = r_3 - r_2 (e_3/f_2)$

④ − ③ × (e_4/f_3)

$f_4 = f_4 - g_3 (e_4/f_3)$

$r_4 = r_4 - r_3 (e_4/f_3)$

→ So here we get the upper △lar mtx. sys. ⇒

$$\left.\begin{aligned} &i = 2 \text{ to } n \text{ in step of } 1. \\ &f_i = f_i - g_{i-1} (e_i/f_{i-1}) \\ &r_i = r_i - r_{i-1} (e_i/f_{i-1}) \end{aligned}\right\} \rightarrow \text{In Code}$$

→ Now, we will do Backward Substitution here ⇒

$$\begin{aligned} f_1 x_1 + g_1 x_2 &= r_1 \\ f_2 x_2 + g_2 x_3 &= r_2 \\ f_3 x_3 + g_3 x_4 &= r_3 \\ f_4 x_4 &= r_4 \end{aligned} \uparrow$$

$x_4 = r_4 / f_4$

$x_3 = \dfrac{r_3 - g_3 x_4}{f_3}$

$x_2 = \dfrac{r_2 - g_2 x_3}{f_2}$

$x_1 = \dfrac{r_1 - g_1 x_2}{f_1}$

We have,

$$** \left[\begin{aligned} &x_n = r_n / f_n ; \\ &x_i = \left(\frac{r_i - g_i * x_{i+1}}{f_i}\right) \end{aligned}\right. \qquad i = (n-1) \text{ to } 1 \text{ in steps of } (-1).$$

⇒ Quiz next (Tuesday) ≈ 30 min. Root finding
Sys. of Linear Eq^n

H.W.

Q.
$$\begin{bmatrix} 2.04 & -1 & 0 & 0 \\ -1 & 2.04 & -1 & 0 \\ 0 & -1 & 2.04 & -1 \\ 0 & 0 & -1 & 2.04 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 40.8 \\ 0.8 \\ 0.8 \\ 200.8 \end{pmatrix}$$

(By code).

Write down the
$e = [\quad]$
$f = [\quad]$
$g = [\quad]$
$r = [\quad]$
→ 1D Mtx

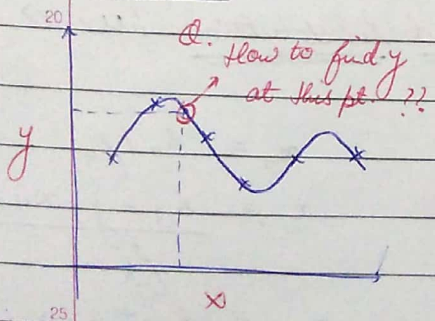
Ans.
$x_1 = 20.287$
$x_2 = 0.5886$
$x_3 = 0.3971$
$x_4 = 0.0102$

Note → Don't create an Augmented Mtx. here.

here
$e = [0 \quad -1 \quad -1 \quad -1]$
$f = [2.04 \quad 2.04 \quad 2.04 \quad 2.04]$
$g = [-1 \quad -1 \quad -1 \quad 0]$
$r = [40.8 \quad 0.8 \quad 0.8 \quad 200.8]$

29/01/19

* Interpolation →



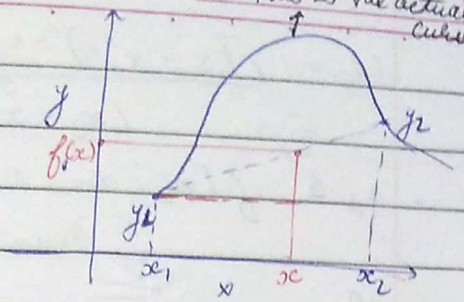
x	y
x_1	y_1
x_2	y_2
x_3	y_3
⋮	⋮
x_n	y_n

→ This can be done by Interpolation.

I Newton's Interpolating Polynomial ⇒

Linear Interpolation →

This is the actual curve



x	y
x_1	$y_1 = f(x_1)$
x_2	$y_2 = f(x_2)$

$$\frac{f_1(x) - f(x_1)}{(x - x_1)} = \frac{f(x_2) - f(x_1)}{(x_2 - x_1)}$$

$$\Rightarrow \boxed{f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{(x_2 - x_1)}(x - x_1)}$$

↳ 1st order polynomial fitting.

Quadratic Interpolation →

$f_2(x) = c_0 + c_1 x + c_2 x^2$

→ We need atleast 3 datapoints to ~~fit~~ find c_0, c_1 & c_2

So, by generalization to find n^{th} order polynomial we need atleast $(n+1)$ data points.

$f_2(x) = a_1 + \cancel{(a_2 - a_1)}\, a_2 (x - x_1) + a_3 (x - x_1)(x - x_2)$

- $f(x_1) = a_1$
- $f(x_2) = a_1 + a_2 (x_2 - x_1) = f(x_1) + a_2 (x_2 - x_1)$

~~$f(x_3) =$~~

$\Rightarrow a_2 = \dfrac{f(x_2) - f(x_1)}{x_2 - x_1} = f[x_2, x_1]$

$$a_3 = \frac{f[x_3 - x_1] - f[x_2 - x_1]}{(x_3 - x_1)}$$

$$\cdot\ a_3 \equiv f[x_3, x_2, x_1]$$

Now,

$$f_2(x) = f(x_1) + f[x_2, x_1](x - x_1) + f[x_3, x_2, x_1](x - x_1)(x - x_2)$$

* General form of Newton's Interpolating Poly. ⇒

$$f_{n-1}(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + \dots + a_n(x - x_1)(x - x_2)\dots(x - x_{n-1})$$

$x < x_1$ or $x > x_n$

$\Rightarrow \boxed{x_1 < x < x_n}$ → If we know the limits then ~~only~~ we can find the point bet^n them only.

$a_1 = f(x_1)$

$a_2 = f[x_2, x_1]$

$\vdots$

$a_n = f[x_n, x_{n-1}, \dots, x_1]$

where,

$$\rightarrow f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{(x_i - x_j)}$$

$$\rightarrow f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{(x_i - x_k)}$$

$\vdots$

$$f[x_n, x_{n-1}, \ldots, x_1] = \frac{f[x_n, x_{n-1}, \ldots, x_2] - f[x_{n-1}, \ldots, x_1]}{(x_n - x_1)}$$

$\Downarrow$

(n^{th} finite distance formula)

Eg.:- (for $n = 4$) $\Rightarrow$ no. of data points

$$f[x_4, x_3, x_2, x_1] = \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{(x_4 - x_1)}$$

$$f(x) = f(x_1) + f[x_2, x_1](x - x_1) + f[x_3, x_2, x_1](x - x_1)(x - x_2) + f[x_4, x_3, x_2, x_1](x - x_1)(x - x_2)(x - x_3)$$

To code this we write it in table format $\Rightarrow$

$j \longrightarrow$, $i \downarrow$

x_i	$j=1$: $y_i \equiv f(x_i)$	$j=2$	$j=3$	$j=4$
x_1	$f(x_1) = b_{11}$	$f[x_2, x_1] = b_{12}$	$f[x_3, x_2, x_1] = b_{13}$	$f[x_4, x_3, x_2, x_1] = b_{14}$
x_2	$f(x_2) = b_{21}$	$f[x_3, x_2] = b_{22}$	$f[x_4, x_3, x_2] = b_{23}$	0
x_3	$f(x_3) = b_{31}$	$f[x_4, x_3] = b_{32}$	0	0
x_4	$f(x_4) = b_{41}$	$b_{42} = 0$	0	0

So, we need to determine this B matrix $\Rightarrow$

2nd col^n

$$b_{12} = \frac{b_{21} - b_{11}}{(x_2 - x_1)}$$

$$b_{22} = \frac{b_{31} - b_{21}}{(x_3 - x_2)}$$

$$b_{23} = \frac{b_{41} - b_{31}}{(x_4 - x_3)}$$

3rd col

$$b_{13} = \frac{b_{22} - b_{21}}{x_3 - x_1}$$

$$b_{23} = \frac{b_{32} - b_{22}}{x_4 - x_2}$$

4th col⁴ ⇒

$$b_{14} = \frac{b_{23} - b_{13}}{(x_4 - x_1)}$$

imax

for $j = 2$ to n

  for $i = 1$ to $n-j+1$

$$b(i,j) = \frac{b(i+1, j-1) - b(i, j-1)}{x(i+j-1) - x(i)}$$

*p

  end

end

For p →

Proposed one

$j=2$	$i=1$	$p=2$	$i+j=3$	$i+j-1$ 2
	2	3	4	3
	3	4	5	4
$j=3$	$i=1$	3	4	3
	2	4	5	4
$j=4$	$i=1$	4	5	4

Correct one

$$y_f = f_3(x) = f_1(x) + f[x_2, x_1](x_f - x_1) + f[x_3, x_2, x_1](x_f - x_1)(x_f - x_2) + f[x_4, x_3, x_2, x_1](x_f - x_1)(x_f - x_2)(x_f - x_3)$$

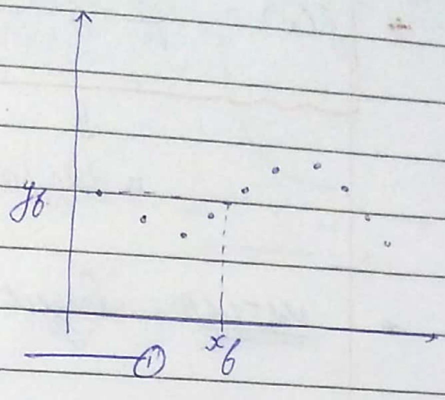
$$y_f = b_{11} + b_{12}(x_f - x_1) + b_{13}(x_f - x_1)(x_f - x_2) + b_{14}(x_f - x_1)(x_f - x_2)(x_f - x_3)$$

MATLAB : polyfit & polyval.

31/01/19

Lagrange's Interpolation →

→ No error in the datapoints.



$n = 3$

$$f(x) = c_1(x - x_2)(x - x_3) + c_2(x - x_1)(x - x_2) + c_3(x - x_1)(x - x_2) \quad \text{—— (1)}$$

x	y
x_1	$y_1 \equiv f(x_1)$
x_2	$y_2 \equiv f(x_2)$
x_3	$y_3 \equiv f(x_3)$

$$c_1 = \frac{f(x_1)}{(x_1 - x_2)(x_1 - x_3)}$$

$$c_2 = \frac{f(x_2)}{(x_2 - x_1)(x_2 - x_3)}$$

$$c_3 = \frac{f(x_3)}{(x_3 - x_1)(x_3 - x_2)}$$

Replace c_1, c_2, c_3 in eqn (1)

$$f(x) = f(x_1)\frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + f(x_2)\frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + f(x_3)\frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}$$

Now, for ~~3~~ n data points ⇒

$$\boxed{f(x) = \sum_{i=1}^{n} f(x_i) \prod_{\substack{j=1 \\ i \neq j}}^{n} \frac{(x - x_j)}{(x_i - x_j)}}$$

$\Downarrow$

n data points

---

* MATLAB : Polyfit →

Maths → $f(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$

MATLAB →

$$f(x) = p_1 x^n + p_2 x^{n-1} + \dots + p_n x + p_{n+1}$$

Eg → $T = [300, 400, 500]$

density = $[0.616, 0.525, 0.427]$

$T_f = 350$;

p = polyfit (T, density, 2)

↓ Order of poly.

↓ (Array of Coefficient)

$p = [p_1, p_2, p_3]$

d_f = polyval (p, T_f) ;

↓ density

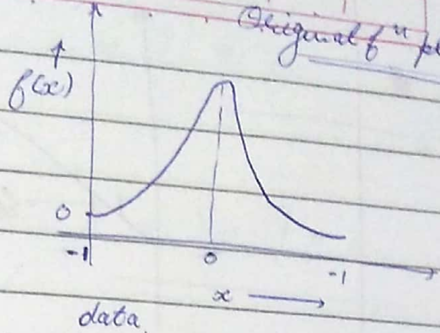
It gives the value of polynomial p at T_f

---

* Dangers of Higher Order Polynomial →

$f(x) = \dfrac{1}{1+25x^2}$

Original f^n plot

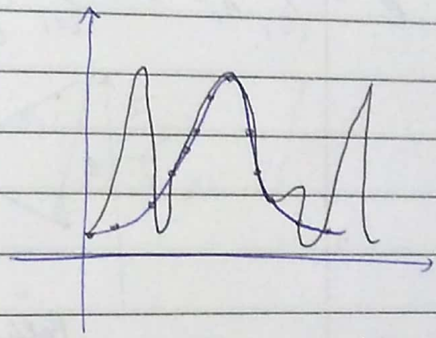


Let we have $n = 12 \longrightarrow$ no. of data points

```
xf = linspace (-1, 1)
p = polyfit (x, y, 11)
yf = polyval (p, xf)
```

```
plot (x,y);
hold ('on');
plot (xf, yf);
hold ('off');
```

$\longrightarrow$



- The plot with the help of polyfit & polyval will not fit at the end perfectly.

* Also, plotting higher order polynomial is can be erraneous. So, if you have 20 dp then you don't need to make a 19 order polynomial.

* Linear Regression →

→ When linear data has some error & we need to find the best fit line.

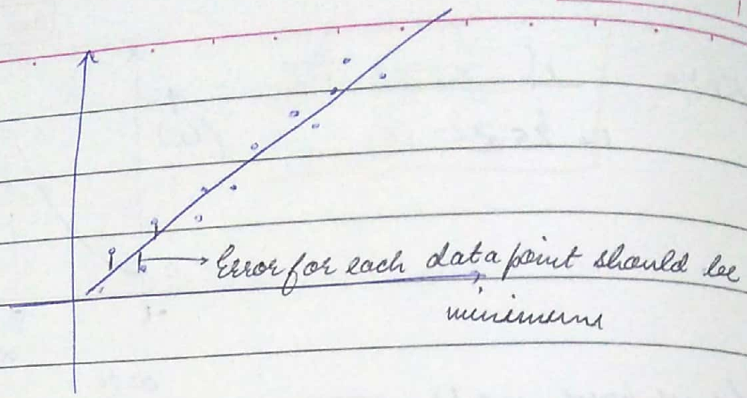
$y = a_1 + a_2 x + e$

$y_1 = a_1 + a_2 x_1 + e_1$

$y_2 = a_2 + a_3 x_2 + e_2$

$\longrightarrow$ Q. Which one is the best fit line??

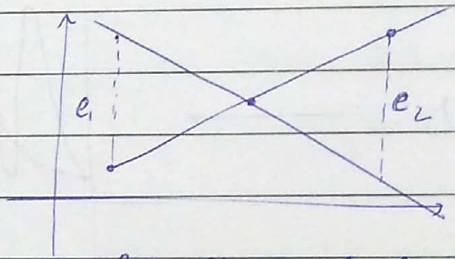
Q.



$e = y - a_1 - a_2 x$

- Criteria for <u>best fit</u> ⇒

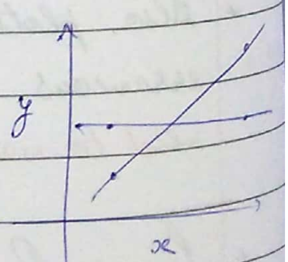
I $e = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - a_1 - a_2 x)$



Both line can be best fit as +ve & -ve error cancel out each other.

So, it is not a good criteria.

II $e = \sum_{i=1}^{n} |e_i|$



→ It is also not a very good criteria.

III ** $S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - a_1 - a_2 x_i)^2$

→ We need to minimize S_r (summation of square of error).

and find a_1 & a_2.

$$\frac{\partial S_r}{\partial a_1} = -2 \sum (y_i - a_1 - a_2 x_i) = 0 \quad \text{(i)}$$

$$\frac{\partial S_r}{\partial a_2} = -2 \sum [(y_i - a_1 - a_2 x_i) x_i] = 0 \quad \text{(ii)}$$

⇒ Taking (i) & (ii) ⇒

$$\sum y_i \; \cancel{= na_1} = n a_1 + \left(\sum x_i\right) a_2$$

$$\left(\sum x_i\right) a_1 + \left(\sum x_i^2\right) a_2 = \sum x_i y_i$$

So, we have →

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

If we have two datasets →

x	y
10	25
20	70
30	380
40	550
50	610
60	1220
70	830
80	1450

⟶ Find the best fit line. (MATLAB) ✓

* Linearization of non-linear data →

x = [ ]; y = [ ];

① $y = \alpha_1 e^{\beta_1 x}$

→ Fitting non-linear data in linear line.

$$\underbrace{\ln y}_{y'} = \underbrace{\ln \alpha_1}_{a_1} + \underbrace{\beta_1}_{a_2} x$$

② $y = \alpha_2 x^{\beta_2}$

$$\underbrace{\log y}_{y'} = \underbrace{\log \alpha_2}_{a_1} + \underbrace{\beta_2}_{a_2} \underbrace{\log x}_{x'}$$

x'	y'

③ $y = \dfrac{\alpha_3 x}{\beta_3 + x}$

$\Rightarrow \beta_3 y + xy = \alpha_3 x$

$\Rightarrow x = \left(\dfrac{\beta_3}{\alpha_3}\right) y + \dfrac{xy}{\alpha_3}$

$\Rightarrow \left(\dfrac{x}{y}\right) = \left(\dfrac{\beta_3}{\alpha_3}\right) + \left(\dfrac{x}{\alpha_3}\right)$

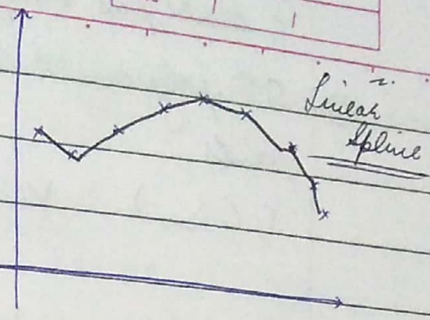
$$\Rightarrow \underbrace{\frac{1}{y}}_{y'} = \underbrace{\left(\frac{\beta_3}{\alpha_3}\right)}_{a_2} \underbrace{\frac{1}{x}}_{x'} + \underbrace{\left(\frac{1}{\alpha_3}\right)}_{a_1}$$

- These all are the ways of linearization of data

5/02/19

* Spline Interpolation →

I Linear Spline →



→ For n data points, fit a polynomial bet^n a pair of data points ⟶ $(n-1)$

x	y
x_1	y_1
x_2	y_2
⋮	⋮
x_n	y_n

⟶ Linear Spline

$$S_i = a_i + b_i(x - x_i)$$

No. of unknown = $2(n-1)$

Steps ⇒

1. Find the coefficients $a_i, b_i, \ldots$
2. Check $x_1 < x_f < x_n$.
3. Find the k^{th} bin where x_f lies.
4. $y_f = S_k(x_f) = \ldots$

---

1. The S_i^{th} spline / polynomial should pass through i^{th} point.

$$y_i = S_i(x_i) = a_i + b_i\cancel{(x_i - x_i)}$$

$$\Rightarrow \boxed{y_i = a_i} \quad ; \; i = 1, 2, \ldots, (n-1)$$

S_1 should pass through pt. 2, S_2 pass through pt. 3 & so on.

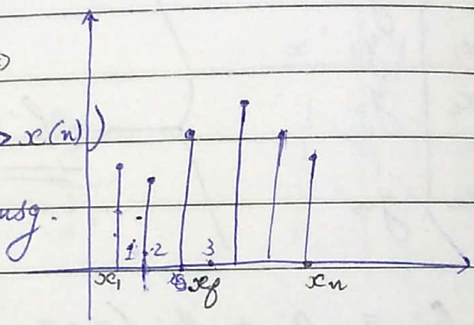
① S_i^{th} polynomial should pass through $(i+1)^{th}$ data points

$S_i(x_{i+1}) = y_{i+1} = a_i + b_i(x_{i+1} - x_i)$

$$b_i = \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) \quad ; \quad i = 1, 2, \ldots (n-1)$$

Find the k^{th} bin →

I → Sequential Search ⇒




```
if (xf < x(1) && xf > x(n))
    exit with suitable msg.
else
    bin = 0;
    for i = 1 : n
        if (x(i) < xf)) ~~(x(i)... xf < x(i+1))~~
            bin++;
        else
            break;
        end
    end.
end.
```

Mine

Sir's ⇒ ① n = length (x)

② Check whether x_f lies in betn x_1 & x_n or not.

③ $k = 1$

while(1)

if x_f less than $x(k+1)$

break loop.

end $k = k+1$;

Camlin Page
Date / /

. But this is not a good method if no. of data points are very large.
So, we will use II **Binary Search**.

i/p → x, x_f
o/p → k [bin no.]

~~for while (1)~~

```
index = round((1+n)/2);  i2 = 1;
while (1)
    if (xf < x(index) } && xf > x(index-1))
        k = index-1 ; break;
    else
        if ( xf < x(index) )
            ~~index = i2 = index;~~
            index = round((i2+ index)/2 )
            ;
```

upto point ② same

```
③ iL = 1  ;  iU = n;
while (1)
    if (iU - iL <= 1)
        ~~k = iU~~, break;
    end
    iM = floor ((iL +iU)/2 )
    if (x(iM) < xf )
        xL = iM;
    else
        xU = xM;
    end
end
k = iL;
```

Q. HW (MATLAB) $x = [-40, 0, 20, 50, 100, 150, 200, 250, 300, 400, ~~500~~, 450.];$

$x_f = 350$; find the k^{th} bin.

II Quadratic Spline →

$$S_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2$$

Unknown → $3(n-1)$ $\qquad i = 1, 2, 3, \dots (n-1)$

① S_i must pass through i^{th} data points.

$$S_i(x_i) = y_i = a_i$$

$$\Rightarrow a_i = y_i \longrightarrow i = 1, 2, \dots, (n-1)$$

Unknowns = $3(n-1) - (n-1) = 2(n-1)$

② S_i must pass through $(i+1)^{th}$ point.

$$\rightarrow S_i(x_{i+1}) = y_{i+1} = \overset{a_i}{y_i} + b_i(x_{i+1}-x_i) + c_i(x_{i+1}-x_i)^2$$

$$= y_i + b_i h_i + c_i h_i^2$$

; where $h_i = (x_{i+1} - x_i)$

Unknown → $2(n+1) - (n-1) = (n-1)$

[As we have $(n-1)$ condⁿ to find out the unknown from above]

③ Visually smooth →

→ 1^{st} order derivative at the nodes must be equal.

Camlin Page Date / /

$S_i'(x) = b_i + 2C_i(x - x_i)$

$\Rightarrow$ To make smooth $\rightarrow$

$$\boxed{S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})}$$

; $i = 2, 3, \ldots, (n-1)$

$\Rightarrow (n-2)$ cond^n

$\Rightarrow (n-1) - (n-2) = \underline{\underline{1}}$

① Natural Spline

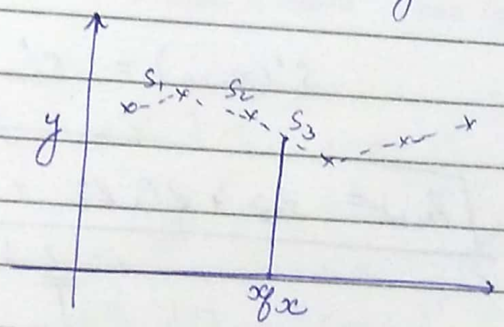
$S_i''(x) = 2C_i$

$C_i = 0$

1^{st} spline is a straight line

07/02/19

* Cubic Spline $\rightarrow$



$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

i^{th} interval (line)

Unknowns $= 4(n-1)$

① S_i must pass through (x_i, y_i)

$S_i(x_i) = y_i = a_i \quad \Rightarrow \boxed{a_i = y_i}$

$i = 1, 2, \ldots, (n-1)$

No. of reqd. const. $= 4(n-1) - (n-1) = 3(n-1)$

Scanned by CamScanner

Linear Spline — 1st order derivative of line 1-2 2-3 should be ... to make ... at 2 ... diff eqn

② $S_i(x)$ must pass through (x_{i+1}, y_{i+1})

$$y_{i+1} = S_i(x_{i+1}) = \underset{\underset{y_i}{\uparrow}}{a_i} + b_i h_i + c_i (h_i)^2 + d_i h_i^3$$

$i = 1, 2, \ldots, (n-1)$

$h_i = (x_{i+1} - x_i)$; $i = 1, 2, \ldots, n-1$

No of reqd. constraints $= 3(n-1) - (n-1) = 2(n-1)$

③ 1st order derivative for the interior nodes must be equal.

$$S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$$

$$\boxed{b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2}$$

$i = 2, 3, \ldots (n-1)$ — As const. is for interior nodes

No. of cons. reqd. → $2(n-1) - (n-2) = n$

④ 2nd order derivative at the interior nodes must be equal.

$$S_i''(x) = 2c_i + 6d_i(x - x_i)$$

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

$$\Rightarrow \boxed{d_i = \frac{c_{i+1} - c_i}{3h_i}}$$

No. of const. reqd. $= n - (n-2) = 2$

$i = 2, 3, \ldots (n-1)$

(5) Natural Spline Cond^n – (to get 2 more cond^n)

2nd order derivative of end nodes are zero.

$$S_1''(x_1) = c_1 + 6d_1(x_1 - x_1) = 0$$

$$\Rightarrow \boxed{c_1 = 0}$$

$$S_{n-1}''(x_n) = 2c_{n-1} + 6d_{n-1}(x_n - x_{n-1})$$

$$= \underbrace{c_{n-1} + 3d_{n-1}h_{n-1} = 0}_{c_n = 0}$$

$S_n(x) \longrightarrow$ $c_n = 0$

→ Last 2 cond^n are these.

This eq^n come from higher algebra

$$\boxed{h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(y[x_{i+1}, x_i] - y[x_i, x_{i-1}]\right)}$$

To find c_i $\quad i = 2, 3, \ldots, (n-1)$

* notation $\quad y[x_i, x_j] \equiv y_{ij} = \dfrac{y_i - y_j}{x_i - x_j} = \dfrac{y_i - y_j}{h_i}$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 \quad (*)$$

$$\boxed{b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(2c_i + c_{i+1})}$$

$$\boxed{d_i = \frac{c_{i+1} - c_i}{3h_i}} \quad \boxed{a_i = y_i}$$

$i = 2, 3, \ldots (n-1)$

$$\boxed{h_i = (x_{i+1} - x_i)} \quad ; \; i = 1, 2, \ldots (n-1)$$

$i = 1$:

$$\underline{\underline{C_1 = 0.}}$$

$i = 2$:

$$h_1 C_1 + 2(h_1 + h_2) C_2 + h_2 C_3 = 3\left(y[x_3, x_2] - y[x_2, x_1]\right)$$

$i = 3$:

$$h_2 C_2 + 2(h_2 + h_3) C_3 + h_3 C_4 = 3\left(y[x_4, x_3] - y[x_3, x_2]\right)$$

$\vdots$

$i = (n-1)$

$$h_{n-2} C_{n-2} + 2(h_{n-2} + h_{n-1}) C_{n-1} + h_{n-1} C_n = 3\left(y[x_n, x_{n-1}] - y[x_{n-1}, x_{n-2}]\right)$$

$i = n$:

$$\underline{\underline{C_n = 0}}$$

* The Mtx. form the above eq^n will form: <u>Tridiagonal Mtx.</u>

$$\begin{pmatrix} 1 & 0 & 0 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & h_3 & \ddots & \ddots & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h(n-1) \\ & & & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix} = \begin{bmatrix} 0 \\ 3(y_{3,2} - y_{2,1}) \\ 3(y_{4,3} - y_{3,2}) \\ \vdots \\ 3(y_{n,n-1} - y_{n-1,n-2}) \\ 0 \end{bmatrix}$$

Scanned by CamScanner

a, b, d $\longrightarrow$ array (n-1) size
c $\longrightarrow$ array

$$\begin{pmatrix} f_1 & g_1 & & & & \\ e_2 & f_2 & g_2 & & & \\ & e_3 & f_3 & g_3 & & \\ & & e_4 & f_4 & \ddots & \\ & & & e_5 & \ddots & \\ & & & \ddots & f_{n-1} & g_{n-1} \\ & & & & e_n & f_n \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ | \\ | \\ c_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ | \\ | \\ | \\ r_n \end{pmatrix}$$

$e = [0, h_1, h_2, \ldots, h_{n-2}, 0] = [0, h(1:(n-2)), 0]$
$g = [0, h_2, h_3, \ldots, h_{n-1}, 0] = [0, h(2:(n-1)), 0]$

$f_1 = 1, \quad f_n = 1$
$f_i = 2(h_{i-1} + h_i)$
$r_1 = 0$
$r_n = 0$
$$r_i = \frac{3(y_{i+1} - y_i)}{h_i} - \frac{3(y_i - y_{i-1})}{h_{i-1}} \quad ; i = 2, 3, \ldots, (n-1)$$

$$\boxed{h_i = x_{i+1} - x_i} \quad ; i = 1, 2, \ldots, (n-1)$$

* **Algo** $\Rightarrow$

Script file $\Rightarrow$

x = [ ]
y = [ ]
xf = ____
yf = cubicspline (x, y, xf)

plot (x, y, 'b-', xf, yf, 'ro');

Camlin Page
Date / /

Function file ⇒

function yf = cubicspline (x, y, xf)

Check → length (x) == length (y)

Check → $x_1 < x_f < x_n$

find n
find h ⟶ (n-1)
find e, f, g, r ⟶ n

n ⟶ c = tridiag (e, f, g, r)
find a, b, d ⟶ (n-1)

(Integer) [k = lookup bin (x, xf)
⟶ [Done by binary search]

① (k+1) is giving the apt ans. in place of k.

$$yf = a(k) + b(k)(xf - x(k)) + c(k)(xf - x(k))^2 + d(k)(xf - x(k))^3$$

end
⟶ It is k.

→ 3 f^n file , 1 script file.

Q. $x = [3, 4.5, 7, 9]$
$y = [2.5, 1, 2.5, 0.5]$
$xf = 5$
$yf = ?$

→ ① Ans. is not apt in some cases.

* ① → Ans. is coming right when you are using (K+1) in place of K in the eqⁿ.
→ It is K.

$h \to 3$

$e \to [0, h(1), 0]$

$g \to [0, h(2), 0]$

$f \to [1, 2(h_1+h_2), 2(h_2+h_3), 1]$

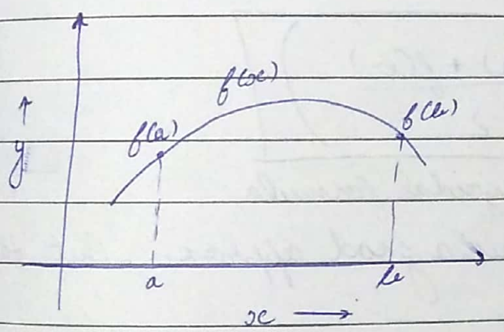
$r \to [0, \_\_\_, \_\_\_] = 3$

12/02/19

* Quiz Solⁿ → (you know the ans.)

* Integration →

$$I = \int_a^b f(x)\,dx$$



There are two types of integration:

① $f(x)$ → difficult fⁿ to integrate analytically.
② (x, y) → set of data points given.

$$I = \int_a^b f(x)\,dx = \int_a^b f_n(x)\,dx$$

Order of polynomial

I Newton-Cotes formula

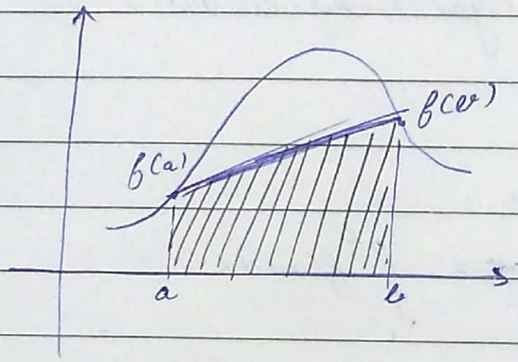
→ Substitute $f(x)$ → complicated fⁿ with a proper poly. $f_n(x)$.

- $n = 1 \longrightarrow$ Linear Polynomial

$\Downarrow$ (TM)

Name: <u>Trapezoidal Method</u> (for $n=1$)

$$I = \int_a^b f(x)\,dx = \int_a^b \left\{ f(a) + \left(\frac{f(b)-f(a)}{b-a} \right)(x-a) \right\} dx$$



$$\boxed{I = (b-a)\left(\frac{f(a)+f(b)}{2} \right)}$$

Trapezoidal formula

→ It will not give us a good approx., but this is the beginning.

- Error in Trapezoidal Method →

① It will give good results if a & b are very close to each other otherwise not.

→ Expand $f(x)$ in the interval $[0, h]$ in Taylor series abt zero.

$$I = \int_0^h f(x)\,dx$$

$\cancel{\rightarrow h = \left(\frac{b-a}{n}\right)}$

h = diff. betⁿ a & b is very small.

(Force Post Prob. → Google it) if confuse

$$f(x) = f(0) + x f'(0) + \frac{x^2}{2!} f''(0) + \frac{x^3}{3!} f'''(0) + \cdots$$

$$I_1 = \int_0^h f(x)\,dx = \int_0^h \left[\text{Taylor series}\right] dx$$

$$I_1 = h f(0) + \frac{h^2}{2} f'(0) + \frac{h^3}{3!} f''(0) + \cdots$$

$$I_2 = \frac{h}{2}\left[f'(0) + f(h)\right] \longrightarrow \cancel{\text{Trapezoidal}}\ \text{TM}$$

Expand $f(h)$ about 0.

$$= \frac{h}{2}\left[f(0) + \left\{ f(0) + h f'(0) + \frac{h^2}{2!} f''(0) + \cdots \right\}\right]$$

$$I_2^{\text{trap.}} = h f(0) + \frac{h^2}{2} f'(0) + \frac{h^3}{4} f''(0) + \cdots$$

$$\Rightarrow e_t^{\theta} = \frac{-h^3}{12} f''(0) - \frac{h^4}{24} f'''(0)$$

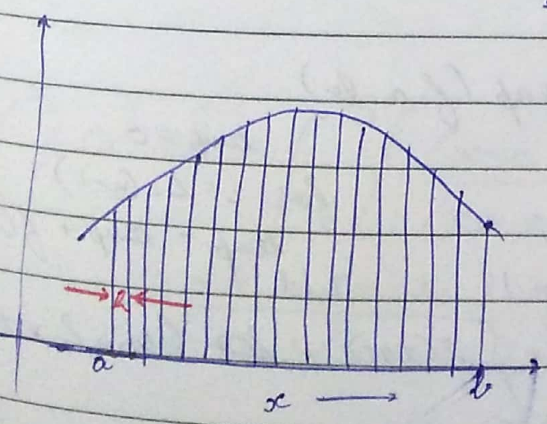
$$\boxed{e_t = I_1^{\text{exact}} - I_2^{\text{trap.}}}$$

$h \ll 1$

$$\rightarrow \boxed{e_t \simeq \frac{-h^3}{12} f''(0)}$$

Error due to TM

* Composite ~~tr~~ TM →



$$\boxed{h = \left(\frac{b-a}{n-1}\right)}$$

$$I = \int_a^b f(x)\,dx$$

$$= \int_{x_1 = a}^{x_2} f(x)\,dx + \int_{x_2}^{x_3} f(x)\,dx + \dots + \int_{x_{n-1}}^{x_n = b} f(x)\,dx$$

Imp. one

(This is derived from the prev. T^{M}.) **

$$I = \frac{h}{2}\left[\{f(x_1) + f(x_2)\} + \{f(x_2) + f(x_3)\} + \dots + \{f(x_{n-1}) + f(x_n)\}\right]$$

** $\Rightarrow$ $$I = \frac{h}{2}\left[f(x_1) + 2\sum_{i=2,3}^{n-1} f(x_i) + f(x_n)\right]$$

Code $\Rightarrow$

Script file

```
clear variables;

a = 0 ;
b = 3 * pi;
f = @(x) sin (x.^2) ;

res = trap (f, a, b);
```

f^n file

```
function res = trap (f, a, b)
    h = 0.01;
    x = a:h:b;
    n = length(x);
    temp = 0
    for i = 2:(n-1)
        temp = temp + f(xi)
    end
    res = h/2 * { f(x(1)) + ~~2*~~ ~~(sum~~ 2 * temp + f(x(n)) }
```

* Simpson's 1/3 rule $\Rightarrow$

$f_2(x)$ = quadratic polynomial

$$I = \int_a^b f(a)\,dx = \int_a^b f_2(x)\,dx$$

$$f_2(x) = C_1(x-x_2)(x-x_3) + C_2(x-x_1)(x-x_3) + C_3(x-x_1)(x-x_2)$$

$x_1 = a$, $x_3 = b$

$$C_1 = \frac{f(x_1)}{(x_1-x_2)(x_1-x_3)}$$

$$C_2 = \frac{f(x_2)}{(x_2-x_1)(x_2-x_3)}$$

$$C_3 = \frac{f(x_3)}{(x_3-x_1)(x_3-x_2)}$$

$$I = \int_{x_1}^{x_3} f_2(x)\,dx$$

$$= \int_{x_1}^{x_3} f_2(x)\,dx = \int_{x_1}^{x_3} \left[\frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f(x_1) + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f(x_2) + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} f(x_3) \right] dx.$$

$$\boxed{I = \frac{h}{3}\left[f(x_1) + 4f(x_2) + f(x_3)\right]}$$

→ Simpson's $\frac{1}{3}$ rule

Require 3 data points

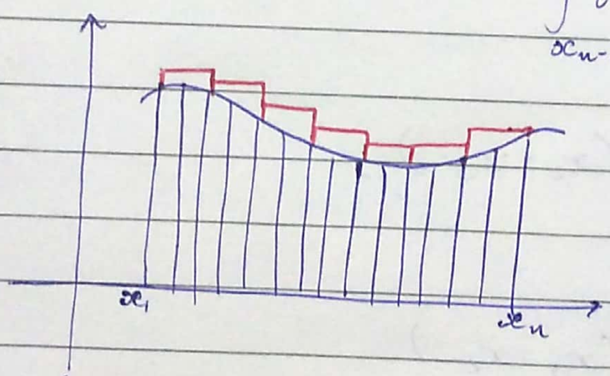
$$\boxed{e_{S_{1/3}} = \frac{-h^5}{90} f''''(c)}$$

$h = (x_2 - x_1) = (x_3 - x_2)$

* n should be odd for using this.

* Composite Simpson's $\frac{1}{3}$ ~~Formula~~ rule →

$$I = \int_{x_1}^{x_3} f(x)\,dx + \int_{x_3}^{x_5} f(x)\,dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,dx$$



n should be odd.

$$I = \frac{h}{3}\Big[\{f(x_1) + 4f(x_2) + f(x_3)\} + \{f(x_3) + 4f(x_4) + f(x_5)\} + \cdots + \{f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\}\Big]$$

$h =$

$$I = \frac{h}{3}\left[f(x_1) + 4\sum_{i=2,4,6,\dots}^{n-1} f(x_i) + 2\sum_{i=3,5,7,\dots}^{n-2} f(x_i) + f(x_n) \right]$$

* If n is even then ⇒
  Use TM for the last interval & for the rest use the SM.

(One class missing)

21/02/19

* System of Linear Equations :

* LU decomposition without pivoting → (Recap)

① $[A] = [L][U]$ ——— ①
② $[L][d] = [b]$ ——— ②
③ $[U][x] = [d]$ ——— ③

(Forward Eli. 1)

1) Decompose $[A]$ into $[L]$ and $[U]$. [FE1]
2) Use eqⁿ ② to find $[d]$. [FE2]
3) Use eqⁿ ③ to find $[x]$. [BS]. (Back Substitution)

* LU decomposition with pivoting →
We will introduce a permutation matrix $[P]$ which will keep a track of the swapping of rows.

Initially, $[P]$ = identity matrix.

$[L]([U][x] - [d]) = [P]([A][x] - [b])$

Eg. → $[A] = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $[x] = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$[b] = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$, $[P] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Lets assume $a_{11} << a_{21}$.

Pivoting.

$$[A] = \begin{pmatrix} a_{21} & a_{22} \\ a_{11} & a_{12} \end{pmatrix} = \begin{pmatrix} a_{11}' & a_{12}' \\ a_{21}' & a_{22}' \end{pmatrix}$$

$$[P] = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- After Pivoting we proceed to FE 1.

I. Proceed for FE1 ⇒

$$[L][d] = [P][b]$$

II. Before FE 2 pre-multiply the [b] mtx with [P].

→ Earlier we created augmented mtx & swapped b too along with A but here the Permutation mtx. is taking track of the swapping & b is not swapped.

$$[P][b] = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{pmatrix} b_2 \\ b_1 \end{pmatrix} = \begin{pmatrix} b_1' \\ b_2' \end{pmatrix}$$

⇒ Lets see the relation :

Naive Gauss → LU decompose without pivot

↓ ↓

Pivot Gauss ↔ LU decompose with pivot

Code

Pivot Gauss

→ Here there will two checks;
① A is a square mtx.
② No. of colⁿ of A = No. of rows of A.

```
I) FE1    P = eye(n)
for k = 1 : (n-1)
  Pivoting {
    [bigvalue, p ] = max (abs (aug(k:n, k)))      A
    q = k+p-1 ;
    if q ~= k
        aug ([k, q], : ) = aug ([q, k], : )      A
        P ([k, q], : ) = P ([q, k], : )
    end
  }
    for i = (k+1) : n
        factor = aug (i, k) / aug (k, k)      A
        aug (i, k) = 0  factor
        aug ( i, (k+1) : (n+1) ) = - - -      n+1 → n
    end
end
```

→ Now, ~~after~~ for LU decompose without pivoting :

✓ 1. No augmented matrix. [Replace aug by A].

✓ 2. Pivot matrix ⟶ swap rows of [P] mtx when pivoting.

✓ 3. Store factor.

Camlin Page
Date / /

II FE II →

$[L][d] = [P][b]$

$b = P * b$

```
d = zeros(n, 1)
for i = 1:n
   |
   |
end
```

* **Backward Substitution** →

Similar to what done in LU decomposition

```
x = zeros(n, 1)
x[n] =
for i = (n-1):(-1):1
     x(i) = __
end
```

Q. $A = \begin{bmatrix} 0.1 & -0.1 & -0.2 \\ 3 & -0.2 & -0.3 \\ 0.3 & 7.0 & 10 \end{bmatrix}$

$b = \begin{bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{bmatrix}$, find x.

Ans $x = \begin{bmatrix} -6.53 \\ 249.26 \\ -167.14 \end{bmatrix}$

inverse using LU

* Matrix decomposition without pivoting →

i/p → A → Square matrix.

o/p → Ainv

① Check whether A is a square mtx or not.

② Decompose A into L & U with pivoting 1st $P = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$

→FE 1 → ~~we have generate P mtx as before~~ 2nd $P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ & so on

FE2 → Ainv = zeros(n,n)

③ Create b mtx. ⟶

1st iteration : $b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

Do FE2 & backward subs. to find x.

Ainv(:, 1) = x(:)

Create b mtx again : $b = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

⋮

& so on.

Q. $A = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & 0.3 \\ 0.3 & 0.2 & 10 \end{bmatrix}$

Ainv = inv(A).

$Ax = b$

$x = A^{-1}b$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a_{11}' & a_{12}' & a_{13}' \\ a_{21}' & a_{22}' & a_{23}' \\ a_{31}' & a_{32}' & a_{33}' \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a_{11}' \\ a_{21}' \\ a_{31}' \end{pmatrix}$$

16UCC102

Akshat Jethlia

Computational Physics

Assignment: Miscellaneous

1. Given the equations

$$\begin{aligned} 2x_1 - 6x_2 - x_3 &= -38 \\ -3x_1 - x_2 + 7x_3 &= -34 \\ -8x_1 + x_2 - 2x_3 &= -20 \end{aligned}$$

   Solve by Gauss elimination with pivoting. Check your answer by substituting your results into the original equations.

2. It is known that the tensile strength of a plastic increases as a function of the time it is heat treated. The following data are collected:

Time (min)	10	15	20	25	40	50	55	60	75
Tensile Strength (N/m^2)	5	20	18	40	33	54	70	60	78

   Fit a straight line to these data and use the equation to determine the tensile strength (s_f) at a time (t_f) of 32 min. Plot the data, the fitted curve and the value (t_f, s_f).

3. The population of a city in a census taken once in ten years is given below. Estimate the population in the year 1975 with the help of cubic spline interpolation technique without using any of the internal matlab commands. Plot the data along with the interpolated value.

Year	1921	1931	1941	1951	1961	1971	1981
Population	35	42	58	84	120	165	220

4. The bending moment (M) for a beam of length $L = 11$ m is given by

$$M = \int_0^L f(x)\,dx\,, \qquad \text{where,} \qquad f(x) = 5 + 0.25x^2\,.$$

Here, $f(x)$ is the the shear force. Calculate M using (a) trapezoidal method and (b) Simpson's 1/3 rule. Choose suitable h and n.

5. Integration provides a means to compute how much mass (M) enters or leaves a reactor over a specified time period as a product of flow (Q) and concentration (c) as

$$M = \int_{t_2}^{t_1} Q \times c\,dt\,.$$

Use Simpson's 1/3 rule (along with trapezoidal method) to evaluate this equation for the data listed below:

t (min)	0	10	20	30	35	40	45	50
Q (m^3/min)	4	4.8	5.2	5.0	4.6	4.3	4.3	5.0
c (mg/m^3)	10	35	55	52	40	37	32	34

Akshat Tettlia $n = \text{floor}\left(\frac{t(\text{end}) - t(1)}{th}\right) + 1 \left(\frac{31}{6}\right) + 1$

16UCC102

Computational Physics

4.

35

Assignment: Matlab Fundamentals

1. Use the `linspace` function to create vectors identical to the following created with colon notation:

$x_n = u \times h + x_1,$

$\left(\frac{35-4}{6}\right)$ (a) `t = 4:6:35 ;` (b) `x = -4:2 ;`

4, 10, 16, 22, 28, 34

2. Use colon notation to create vectors identical to the following created with the `linspace` function:

Correction (a) `v = linspace(-2,1.5,8);` (b) `r = linspace(8,4.5,8);`

Note: No need to apply floor or round here.

3. If a force $F(N)$ is applied to compress a spring, its displacement x (m) can often be modeled by Hooke's law: $F = kx$, where k = spring constant (N/m). The potential energy stored in the spring U (J) can then be computed as $U = (1/2)kx^2$. Five springs are tested and the following data compiled:

F, (N)	14	18	8	9	13
x, (m)	0.013	0.020	0.009	0.010	0.012

Use MATLAB to store F and x as vectors and then compute vectors of the spring constants and the potential energies. Use the `max` function to determine the maximum potential energy.

4. It is general practice in engineering and science that equations be plotted as lines and discrete data as symbols. Here are some data for concentration (c) versus time (t) for the photo-degradation of aqueous bromine:

t, (min)	10	20	30	40	50	60
c, (ppm)	3.4	2.6	1.6	1.3	1.0	0.5

These data can be described by the following function:

$$c = 4.84\exp(-0.034\,t)$$

Use MATLAB to create a plot displaying both the data (using red diamond-shape) and the function (using a green, dashed line). Plot the function for $t = 0$ to 70 min.

5. The butterfly curve is given by the following parametric equations:

$$x = \sin(t)\left[e^{\cos(t)} - 2 * \cos(4 * t) - \sin^5\left(\frac{t}{12}\right)\right]$$

$$y = \cos(t)\left[e^{\cos(t)} - 2 * \cos(4 * t) - \sin^5\left(\frac{t}{12}\right)\right]$$

Generate the values of x and y for values of t from 0 to 100 with $\Delta t = 1/16$. Construct the plot of y vs x.

6. The butterfly curve can also be represented in polar coordinates as

$$r = \exp(\sin(\theta)) - 2\cos(4\theta) - \sin\left(\frac{2\theta - \pi}{24}\right)^5$$

Generate values of r for values of θ from 0 to 8π with $\Delta = \pi/32$. Use the MATLAB function `polar` or `polarplot` to generate the polar plot of the butterfly curve with a dashed red line.

Note → polarplot is more recommended than polar

polarplot(θ, r);

16UCC102

Akshat Jethlia

Computational Physics

Assignment: Matlab Programming

1. The sine function can be evaluated by the following infinite series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$$\sin(x) = \sum_{i=1}^{n} \frac{(-1)^{i-1}\, x^{(2i-1)}}{(2i-1)!}$$

Create an M-file to implement the summation formula so that it computes and displays the values of $\sin(x)$ as each term in the series is added. In other words, compute and display in sequence the values for

$$\sin(x) = x$$

$$\sin(x) = x - \frac{x^3}{3!}$$

$$\vdots$$

up to $n = 5$ order, i.e. up to the term $x^9/9!$ for $x = 0.9$. For each iteration compute the fractional error given by the formula

$$\text{fractional\_error} = \text{abs}\left(\frac{\text{true\_value} - \text{series\_approx}}{\text{true\_value}}\right)$$

Hint: Use the `factorial()` command, but not the `sum()` command.

2. Develop a vectorized version of the following code:

```
tstart=0; tend=20; ni=8;
t(1)=tstart;      y(1)= 10 + 5* cos(2*pi* t(1) / (tend - tstart));
for i=2:ni+1
    t(i)=t(i-1)+(tend-tstart)/ni;
    y(i)=10 + 5*cos(2*pi*t(i)/ (tend-tstart));
end
```

3. Develop an M-file to determine the polar coordinates (r, θ) for the values of (x, y) given by

```
x = [2 2 0 -3 -2 -1 0 0 2] ;
y = [0 1 3 1 0 -2 0 -2 2] ;
```

The radius r can be computed by the formula

$$r = \sqrt{x^2 + y^2}$$

whereas θ can be computed with the following table

x	y	θ
> 0	All values	$\tan^{-1}(y/x)$
< 0	> 0	$\tan^{-1}(y/x) + \pi$
< 0	< 0	$\tan^{-1}(y/x) - \pi$
< 0	$= 0$	π
$= 0$	> 0	$\pi/2$
$= 0$	< 0	$-\pi/2$
$= 0$	$= 0$	0

4. A function $f(x)$ is given by the formula

$$f(x) = \frac{\sin(x)}{x^a + b}$$

where $a = 2$, $b = 1$ are parameters. Write a function file `findmean1.m` to find the average of $f(x)$. The first couple of lines of the ~~`funmean1.m`~~ findmean1.m file is given below:

```
function avgval = findmean1(func,x)
% Input:
%   func: Input function whose average has to be calculated.
%   x: Range of x over which the function has to be evaluated.
% Output:
%   avgval: Function average value.
% ...
end
```

Hint: `yavg = findmean1(func,x);`

5. Repeat the problem of Q 4, to create another function file `findmean2.m`. In this case pass the parameters a, b as anonymous variables to the function file. The first couple of lines of the `findmean2.m` file is given below:

```
function [avgval] = findmean2(f,x,varargin)
% Find the mean of a function.
% Input:
%   func: Input function whose average has to be calculated.
%   x: Range of x over which the function has to be evaluated.
%   varargin: Variable number of arguments
% Output:
%   avgval: Function average value.
% ...
end
```

Hint: `yavg = findmean2(func,x,a,b);`