By,

JAYDEV KAMANI
U19EE006
&
AVIRAJ RATHOD
U19EE009

Faculty Advisor-

Prof. A.K.Panchal

Electrical Engg. Department
S.V.National Institute of Technology-Surat

Honor Code

WE CERTIFY THAT WE HAVE WORKED ALL THE THINGS ON OUR OWN , THAT WE HAVE NOT COPIED OR TRANSCRIBED CODES FROM CLASSMATE , SOMEONE OUTSIDE CLASS OR FROM ANY OTHER SOURCE. WE ALSO CERTIFY THAT WE HAVE NOT FACILITATED OR ALLOWED ANY OF OUR CLASSMATES TO COPY ANYTHING FROM OUR PROJECT WORK.

Index for MATLAB codes of Numerical analysis :

Interpolation methods	
1.	Direct method
2.	Lagrange's method
3.	Newton Forward and Backward method
4.	Newton's divided difference
5.	Spline method
Regression methods	
6.	Linear
7.	Quadratic
8.	Cubic
9.	4^{th} degree polynomial
Numerical integration methods	
10.	Trapezoid rule
11.	Simpson's 1/3 rule
12.	Simpson's 3/8 rule
13.	Weddle's rule
14.	Boole's rule
15.	Gaussian Quadrature (Most accurate)
Solution of Transcendental or Algebraic equations	
16.	Bisection method
17.	Regula Falsi method
18.	Secant method
19.	Newton Raphson method
Solution of Differential equations	
20.	Euler's method
21.	Modified Euler's method
22.	Runge Kutta's 2^{nd} order
23.	Runge Kutta's 4^{th} order
Comparison	
24.	Compared the data set with similar one

Note :

Before running these codes make sure that the FILE named **"DATA-SHEET"** is saved in the same folder where you've saved the codes . Don't change the sheet name and input line of code for Y variable.

Interpolation :

All the methods are included in a single code with choice of method

```
% **__Interpolation__**
clc;clear all;close all;

X_ = (3:2:49);
Y_ = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\DATA-
SHEET.xlsx','Sheet1','B1:B24');
% disp(X_);
% disp(Y_);
A_V = 2.2372e+03;
disp('Actual value is : ');
disp(A_V);
x = input('Enter x for which you want to get stock value : ');

fprintf('Direct method : 1 \nLagrange Interpolation : 2 \nNewton Forward-
Backward : 3 \nNewton Divided Difference : 4 \nSpline Interpolation : 5\n\n');
Value = input('Enter number accordingly for particular method : ');

if Value == 1
    % **__Direct method of Interpolation__**
    n = 9; % Order of interpolation
    x_m = zeros(n+1,n+1);
    x_temp = (31:2:49);
    y_m = zeros(n+1, 1);
    y_m = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\D-
Mart Stock.xlsx','Sheet1','B15:B24');
    tic
    for i = 1:n+1
            for j = 1:n+1
                x_m(i, j) = (x_temp(i))^(j-1);
            end
    end
    % disp(x_m);
    % disp(y_m);

    coeff_m = x_m\y_m;
    y = 0;
    length = size(x_m);
    % disp('Printing the co-efficient matrix :  ');
    for i=1:length(2)
        y = y + (coeff_m(i)*(x^(i-1)));
    %     disp(coeff_m(i));
    end
    toc
    disp('Direct method of Interpolation : ');
```

```
    disp(y);
    fprintf('Relative_Error in Direct method is %f\n',abs(y-A_V)/abs(A_V));


elseif Value == 2
    % **__Lagrange's Interpolation__**
    X = (31:2:49);
    Y = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\D-
Mart Stock.xlsx','Sheet1','B15:B24');
    n = 9; % Order of interpolation
    F = 0;
    tic
    for j = 1:n+1
        N = 1;D = 1;
        for k = 1:n+1
            if k ~= j
                N = N * (x - X(k));
                D = D * (X(j) - X(k));
            end
        end
        F = F + (N/D) * Y(j);
    end
    toc
    disp('Lagrange Interpolation : ');
    disp(F);
    fprintf('Relative_Error in Lagrange method is %f\n',abs(F-A_V)/abs(A_V));


elseif Value == 3
% **__Newton Forward-Backward Interpolation__**
    X__ = (35:2:45);
    Y__ = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\D-
Mart Stock.xlsx','Sheet1','B17:B22');
    tic
    % Del^1 f
    for i=1:5
        df(i) = Y__(i+1) - Y__(i);
    end
    % Del^2 f
    for i=1:4
        d2f(i) = df(i+1) - df(i);
    end
    % Del^3 f
    for i=1:3
        d3f(i) = d2f(i+1) - d2f(i);
    end
    % Del^4 f
    for i=1:2
```

```
        d4f(i) = d3f(i+1) - d3f(i);
    end
    % Del^5 f
    for i=1
        d5f(i) = d4f(i+1) - d4f(i);
    end
    a = toc;
    fprintf('Elapsed time is %f seconds.(For calculation of table)\n',a);
    h = 2;

    tic
    p = (x - X__(1)) / h;
    y1 = Y__(1) + p*df(1) + (p)*(p-1)*d2f(1)/2 + (p)*(p-1)*(p-
2)*d3f(1)/6 + (p)*(p-1)*(p-2)*(p-3)*d4f(1)/24 + (p)*(p-1)*(p-2)*(p-3)*(p-
4)*d5f(1)/120;
    b = toc;
    fprintf('Elapsed time is %f seconds.\n',a+b);
    disp('Newton Forward differnce : ');
    disp(y1);

    tic
    p = (x - X__(6)) / h;
    y2 = Y__(6) + p*df(5) + (p)*(p+1)*d2f(4)/2 + (p)*(p+1)*(p+2)*d3f(3)/6 + (p
)*(p+1)*(p+2)*(p+3)*d4f(2)/24 + (p)*(p+1)*(p+2)*(p+3)*(p+4)*d5f(1)/120;
    c = toc;
    fprintf('Elapsed time is %f seconds.\n',a+c);
    disp('Newton Backward differnce : ');
    disp(y2);
    fprintf('Relative_Error in Newton Forward differnce is %f\n',abs(y1-
A_V)/abs(A_V));
    fprintf('\nRelative_Error in Newton Backward differnce is %f\n',abs(y2-
A_V)/abs(A_V));


elseif Value == 4
%__Newton's Divided Difference for 5th Order__
    x_ = (35:2:45);
    y_ = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\D-
Mart Stock.xlsx','Sheet1','B17:B22');
    tic
    for j = 1:5
        df(j) = (y_(j+1)-y_(j))/(x_(j+1)-x_(j));
    end
    for k = 1:4
        d2f(k) = (df(k+1)-df(k))/(x_(k+2)-x_(k));
    end
    for i = 1:3
        d3f(i) = (d2f(i+1)-d2f(i))/(x_(i+3)-x_(i));
```

```
    end
    for i = 1:2
        d4f(i) = (d3f(i+1)-d3f(i))/(x_(i+4)-x_(i));
    end
    d5f = (d4f(2) - d4f(1))/(x_(6) - x_(1));
    f1 = y_(1) + (x - x_(1))*df(1) + (x - x_(1))*(x - x_(2))*d2f(1);
    f2 = (x - x_(1))*(x - x_(2))*(x - x_(3))*d3f(1) + (x - x_(1))*(x -
 x_(2))*(x - x_(3))*(x - x_(4))*d4f(1);
    f3 = (x - x_(1))*(x - x_(2))*(x - x_(3))*(x - x_(4))*(x - x_(5))*d5f;
    f = f1 + f2 + f3 ;
    toc
    disp('Newton Divided Difference for 5th Order : ');
    disp(f);
    fprintf('Relative_Error in Newton Divided Difference for 5th Order %f\n',a
bs(f-A_V)/abs(A_V));


elseif Value == 5
    % **__Spline Interpolation__**
    tic
    G = interp1(X_, Y_, x, 'spline');
    toc
    disp('Spline Interpolation : ');
    disp(G);
    fprintf('Relative_Error in Spline method is %f\n',abs(G-A_V)/abs(A_V));
end
plot(X_, Y_)
hold on
scatter(X_, Y_);
```

Command Window

```
Actual value is :
   2.2372e+03

Enter x for which you want to get stock value : 40
Direct method : 1
Lagrange Interpolation : 2
Newton Forward-Backward : 3
Newton Divided Difference : 4
Spline Interpolation : 5

Enter number accordingly for particular method : 1
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
1.006513e-24.
> In ASSI at 32
Elapsed time is 0.926830 seconds.
Direct method of Interpolation :
   2.2403e+03

Relative_Error in Direct method is 0.001401
>>
```

```
Command Window
Actual value is :
   2.2372e+03

Enter x for which you want to get stock value : 40
Direct method : 1
Lagrange Interpolation : 2
Newton Forward-Backward : 3
Newton Divided Difference : 4
Spline Interpolation : 5

Enter number accordingly for particular method : 2
Elapsed time is 0.000018 seconds.
Lagrange Interpolation :
   2.2403e+03

Relative_Error in Lagrange method is 0.001401
fx >>
```

```
Command Window
Actual value is :
   2.2372e+03

Enter x for which you want to get stock value : 40
Direct method : 1
Lagrange Interpolation : 2
Newton Forward-Backward : 3
Newton Divided Difference : 4
Spline Interpolation : 5

Enter number accordingly for particular method : 3
Elapsed time is 0.002113 seconds.(For calculation of table)
Elapsed time is 0.002120 seconds.
Newton Forward differnce :
   2.2338e+03

Elapsed time is 0.002119 seconds.
Newton Backward differnce :
   2.2338e+03

Relative_Error in Newton Forward differnce is 0.001529

Relative_Error in Newton Backward differnce is 0.001529
fx >>
```

```
Command Window
Actual value is :
   2.2372e+03

Enter x for which you want to get stock value : 40
Direct method : 1
Lagrange Interpolation : 2
Newton Forward-Backward : 3
Newton Divided Difference : 4
Spline Interpolation : 5

Enter number accordingly for particular method : 4
Elapsed time is 0.027774 seconds.
Newton Divided Difference for 5th Order :
   2.2338e+03

Relative_Error in Newton Divided Difference for 5th Order 0.001529
fx >>
```

```
Command Window
Actual value is :
   2.2372e+03

Enter x for which you want to get stock value : 40
Direct method : 1
Lagrange Interpolation : 2
Newton Forward-Backward : 3
Newton Divided Difference : 4
Spline Interpolation : 5

Enter number accordingly for particular method : 5
Elapsed time is 1.584320 seconds.
Spline Interpolation :
   2.2378e+03

Relative_Error in Spline method is 0.000256
fx >>
```

Regression :

All the methods are included in a single code with choice of method

```
% **__Regression__**
clear all;close all;clc;

n = 24;
x = (3:2:49); % month
y = zeros(1,n);
y__ = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\DATA-
SHEET.xlsx','Sheet1','B1:B24'); % closing values
for i = 1:n
    y(i) = y__(i);
end
% p1 = polyfit(x,y,4)
s_xiyi = 0; s_xi = 0;
s_yi = 0; s_xi2 = 0; s_yi2 = 0;
s_xi3 = 0; s_xi4 = 0; s_xi2yi = 0;
s_xi5 = 0; s_xi6 = 0; s_xi3yi = 0;
s_xi7 = 0; s_xi8 = 0; s_xi4yi = 0;

tic
for i=1:n
    s_xiyi = s_xiyi + y(i)*x(i);
    s_xi = s_xi + x(i);
    s_yi = s_yi + y(i);
    s_xi2 = s_xi2 + x(i)^2;
    s_yi2 = s_yi2 + y(i)^2;
end
a = toc;

r = (n * (s_xiyi) - (s_xi * s_yi)) / sqrt((n * s_xi2 - (s_xi)^2)*(n * s_yi2 -
 (s_yi)^2));
fprintf('\nCoefficient of Correlation is %0.2f \nCoefficient of Determination
is  %0.2f \n\n',r,r^2);

fprintf('Linear Regression : 1 \nQuadratic Regression : 2 \nCubic Regression :
 3 \n4th degree Polynomial : 4\n\n');
Value = input('Enter number accordingly for particular method : ');
z = input('For which value you want to regress[value > 49] : ');
fprintf('\n');

if Value == 1
    disp('Linear Regression');
    tic
    disp('a1 is...');
    a1 = (n*s_xiyi - s_xi*s_yi)/(n*s_xi2 - (s_xi)^2); disp(a1);
```

```
    disp('a0 is...');
    a0 = (s_yi/n) - a1*(s_xi/n); disp(a0);

    y_ =@(x_v) a0 + a1*x_v;
    % x_v = 3:2:51;
    % y_v =a0 + a1*x_v;
    disp(y_(z));
    b = toc;
    fprintf('Elapsed time is %f seconds.\n',a+b);

elseif Value == 2
    disp('Quadratic Regression');
    tic
    for i=1:n
        s_xi3 = s_xi3 + x(i)^3;
        s_xi4 = s_xi4 + x(i)^4;
        s_xi2yi = s_xi2yi + y(i)*(x(i)^2);
    end
    A = [n s_xi s_xi2; s_xi s_xi2 s_xi3; s_xi2 s_xi3 s_xi4;];
    disp(A);

    B = [s_yi; s_xiyi; s_xi2yi;];
    disp(B);

    X = A \  B;
    disp('Matrix for coefficient');disp(X);

    a0 = X(1, 1);
    a1 = X(2, 1);
    a2 = X(3, 1);

    y_ =@(x_) a0 + a1*x_+ a2*(x_^2);
%     x_v = 3:2:51;
%     y_v =a0 + a1*x_v + a2*(x_v.^2);
    disp(y_(z));
    b = toc;
    fprintf('Elapsed time is %f seconds.\n',a+b);

elseif Value == 3
    disp('Cubic Regression');
    tic
    for i=1:n
        s_xi3 = s_xi3 + x(i)^3;
        s_xi4 = s_xi4 + x(i)^4;
        s_xi5 = s_xi5 + x(i)^5;
        s_xi6 = s_xi6 + x(i)^6;
        s_xi2yi = s_xi2yi + y(i)*(x(i)^2);
```

```
        s_xi3yi = s_xi3yi + y(i)*(x(i)^3);
    end
    A = [n s_xi s_xi2 s_xi3; s_xi s_xi2 s_xi3 s_xi4; s_xi2 s_xi3 s_xi4 s_xi5;
s_xi3 s_xi4 s_xi5 s_xi6;];
    disp(A);

    B = [s_yi; s_xiyi; s_xi2yi; s_xi3yi];
    disp(B);

    X = A \  B;
    disp(X);

    a0 = X(1, 1);
    a1 = X(2, 1);
    a2 = X(3, 1);
    a3 = X(4, 1);

    y_ =@(x_) a0 + a1*x_+ a2*(x_^2) + a3*(x_^3);
%     x_v = 3:2:49;
%     y_v =a0 + a1*x_v + a2*(x_v.^2) + a3*(x_v.^3);
    disp(y_(z));
    b = toc;
    fprintf('Elapsed time is %f seconds.\n',a+b);

elseif Value == 4
    disp('4th Degree Polynomial Regression');
    tic
    for i=1:n
        s_xi3 = s_xi3 + x(i)^3;
        s_xi4 = s_xi4 + x(i)^4;
        s_xi5 = s_xi5 + x(i)^5;
        s_xi6 = s_xi6 + x(i)^6;
        s_xi2yi = s_xi2yi + y(i)*(x(i)^2);
        s_xi3yi = s_xi3yi + y(i)*(x(i)^3);
        s_xi7 = s_xi7 + x(i)^7;
        s_xi8 = s_xi8 + x(i)^8;
        s_xi4yi = s_xi4yi + y(i)*(x(i)^4);
    end
    A = [n s_xi s_xi2 s_xi3 s_xi4; s_xi s_xi2 s_xi3 s_xi4 s_xi5; s_xi2 s_xi3 s
_xi4 s_xi5 s_xi6; s_xi3 s_xi4 s_xi5 s_xi6 s_xi7; s_xi4 s_xi5 s_xi6 s_xi7 s_xi8
;];
    disp(A);

    B = [s_yi; s_xiyi; s_xi2yi; s_xi3yi; s_xi4yi];
    disp(B);

    X = A \  B;
    disp(X);
```

```
    a0 = X(1, 1);
    a1 = X(2, 1);
    a2 = X(3, 1);
    a3 = X(4, 1);
    a4 = X(5, 1);

    y_ =@(x_) a0 + a1*x_+ a2*(x_^2) + a3*(x_^3) + a4*(x_^4);
    % x_v = 3:2:51;
    % y_v =a0 + a1*x_v + a2*(x_v.^2) + a3*(x_v.^3) + a4*(x_v.^4);
    disp(y_(z));
    b = toc;
    fprintf('Elapsed time is %f seconds.\n',a+b);
else
    disp('Enter the correct value');
    break;
end

A_V = 2982.25; %for x = 50
R_E = abs(y_(z)-A_V)/A_V;
disp('Relative error');disp(R_E);

scatter(x, y);
% hold on
% plot(x,polyval(p1,x));
```

Command Window

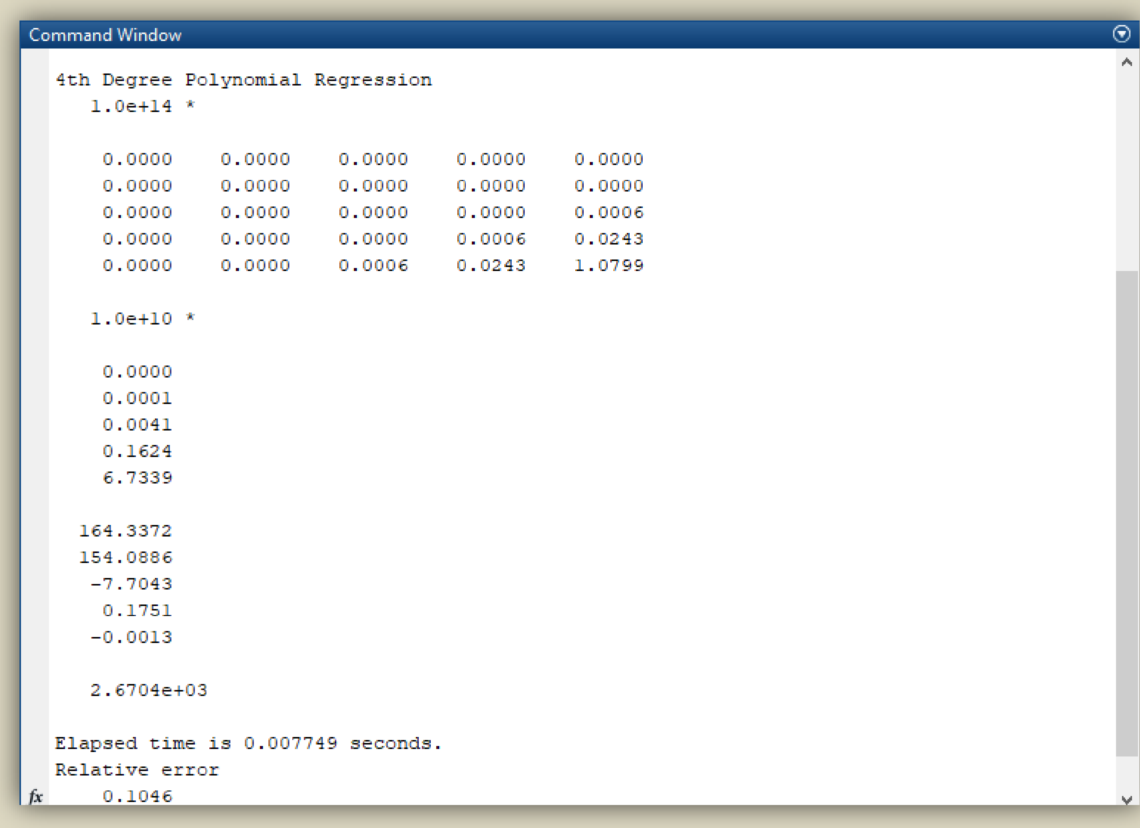
```
Coefficient of Correlation is 0.94
Coefficient of Determination is  0.89

Linear Regression : 1
Quadratic Regression : 2
Cubic Regression : 3
4th degree Polynomial : 4

Enter number accordingly for particular method : 1
For which value you want to regress[value > 49] : 50

Linear Regression
a1 is...
   36.0019

a0 is...
  617.2021

   2.4173e+03

Elapsed time is 0.005920 seconds.
Relative error
    0.1894
```

Command Window

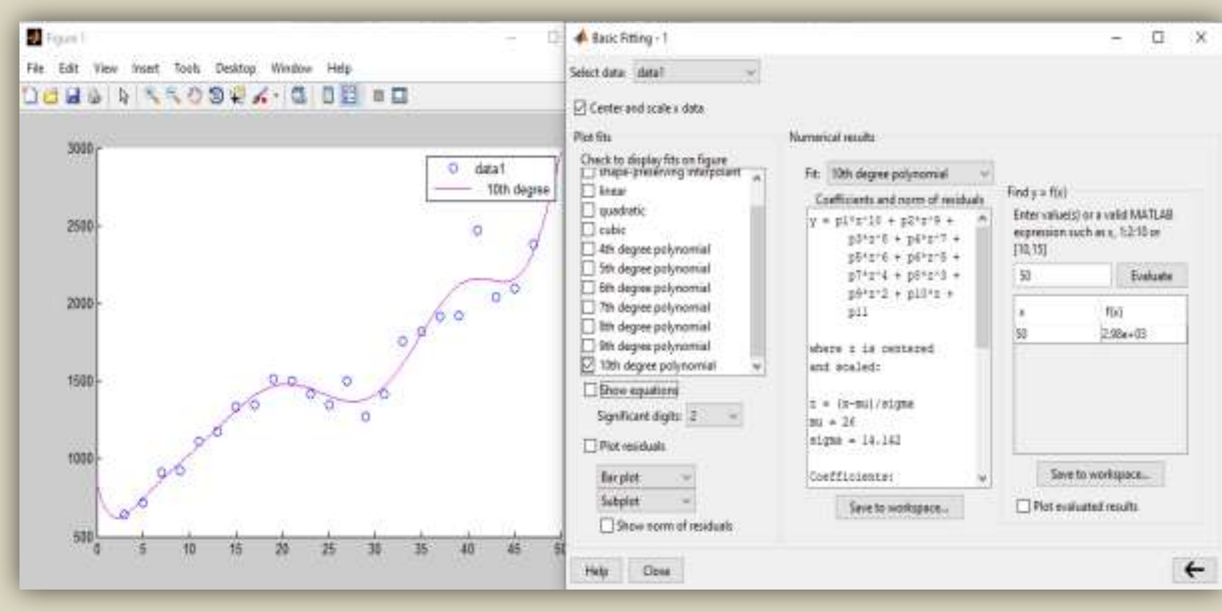
```
Linear Regression : 1
Quadratic Regression : 2
Cubic Regression : 3
4th degree Polynomial : 4

Enter number accordingly for particular method : 2
For which value you want to regress[value > 49] : 50

Quadratic Regression
          24         624       20824
         624       20824      780624
       20824      780624    31208344

   1.0e+07 *

    0.0037
    0.1135
    4.1140

Matrix for coefficient
  743.8796
   22.4013
    0.2616

   2.5178e+03

Elapsed time is 0.101228 seconds.
Relative error
    0.1557
```

Command Window

```
Enter number accordingly for particular method : 3
For which value you want to regress[value > 49] : 50

Cubic Regression
   1.0e+10 *

    0.0000    0.0000    0.0000    0.0001
    0.0000    0.0000    0.0001    0.0031
    0.0000    0.0001    0.0031    0.1299
    0.0001    0.0031    0.1299    5.5647

   1.0e+09 *

    0.0000
    0.0011
    0.0411
    1.6243

  356.8068
   97.9511
   -3.2382
    0.0449

   2.7674e+03

Elapsed time is 0.006915 seconds.
Relative error
    0.0720
```



```
Command Window
4th Degree Polynomial Regression
   1.0e+14 *

    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0006
    0.0000    0.0000    0.0000    0.0006    0.0243
    0.0000    0.0000    0.0006    0.0243    1.0799

   1.0e+10 *

    0.0000
    0.0001
    0.0041
    0.1624
    6.7339

  164.3372
  154.0886
   -7.7043
    0.1751
   -0.0013

   2.6704e+03

Elapsed time is 0.007749 seconds.
Relative error
    0.1046
```



Numerical Integration :

All the methods are included in a single code with choice of method

```
% **__Approximate Area__**
clc;clear all;close all;

p1 = -5.763e-11; p2 = 1.0612e-08; p3 = -6.6509e-07; p4 = 8.3391e-06;
p5 = 0.00093141; p6 = -0.051046; p7 = 1.1734; p8 = -14.322;
p9 = 94.412; p10 = -252.56; p11 = 847.96;

F = @(x) p1*x.^10 + p2*x.^9 + p3*x.^8 + p4*x.^7 + p5*x.^6 + p6*x.^5 + p7*x.^4
+ p8*x.^3 + p9*x.^2 + p10*x + p11;

% Taking Inputs
a = input('Lower limit(x) = ');
b = input('Upper limit(x) = ');
value = input('What is given to you "h" or "n", type here : ','s');
if value == 'n'
    n = input('Enter no. of inervals : ');
    h = (b - a)/n;
elseif value == 'h'
    h = input('Enter the value of h : ');
    n = (b-a)/h;
else
    disp('Enter the correct value');
    break;
end
fprintf('\n');
X0 = a;
Xn = b;
Y0 = F(a);
Yn = F(b);

fprintf('Trapezoid rule : 1 \nSimpson 1/3 rule : 2 \nSimpson 3/8 rule : 3 \nWe
ddle rule : 4 \nBoole rule : 5 \nGaussian quadrature : 6\n\n');
V = input('Enter number accordingly for particular method : ');
fprintf('\n');

if V == 1
    % Trapezoid rule
    fprintf('***Trapezoid rule***\n\n');
    tic
    Y = 0;
    for i = 1:n-1
        Y = Y + F(a + (i * h));
    end
    Y = 2 * Y;
```

```
    I = (h / 2)*(Y0 + Y + Yn);
    toc
    disp('The value of approximate Area is ');
    disp(I);

elseif V == 2
    % Simpson's 1/3
    fprintf('***Simpson 1/3 rule***\n\n');
    tic
    Yo = 0;
    Ye = 0;
    for i = 1:2:n-1
        Yo = Yo + F(a + (i * h));
    end
    for i = 2:2:n-1
        Ye = Ye + F(a + (i * h));
    end
    I = (h / 3)*(Y0 + (4 * Yo)+ (2 * Ye) + Yn);
    toc
    disp('The value of approximate Area is ');
    disp(I);

elseif V == 3
    % Simpson's 3/8
    fprintf('***Simpson 3/8 rule***\n\n');
    tic
    Yo = 0;
    Y3 = 0;
    for i = 1:n-1
        if rem(i,3) == 0
            Y3 = Y3 + F(a + (i * h));
        else
            Yo = Yo + F(a + (i * h));
        end
    end
    I = ((3 * h) / 8)*(Y0 + (3 * Yo)+ (2 * Y3) + Yn);
    toc
    disp('The value of approximate Area is ');
    disp(I);

elseif V == 4
    % Weddle's rule
    fprintf('***Weddle rule***\n\n');
    tic
    Y = 0;
    for i = 6:6:n-1
        Y = Y + (F(a+i*h) + 5*F(a+(i+1)*h) + F(a+(i+2)*h) + 6*F(a+(i+3)*h) + F
(a+(i+4)*h)+ 5*F(a+(i+5)*h) + F(a+(i+6)*h));
```

```
    end
    I = ((3 * h) / 10)*((Y0 + 5*F(a+h) + F(a+2*h) + 6*F(a+3*h) + F(a+4*h)+ 5*F
(a+5*h) + F(a+6*h)) + Y + Yn);
    toc
    disp('The value of approximate Area is ');
    disp(I);

elseif V == 5
    % Boole's rule
    fprintf('***Boole rule method***\n\n');
    Y = 0;
    for i = 4:4:n-1
        Y = Y + (7*F(a+i*h) + 32*F(a+(i+1)*h) + 12*F(a+(i+2)*h) + 32*F(a+(i+3)
*h) + 7*F(a+(i+4)*h));
    end
    I = ((2 * h) / 45)*((7*Y0 + 32*F(a+h) + 12*F(a+2*h) + 32*F(a+3*h) + 7*F(a+
4*h) + Y + Yn));
    disp('The value of approximate Area is ');
    disp(I);

elseif V == 6
    % Gaussian quadrature method
    fprintf('***Gaussina quadrature method***\n\n');
    disp('Correct upto 11 degree with 6 no. of points'); % i = 2*n -
 1; i = degree, n = no. of points
    disp('Limits are converted to (-1, 1)');
    disp('Here x is replaced by (j + k*t) with "t" as variable and "j,k" as co
nstant');
    fprintf('\n');
    xa = 0.9324695142; wa = 0.1713244924;
    xb = 0.6612093865; wb = 0.3607615730;
    xc = 0.2386191861; wc = 0.4679139346;
    tic
    j = (b + a)/2;
    k = (b - a)/2;
    F = @(t) p1*(j + k*t).^10 + p2*(j + k*t).^9 + p3*(j + k*t).^8 + p4*(j + k*
t).^7 + p5*(j + k*t).^6 + p6*(j + k*t).^5 + p7*(j + k*t).^4 + p8*(j + k*t).^3
+ p9*(j + k*t).^2 + p10*(j + k*t) + p11;

    I = k*(wa*(F(xa)+F(-xa)) + wb*(F(xb)+F(-xb)) + wc*(F(xc)+F(-xc)));
    toc
    disp('The value of approximate Area is ');
    disp(I);

else
    disp('Enter the correct value');
    break;
end
```

```
syms x;
G = int(p1*x.^10 + p2*x.^9 + p3*x.^8 + p4*x.^7 + p5*x.^6 + p6*x.^5 + p7*x.^4 +
 p8*x.^3 + p9*x.^2 + p10*x + p11,x,a,b); %Actual value
disp('The integrated value of approximate Area is ');
disp(double(G));
Relative_Error = abs(double(G) - I)/abs(double(G)) ;
fprintf('Relative Error is %d\n',Relative_Error);
```

Command Window

```
Lower limit(x) = 37
Upper limit(x) = 47
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12

Trapezoid rule : 1
Simpson 1/3 rule : 2
Simpson 3/8 rule : 3
Weddle rule : 4
Boole rule : 5
Gaussian quadrature : 6

Enter number accordingly for particular method : 1

***Trapezoid rule***

Elapsed time is 0.034701 seconds.
The value of approximate Area is
   2.1247e+04

The integrated value of approximate Area is
   2.1246e+04

Relative Error is 3.943646e-05
>>
```

Command Window

```
Lower limit(x) = 37
Upper limit(x) = 47
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12

Trapezoid rule : 1
Simpson 1/3 rule : 2
Simpson 3/8 rule : 3
Weddle rule : 4
Boole rule : 5
Gaussian quadrature : 6

Enter number accordingly for particular method : 2

***Simpson 1/3 rule***

Elapsed time is 0.000403 seconds.
The value of approximate Area is
   2.1246e+04

The integrated value of approximate Area is
   2.1246e+04

Relative Error is 9.044447e-07
>>
```

Command Window

```
Lower limit(x) = 37
Upper limit(x) = 47
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12

Trapezoid rule : 1
Simpson 1/3 rule : 2
Simpson 3/8 rule : 3
Weddle rule : 4
Boole rule : 5
Gaussian quadrature : 6

Enter number accordingly for particular method : 3

***Simpson 3/8 rule***

Elapsed time is 0.000531 seconds.
The value of approximate Area is
   2.1246e+04

The integrated value of approximate Area is
   2.1246e+04

Relative Error is 2.263706e-06
fx >>
```

Command Window

```
Lower limit(x) = 37
Upper limit(x) = 47
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12

Trapezoid rule : 1
Simpson 1/3 rule : 2
Simpson 3/8 rule : 3
Weddle rule : 4
Boole rule : 5
Gaussian quadrature : 6

Enter number accordingly for particular method : 4

***Weddle rule***

Elapsed time is 0.001015 seconds.
The value of approximate Area is
   2.1791e+04

The integrated value of approximate Area is
   2.1246e+04

Relative Error is 2.563907e-02
fx >>
```

Command Window

```
Lower limit(x) = 37
Upper limit(x) = 47
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12

Trapezoid rule : 1
Simpson 1/3 rule : 2
Simpson 3/8 rule : 3
Weddle rule : 4
Boole rule : 5
Gaussian quadrature : 6

Enter number accordingly for particular method : 5

***Boole rule method***

The value of approximate Area is
   2.1327e+04

The integrated value of approximate Area is
   2.1246e+04

Relative Error is 3.798084e-03
>>
```

Command Window

```
Lower limit(x) = 37
Upper limit(x) = 47
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12

Trapezoid rule : 1
Simpson 1/3 rule : 2
Simpson 3/8 rule : 3
Weddle rule : 4
Boole rule : 5
Gaussian quadrature : 6

Enter number accordingly for particular method : 6

***Gaussina quadrature method***

Correct upto 11 degree with 6 no. of points
Limits are converted to (-1, 1)
Here x is replaced by (j + k*t) with "t" as variable and "j,k" as constant

Elapsed time is 0.011186 seconds.
The value of approximate Area is
   2.1246e+04

The integrated value of approximate Area is
   2.1246e+04

Relative Error is 7.184759e-13
```

Solution of Transcendental or Algebraic equations :

All the methods are included in a single code with choice of method

1. For finding Derivative of functions

```
%__**Finding differentiation of Function__**
clc;clear all;close all;
syms x;

%--D'MART--
p1 = -5.763e-11; p2 = 1.0612e-08; p3 = -6.6509e-07; p4 = 8.3391e-06;
p5 = 0.00093141; p6 = -0.051046; p7 = 1.1734; p8 = -14.322;
p9 = 94.412; p10 = -252.56; p11 = 847.96;

F = p1*x.^10 + p2*x.^9 + p3*x.^8 + p4*x.^7 + p5*x.^6 + p6*x.^5 + p7*x.^4 + p8*x.^3 + p9*x.^2 + p10*x + p11;

disp('1st derivative of F is DF'); DF = diff(F); disp(DF);
disp('2nd derivative of F is D2F'); D2F = diff(DF); disp(D2F);

%--WALMART--
p1 = -8.5697e-08;p2 = 1.5624e-05;p3 = -0.0011391;p4 = 0.042278;
p5 = -0.83936;p6 = 8.541;p7 = -37.431;p8 = 126.99;

f = p1*x.^7 + p2*x.^6 + p3*x.^5 + p4*x.^4 + p5*x.^3 + p6*x.^2 + p7*x + p8 ;

disp('1st derivative of f is Df'); Df = diff(f); disp(Df);
disp('2nd derivative of f is D2f'); D2f = diff(Df); disp(D2f);
```

Command Window

```
1st derivative of F is DF
- (22294526395005145*x^9)/38685626227668133590597632 + (28865521794938499*x^8)/3022314549036

2nd derivative of F is D2F
- (200650737555046305*x^8)/38685626227668133590597632 + (28865521794938499*x^7)/377789318629

1st derivative of f is Df
- (177053030211085*x^6)/295147905179352825856 + (13834172611566627*x^5)/14757395258967641292

2nd derivative of f is D2f
- (531159090633255*x^5)/147573952589676412928 + (69170863057833135*x^4)/14757395258967641292

>>
```

2. For maxima & minima

```
%__**Maxima & Minima by solution of Transcedental equation**__
clc;clear all;close all;

p1 = -5.763e-11; p2 = 1.0612e-08; p3 = -6.6509e-07; p4 = 8.3391e-06;
p5 = 0.00093141; p6 = -0.051046; p7 = 1.1734; p8 = -14.322;
p9 = 94.412; p10 = -252.56; p11 = 847.96;

F = @(x) p1*x.^10 + p2*x.^9 + p3*x.^8 + p4*x.^7 + p5*x.^6 + p6*x.^5 + p7*x.^4
+ p8*x.^3 + p9*x.^2 + p10*x + p11;

DF = @(x) -
(22294526395005145*x.^9)/38685626227668133590597632 + (28865521794938499*x.^8)
/302231454903657293676544 -
 (6281597448183545*x.^7)/1180591620717411303424 + (17228875272567987*x.^6)/295
147905179352825856 + (25772222846540721*x.^5)/4611686018427387904 -
 (36782519452600695*x.^4)/144115188075855872 + (5867*x.^3)/1250 -
 (21483*x.^2)/500 + (23603*x)/125 - 6314/25;

D2F = @(x) -
(25081342194380787*x.^8)/4835703278458516698824704 + (7216380448734625*x.^7)/9
444732965739290427392 -
 (43971182137284815*x.^6)/1180591620717411303424 + (12921656454425991*x.^5)/36
893488147419103232 + (8053819639543975*x.^4)/288230376151711744 -
 (4597814931575087*x.^3)/4503599627370496 + (17601*x.^2)/1250 -
 (21483*x)/250 + 23603/125;

x = 3:49;
plot(x, DF(x));
disp('We have found area form 37 to 47 so we will find maxima and minima in th
at region.');
disp('By the graph we can see that we are having 2 roots for the region.');
fprintf('\n');

for i = 37:46
    if DF(i)<0 && DF(i+1)>0
        h = i; k = i+1;
        disp('Function value changes from -ve to +ve');
        fprintf('This interval of root is in between %.0f to %.0f\n\n',i,i+1);
    elseif DF(i)>0 && DF(i+1)<0
        u = i; v = i+1;
        disp('Function value changes from +ve to -ve');
        fprintf('This interval of root is in between %.0f to %.0f\n\n',i,i+1);
    end
end

if h >= u
```

```
    fprintf('1st interval of root is form %.0f to %.0f\n\n',u,v);
else
    fprintf('1st interval of root is form %.0f to %.0f\n\n',h,k);
end
if h <= u
    fprintf('2nd interval of root is form %.0f to %.0f\n\n',u,v);
else
    fprintf('2nd interval of root is form %.0f to %.0f\n\n',h,k);
end

fprintf('Bisection : 1 \nRegula falsi : 2 \nSecant : 3 \nNewton Raphson : 4 \n
');
Value = input('Enter number accordingly for particular method : ');

% Taking Inputs
n = input('\nEnter no. of iterations : ');
a = zeros(1, n); b = zeros(1, n);
c = zeros(1, n); f = zeros(1, n);
a(1) = input('Enter value of a for which F(x)<0 : ');
b(1) = input('Enter value of b for which F(x)>0 : ');
fprintf('\n');

if Value == 1
    % Bisection method
    disp('Rate of convergence is **Linear**');
    tic
    for i = 1 : n
        c(i) = (a(i)+b(i))/2;
        f(i) = DF(c(i));
        if f(i) < 0
            a(i+1) = c(i);
        else
            a(i+1) = a(i);
        end
        if f(i) > 0
            b(i+1) = c(i);
        else
            b(i+1) = b(i);
        end
    end

    % Displaying values
    % disp('Values of "a"...');   disp(a);
    % disp('Values of "b"...');   disp(b);
    % disp('Values of "c"...');   disp(c);
    % disp('Values of function at point x = c ...');   disp(f);
    % sprintf('The approximate root is x = %f',c(n))
    % disp('Values of function at point x = c ...');   disp(f);
```

```
    fprintf('The approximate root is x = %f\n',c(n))
    toc
    fprintf('\n');

elseif Value == 2
    % Regula method
    disp('Rate of convergence is **1.618**');
    tic
    for i = 1 : n
        c(i) = (a(i)*DF(b(i)) - b(i)*DF(a(i)))/(DF(b(i)) - DF(a(i)));
        f(i) = DF(c(i));
        if f(i) < 0
            a(i+1) = c(i);
        else
            a(i+1) = a(i);
        end
        if f(i) > 0
            b(i+1) = c(i);
        else
            b(i+1) = b(i);
        end
    end

    % Displaying values
    % disp('Values of "a"...');   disp(a);
    % disp('Values of "b"...');   disp(b);
    % disp('Values of "c"...');   disp(c);
    % disp('Values of function at point x = c ...');   disp(f);
    fprintf('The approximate root is x = %f\n',c(n))
    toc
    fprintf('\n');

elseif Value == 3
    % Secant method
    disp('Rate of convergence is **1.618**');
    tic
    j = a(1); k = b(1);
    for i = 1 : n
        c(i) = (j*DF(k) - k*DF(j))/(DF(k) - DF(j));
        j = k;
        k = c(i);
    end

    % Displaying values
    % disp('Values of "a"...');   disp(j);
    % disp('Values of "b"...');   disp(k);
    % disp('Values of "c" i.e x2,x3,x4...');   disp(c);
    fprintf('The approximate root is x = %f\n',c(n))
```

```
    toc
    fprintf('\n');

elseif Value == 4
    % Newton's Raphson method
    disp('Rate of convergence is **2**');
    tic
    j = a(1);
    for i = 1 : n
        c(i) = j - (DF(j)/D2F(j));
        j = c(i);
    end

    % Displaying values
    % disp('Values of "a"...');   disp(j);
    % disp('Values of "c" i.e x1,x2,x3...');   disp(c);
    fprintf('The approximate root is x = %f\n',c(n))
    toc
    fprintf('\n');

else
    disp('Enter correct method no.');
    break;
end

%--Actual method--
disp('By actual method...');
tic
if h <= u
    R1 = fzero(DF,h); disp('1st root is...'); disp(R1);
    R2 = fzero(DF,u); disp('2nd root is...'); disp(R2);
else
    R1 = fzero(DF,u); disp('1st root is...'); disp(R1);
    R2 = fzero(DF,h); disp('2nd root is...'); disp(R2);
end
if D2F(R1) > 0
    fprintf('Minima at %f and value is %f\nMaxima at %f and value is %f\n',R1,
F(R1),R2,F(R2));
else
    fprintf('Minima at %f and value is %f\nMaxima at %f and value is %f\n',R2,
F(R2),R1,F(R1));
end
toc

Interval = input('\nWhich interval you used for finding root (1st or 2nd) Ente
r no : ');

%--Errors--
```

```
if Interval == 1
    fprintf('\nPercentage_Error in found root is %0.12f\n',abs(c(n)-
R1)*100/abs(R1));
elseif Interval == 2
    fprintf('\nPercentage_Error in found root is %0.12f\n',abs(c(n)-
R2)*100/abs(R2));
else
    disp('Enter correct interval no.');
    break;
end
```

Common part for all results.

```
Command Window
We have found area form 37 to 47 so we will find maxima and minima in that region.
By the graph we can see that we are having 2 roots for the region.

Function value changes from +ve to -ve
This interval of root is in between 40 to 41

Function value changes from -ve to +ve
This interval of root is in between 44 to 45

1st interval of root is form 40 to 41

2nd interval of root is form 44 to 45

Bisection : 1
Regula falsi : 2
Secant : 3
Newton Raphson : 4
```

Results :

```
Enter number accordingly for particular method : 1

Enter no. of iterations : 10
Enter value of a for which F(x)<0 : 41
Enter value of b for which F(x)>0 : 40

Rate of convergence is **Linear**
The approximate root is x = 40.618164
Elapsed time is 0.021709 seconds.

By actual method...
1st root is...
   40.6176

2nd root is...
   44.9025

Minima at 44.902506 and value is 2088.130866
Maxima at 40.617589 and value is 2172.157365
Elapsed time is 0.582155 seconds.

Which interval you used for finding root (1st or 2nd) Enter no : 1

Percentage_Error in found root is 0.001415994352
```

```
Enter number accordingly for particular method : 2

Enter no. of iterations : 10
Enter value of a for which F(x)<0 : 41
Enter value of b for which F(x)>0 : 40

Rate of convergence is **1.618**
The approximate root is x = 40.617589
Elapsed time is 0.001410 seconds.

By actual method...
1st root is...
   40.6176

2nd root is...
   44.9025

Minima at 44.902506 and value is 2088.130866
Maxima at 40.617589 and value is 2172.157365
Elapsed time is 0.099769 seconds.

Which interval you used for finding root (1st or 2nd) Enter no : 1

Percentage_Error in found root is 0.000000000005
```

```
Enter number accordingly for particular method : 3

Enter no. of iterations : 10
Enter value of a for which F(x)<0 : 41
Enter value of b for which F(x)>0 : 40

Rate of convergence is **1.618**
The approximate root is x = 40.617589
Elapsed time is 0.001377 seconds.

By actual method...
1st root is...
   40.6176

2nd root is...
   44.9025

Minima at 44.902506 and value is 2088.130866
Maxima at 40.617589 and value is 2172.157365
Elapsed time is 0.102325 seconds.

Which interval you used for finding root (1st or 2nd) Enter no : 1

Percentage_Error in found root is 0.000000000004
```

```
Enter number accordingly for particular method : 4

Enter no. of iterations : 40
Enter value of a for which F(x)<0 : 41
Enter value of b for which F(x)>0 : 40

Rate of convergence is **2**
The approximate root is x = 40.617589
Elapsed time is 0.002974 seconds.

By actual method...
1st root is...
   40.6176

2nd root is...
   44.9025

Minima at 44.902506 and value is 2088.130866
Maxima at 40.617589 and value is 2172.157365
Elapsed time is 0.101404 seconds.

Which interval you used for finding root (1st or 2nd) Enter no : 1

Percentage_Error in found root is 0.000000000013
```

Solution of Differential Equations :

All the methods are included in a single code with choice of method

```
%__**Minima and Maxima by another methods**__
clc; clear all; close all;

format long
F = @(x,y) -
(22294526395005145*x.^9)/38685626227668133590597632 + (28865521794938499*x.^8)
/302231454903657293676544 -
 (6281597448183545*x.^7)/1180591620717411303424 + (17228875272567987*x.^6)/295
147905179352825856 + (25772222846540721*x.^5)/4611686018427387904 -
 (36782519452600695*x.^4)/144115188075855872 + (5867*x.^3)/1250 -
 (21483*x.^2)/500 + (23603*x)/125 - 6314/25;
disp('We are calculating for the function given below...');
disp(F);
syms y(x);

ode = diff(y) == -
(22294526395005145*x.^9)/38685626227668133590597632 + (28865521794938499*x.^8)
/302231454903657293676544 -
 (6281597448183545*x.^7)/1180591620717411303424 + (17228875272567987*x.^6)/295
147905179352825856 + (25772222846540721*x.^5)/4611686018427387904 -
 (36782519452600695*x.^4)/144115188075855872 + (5867*x.^3)/1250 -
 (21483*x.^2)/500 + (23603*x)/125 - 6314/25;
cond = y(input('Enter the value of x for given condition of ODE: ')) == input(
'Enter value of y for value of x you entered : ');
ySol(x) = dsolve(ode,cond);
disp('Simplified ODE is...');
disp(ySol(x));

format long
g = double(ySol(input('Enter value of x for which you want actual value of ODE
 : ')));
tf = isa(g,'double');
if tf == 1
    G = g;
else
    G = cell2mat(g);
end
disp('Actual value is ');
disp(G);

a = input('Enter Lower limit : ');
b = input('Enter upper limit : ');
value = input('What is given to you "h" or "n", type here : ','s');
if value == 'n'
```

```
    n = input('Enter no. of inervals : ');
    h = (b - a)/n;
elseif value == 'h'
    h = input('Enter the value of h : ');
    n = (b-a)/h;
else
    disp('Enter the correct value');
    break;
end
X = linspace(a,b,n+1);
Y(1) = input('Enter the value of Y(initial) given for X(initial) : ');
fprintf('\nPress 1 for "Eulers method"\nPress 2 for "Eulers modified method"\n
Press 3 for "Runge Kutta 2nd order method"\nPress 4 for "Runge Kutta 4th order
 method"');
c = input('\n\nEnter the number = ');

%__Euler's Method--
if c == 1
    tic
    for i = 1:n
        Y(i+1)= Y(i) + h*F(X(i),Y(i));
    end
    disp('Values of X starting from X(1) to X(n+1) are...');
    disp(X);
    disp('Values of Y starting from Y(1) to Y(n+1) are...');
    disp(Y);
    sprintf('The value of Y(%.2f) is %f',X(n+1),Y(n+1))
    Absolute_Error = abs(Y(n+1) - G);
    Percent_Error = Absolute_Error * 100/abs(G);
    Relative_Error = Absolute_Error/abs(G);
    disp('Absolute_Error, Percent_Error, Relative_Error are given below respec
tively');
    disp(Absolute_Error);
    disp(Percent_Error);
    disp(Relative_Error);
    toc
    fprintf('\n');
    disp('Value for root 40.6176 is 2172.16');
    disp('Value for root 44.9025 is 2088.13');
    break;

%__Euler's Modified Method--
elseif c == 2
    tic
    yp(1) = Y(1);
    for i = 1:n
        yp(i+1)= Y(i) + h*F(X(i),Y(i));
        Y(i+1) = Y(i) + (h/2)*(F(X(i),Y(i)) + F(X(i+1),yp(i+1)));
```

```
    end
    disp('Values of X starting from X(1) to X(n+1) are...');
    disp(X);
    disp('Values of Y*(predicted) starting from yp(1) to yp(n+1) are...');
    disp(yp);
    disp('Values of Y starting from Y(1) to Y(n+1) are...');
    disp(Y);
    sprintf('The value of Y(%0.2f) is %f',X(n+1),Y(n+1))
    Absolute_Error = abs(Y(n+1) - G);
    Percent_Error = Absolute_Error * 100/abs(G);
    Relative_Error = Absolute_Error/abs(G);
    disp('Absolute_Error, Percent_Error, Relative_Error are given below respec
tively');
    disp(Absolute_Error);
    disp(Percent_Error);
    disp(Relative_Error);
    toc
    fprintf('\n');
    disp('Value for root 40.6176 is 2172.16');
    disp('Value for root 44.9025 is 2088.13');
    break;

%__Runge Kutta's 2nd order Method--
elseif c == 3
    tic
    for i = 1:n
        k1(i) = h*F(X(i),Y(i));
        k2(i) = h*F(X(i) + h, Y(i) + k1(i));
        Y(i+1) = Y(i) + (k1(i) + k2(i))/2;
    end
    disp('Values of X starting from X(1) to X(n+1) are...');
    disp(X);
    disp('Values of k1 starting from k1(1) to k1(n) are...');
    disp(k1);
    disp('Values of k2 starting from k2(1) to k2(n) are...');
    disp(k2);
    disp('Values of Y starting from Y(1) to Y(n+1) are...');
    disp(Y);
    sprintf('The value of Y(%0.2f) is %f',X(n+1),Y(n+1))
    Absolute_Error = abs(Y(n+1) - G);
    Percent_Error = Absolute_Error * 100/abs(G);
    Relative_Error = Absolute_Error/abs(G);
    disp('Absolute_Error, Percent_Error, Relative_Error are given below respec
tively');
    disp(Absolute_Error);
    disp(Percent_Error);
    disp(Relative_Error);
    toc
```

```
    fprintf('\n');
    disp('Value for root 40.6176 is 2172.16');
    disp('Value for root 44.9025 is 2088.13');
    break;

%__Runge Kutta's 4th order Method--
elseif c == 4
    tic
    for i = 1:n
        k1(i) = h*F(X(i),Y(i));
        k2(i) = h*F(X(i) + h/2, Y(i) + k1(i)/2);
        k3(i) = h*F(X(i) + h/2, Y(i) + k2(i)/2);
        k4(i) = h*F(X(i) + h, Y(i) + k3(i));
        Y(i+1) = Y(i) + (k1(i) + k4(i) + 2*(k2(i) + k3(i)))/6;
    end
    disp('Values of X starting from X(1) to X(n+1) are...');
    disp(X);
    disp('Values of k1 starting from k1(1) to k1(n) are...');
    disp(k1);
    disp('Values of k2 starting from k2(1) to k2(n) are...');
    disp(k2);
    disp('Values of k3 starting from k3(1) to k3(n) are...');
    disp(k3);
    disp('Values of k4 starting from k4(1) to k4(n) are...');
    disp(k4);
    disp('Values of Y starting from Y(1) to Y(n+1) are...');
    disp(Y);
    sprintf('The value of Y(%0.2f) is %f',X(n+1),Y(n+1))
    Absolute_Error = abs(Y(n+1) - G);
    Percent_Error = Absolute_Error * 100/abs(G);
    Relative_Error = Absolute_Error/abs(G);
    disp('Absolute_Error, Percent_Error, Relative_Error are given below respec
tively');
    disp(Absolute_Error);
    disp(Percent_Error);
    disp(Relative_Error);
    toc
    fprintf('\n');
    disp('Value for root 40.6176 is 2172.16');
    disp('Value for root 44.9025 is 2088.13');
    break;

else
    disp('***Enter the correct number..!***');
    break;
end
```

Common Part for all results :

```
Command Window
We are calculating for the function given below...
    @(x,y)-(22294526395005145*x.^9)/38685626227668133590597632+(28865521794938499*x.^8)/30

Enter the value of x for given condition of ODE: 37
Enter value of y for value of x you entered : 1916.85
Simplified ODE is...
- (2786815799375643*x^10)/48357032784585166988247040 + (7216380448734625*x^9)/680020773533

Enter value of x for which you want actual value of ODE : 40.6176
Actual value is
     2.069160465523398e+03

Enter Lower limit : 37
Enter upper limit : 40.6176
What is given to you "h" or "n", type here : n
Enter no. of inervals : 12
Enter the value of Y(initial) given for X(initial) : 1916.85

Press 1 for "Eulers method"
Press 2 for "Eulers modified method"
Press 3 for "Runge Kutta 2nd order method"
Press 4 for "Runge Kutta 4th order method"
```

Results :

```
Enter the number = 1
Values of X starting from X(1) to X(n+1) are...
Values of Y starting from Y(1) to Y(n+1) are...

ans =

The value of Y(40.62) is 2081.196779

Absolute_Error, Percent_Error, Relative_Error are given below respectively
  12.036313230685209

   0.581700328767909

   0.005817003287679

Elapsed time is 0.011608 seconds.

Value for root 40.6176 is 2172.16
Value for root 44.9025 is 2088.13
```

```
Enter the number = 2

ans =

The value of Y(40.62) is 2069.114542

Absolute_Error, Percent_Error, Relative_Error are given below respectively
   0.045923987339847

   0.002219450260385

     2.219450260385233e-05

Elapsed time is 0.009316 seconds.
```

```
Enter the number = 3

ans =

The value of Y(40.62) is 2069.114542

Absolute_Error, Percent_Error, Relative_Error are given below respectively
   0.045923987368496

   0.002219450261770

     2.219450261769809e-05

Elapsed time is 0.010603 seconds.
```

```
Enter the number = 4

ans =

The value of Y(40.62) is 2069.160474

Absolute_Error, Percent_Error, Relative_Error are given below respectively
     8.195515874831472e-06

     3.960792800455135e-07

     3.960792800455135e-09

Elapsed time is 0.014807 seconds.
```

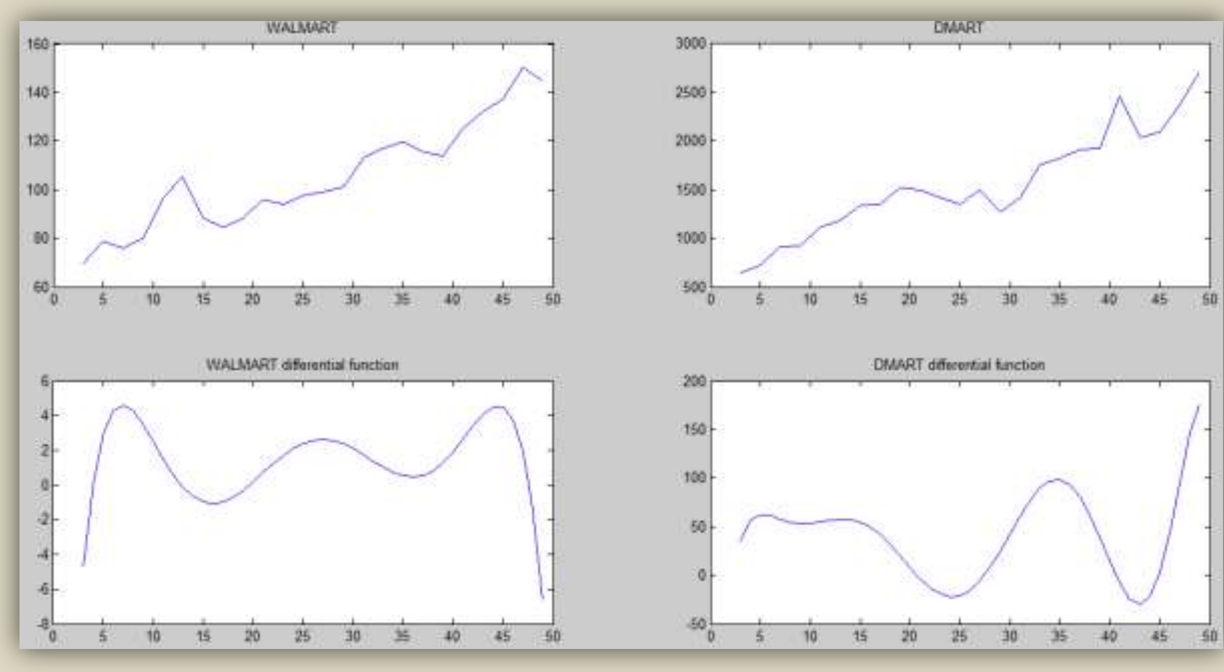
Comparison :

```
%__**Comparison**__
clc;clear all;close all;

X = (3:2:49);

disp('Here we will be finding areas for profit comprison in interval of [37,47
]');
fprintf('\n');

%--WALMART--
Y_ = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\DATA-
SHEET.xlsx','Sheet2','B1:B24');

%--D'MART--
Y = xlsread('C:\Users\lenovo\Desktop\STUDIES\Matlab\DATA-
SHEET.xlsx','Sheet1','B1:B24');

x = 3:49;

P1 = -8.5697e-08;P2 = 1.5624e-05;P3 = -0.0011391;P4 = 0.042278;P5 = -
0.83936;P6 = 8.541;P7 = -37.431;P8 = 126.99;
f = @(x) P1*x.^7 + P2*x.^6 + P3*x.^5 + P4*x.^4 + P5*x.^3 + P6*x.^2 + P7*x + P8
 ;
Df = @(x) -
(177053030211085*x.^6)/295147905179352825856 + (13834172611566627*x.^5)/147573
952589676412928 -
 (13132928858976595*x.^4)/2305843009213693952 + (6092901921471035*x.^3)/360287
97018963968 - (7869*x.^2)/3125 + (8541*x)/500 - 37431/1000;
D2f = @(x) -
(531159090633255*x.^5)/147573952589676412928 + (69170863057833135*x.^4)/147573
952589676412928 -
 (13132928858976595*x.^3)/576460752303423488 + (18278705764413105*x.^2)/360287
97018963968 - (15738*x)/3125 + 8541/500;

p1 = -5.763e-11; p2 = 1.0612e-08;p3 = -6.6509e-07; p4 = 8.3391e-06;
p5 = 0.00093141; p6 = -0.051046;p7 = 1.1734; p8 = -14.322;
p9 = 94.412; p10 = -252.56; p11 = 847.96;
F = @(x) p1*x.^10 + p2*x.^9 + p3*x.^8 + p4*x.^7 + p5*x.^6 + p6*x.^5 + p7*x.^4
+ p8*x.^3 + p9*x.^2 + p10*x + p11;
DF = @(x) -
(2229452639500514 5*x.^9)/38685626227668133590597632 + (28865521794938499*x.^8)
/302231454903657293676544 -
 (6281597448183545*x.^7)/1180591620717411303424 + (17228875272567987*x.^6)/295
147905179352825856 + (2577222846540721*x.^5)/4611686018427387904 -
```

```
 (36782519452600695*x.^4)/144115188075855872 + (5867*x.^3)/1250 -
 (21483*x.^2)/500 + (23603*x)/125 - 6314/25;
D2F = @(x) -
(25081342194380787*x.^8)/483570327845851669882470 + (7216380448734625*x.^7)/9
444732965739290427392 -
 (43971182137284815*x.^6)/1180591620717411303424 + (12921656454425991*x.^5)/36
893488147419103232 + (8053819639543975*x.^4)/288230376151711744 -
 (4597814931575087*x.^3)/4503599627370496 + (17601*x.^2)/1250 -
 (21483*x)/250 + 23603/125;

subplot(2,2,1); % first 2 is for parts of height__2nd 2 is for part of width__
3rd 1 is for position
plot(X, Y_);
title('WALMART');
subplot(2,2,2);
plot(X, Y);
title('DMART');
subplot(2,2,3);
plot(x, Df(x));
title('WALMART differential function');
subplot(2,2,4);
plot(x, DF(x));
title('DMART differential function');

syms x;
G = int(P1*x.^7 + P2*x.^6 + P3*x.^5 + P4*x.^4 + P5*x.^3 + P6*x.^2 + P7*x + P8,
x,37,47); %WALMART
disp('FOR WALMART AREA... '); disp(double(G));
H = int(p1*x.^10 + p2*x.^9 + p3*x.^8 + p4*x.^7 + p5*x.^6 + p6*x.^5 + p7*x.^4 +
 p8*x.^3 + p9*x.^2 + p10*x + p11,x,37,47); %D'MART
disp('FOR DMART AREA... '); disp(double(H));

if G > H
    disp('WALMART is more profitable for investers');
elseif G<H
    disp('DMART is more profitable for investers');
else
    disp('Both are equal profitable for this interval');
end
```

Command Window

```
Here we will be finding areas for profit comprison in interval of [37,47]

FOR WALMART AREA...
     1.271055069276990e+03

FOR DMART AREA...
     2.124630729611951e+04

DMART is more profitable for investers
>>
```



__THANK YOU__