# **Numerical Methods Final Lab Assignment**

**Q1) Newton-Raphson**

f = @(x)x^3-0.165\*x^2+3.993\*10^-4;

df = @(x)3\*x^2-0.33\*x;

x = 0.05;

for i=1:1:3

x1 = x - (f(x)/df(x));

x = x1;

end

fprintf('Approximate Root is %.04f',x1)

x2=[-1:0.005:1];

y = x2.^3-0.165\*x2.^2+3.993\*10^-4;

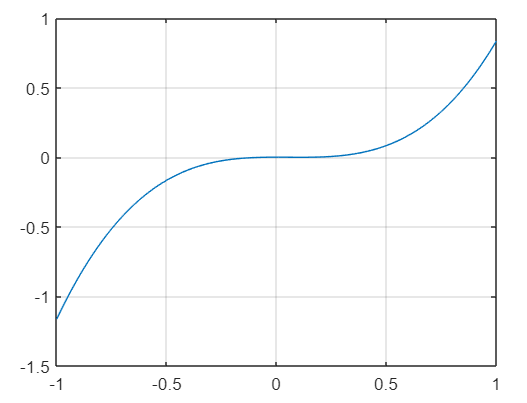
plot(x2,y)

grid on

**Output:**

>>newton\_raphson

Approximate Root is 0.0624



**Q2) Secant Method**

syms x;

y(x)=x^3-2\*x-5;

x0=2;

x1=3;

while abs(x0-x1)>0.0001

a=y(x0);

b=y(x1);

x2=((x0\*b)-(x1\*a))/(b-a);

x0=x1;

x1=x2;

end

fprintf('Approximate Root is %.04f',x2)

**Output:**

Approximate Root is 2.0946

**Q3) Gauss-Seidel**

syms x1 x2 x3;

x\_1(x2,x3) = (58-(2\*x2+3\*x3))/45;

x\_2(x1,x3) = (47+(3\*x1-2\*x3))/22;

x\_3(x1,x2) = (67-(5\*x1+x2))/20;

a=0;

b=0;c=0;

val =0;

while val == 0

x1\_new = x\_1(b,c);

x2\_new = x\_2(x1\_new,c);

x3\_new = x\_3(x1\_new,x2\_new);

if abs(x1\_new-a)<0.0001 && abs(x2\_new-b)<0.0001 && abs(x3\_new-c)<0.0001

val = 1;

else

a = x1\_new

b = x2\_new

c = x3\_new

end

end

a =

4312446862657/4312440000000

=1.0000

b =

189746392038481/94873680000000

= 2.0000

c =

632491223674361/210830400000000

= 3.0000

**Q4) Power Method**

format long;

A = [1 2;3 4];

v0 = [1;1];

val = 0;

while val==0

y1 = A\*v0;

v1 = y1./max(y1);

lambda = y1./v0;

if abs(lambda(1)-lambda(2))<0.00005

val =1;

else

v0 = v1;

end

end

disp (y1)

**Output:**

power

2.457426432178304

5.372279296534912

**Q5) Newton Forward Interpolation**

syms x;

x\_val = [0 1 2 3];

f\_val = [-1 1 1 2];

x0 = 0;

x1 = 1;

x2 = 2;

x3 = 3;

d = divdiff(x\_val,f\_val)

f(x) = d(1,1)+(x-x0)\*d(1,2)+(x-x0)\*(x-x1)\*d(1,3)+(x-x0)\*(x-x1)\*(x-x2)\*d(1,4)

val = f(1.5)

function TDD = divdiff(X, Y)

if nargin ~= 2

error('divdiff: invalid input parameters');

end

[ p, m ] = size(X); % m points, polynomial order <= m-1

if p ~= 1 || p ~=size(Y, 1) || m ~= size(Y, 2)

error('divdiff: input vectors must have the same dimension');

end

TDD = zeros(m, m);

TDD(:, 1) = Y';

for j = 2 : m

for i = 1 : (m - j + 1)

TDD(i,j) = (TDD(i + 1, j - 1) - TDD(i, j - 1)) / (X(i + j - 1) - X(i));

end

end

end

**Output:**

>>newton

d =

-1.0000 2.0000 -1.0000 0.5000

1.0000 0 0.5000 0

1.0000 1.0000 0 0

2.0000 0 0 0

f(x) =

2\*x - x\*(x - 1) + (x\*(x - 1)\*(x - 2))/2 - 1

val =

1.0625 **% f(1.5)**

**Q6) Lagrange Interpolation**

syms x

x0 = 0.1;

x1 = 0.2;

f0 = 0.09983;

f1 = 0.19867;

l0 = (x-x1)/(x0-x1);

l1 = (x-x0)/(x1-x0);

f(x) = (l0\*f0)+(l1\*f1);

disp(f(x))

f(0.15)

**Output:**

>>lagrange

(2471\*x)/2500 + 99/100000 % Polynomial

ans =

0.1492 % sin(0.15)

**Q7) Cubic Spline Interpolation**

x = [0 1 2];

y = [-1 3 29];

spline(x,y,0.5)

spline(x,y,1.5)

**OUTPUT:**

>> Splines

ans =

-1.7500

ans =

13.2500

**Q8) Newton Forward Difference formula**

syms x0 x;

t=[3.0 3.2 3.4 3.6 3.8 4.0];

o=[-14 -10.032 -5.296 -0.256 6.672 14 ];

dt=zeros(6,7);

siz = length(t)

for i=1:6

dt(i,1)=t(i);

dt(i,2)=o(i);

end

n=5;

for j=3:7

for i=1:n

dt(i,j)=dt((i+1),(j-1))-dt(i,(j-1))

end

n=n-1;

end

h=t(2)-t(1)

sq = h\*h;

x = 3;

diff = 0;

%i = input('Enter the value:');

x0 =dt(i,1);

s = (x-x0)/h

f1 = 1/h\*(dt(i,3)+0.5\*((2\*s)-1)\*(dt(i,4))+0.166\*((3\*s\*s)-(6\*s)+2)\*(dt(i,5))+0.0416\*((4\*s\*s\*s)-(18\*s\*s)+(22\*s)-6)\*(dt(i,6))+0.2\*(dt(i,7)))

f2 = 1/sq\*(dt(i,4)-dt(i,5)+0.9166\*(dt(i,6))-0.833\*(dt(i,7)))

**OUTPUT:**

dt =

3.0000 -14.0000 3.9680 0.7680 -0.4640 2.0480 -5.1200

3.2000 -10.0320 4.7360 0.3040 1.5840 -3.0720 0

3.4000 -5.2960 5.0400 1.8880 -1.4880 0 0

3.6000 -0.2560 6.9280 0.4000 0 0 0

3.8000 6.6720 7.3280 0 0 0 0

4.0000 14.0000 0 0 0 0 0

h =

0.2000

s =

0

f1 =

9.4739

f2 =

184.3539

**Q9) Gauss one point, two point, three point**

clc;

syms x;

f = @(x) 2\*x/(1+x^4);

a = 1;

b = 2;

if a~=-1 || b~=1

y= flege(f,a,b);

disp(y)

G=subs(y,x);

else

G=feval(f,x);

end

fprintf("Gauss one-point:")

g1 = 2\*subs(y,0)

fprintf("Gauss two-point:")

g2 = subs(y,-1/sqrt(3)) + subs(y,1/sqrt(3))

fprintf("Gauss three-point:")

g3 = 1/9\*(5\*subs(y,-sqrt(3/5))+8\*subs(y,0) + 5\*subs(y,sqrt(3/5)))

function y=flege(f,a,b)

%Performs variable change if a=!-1 y b=!1

syms x;

x=((b-a)./2).\*x+(b+a)./2;

dx=(b-a)./2;

y=feval(f,x)\*dx;

end

I = atan(4) - (pi/4)

**Output:**

y =

(x + 3)/(2\*((x/2 + 3/2)^4 + 1))

Gauss one-point:

g1 =

0.4948

Gauss two-point:

g2 =

0.5434

Gauss three-point:

g3 =

0.5406

I =

0.5404

**Q10) Picard’s Method**

syms x ;

f = @(x,y) x-y.^2;

x0 = 0;

y0 = 1;

for i=1:1:2

y1 = 1 + int(f(x,y0),x0,x)

y0 = y1;

end

**Output:**

y1 =

(x\*(x - 2))/2 + 1

y1 =

1 - (x\*(3\*x^4 - 15\*x^3 + 40\*x^2 - 90\*x + 60))/60 (% y2)

**Q11) Runge-kutta 4th Order**

syms x ;

f = @(x,y) -2\*x\*y^2;

x0 = 0;

y0 = 1;

h = 0.2;

for i=1:1:2

k1 = h\*f(x0,y0);

k2 = h\*f(x0+(h/2),y0+(k1/2));

k3 = h\*f(x0+(h/2),y0+(k2/2));

k4 = h\*f(x0+h,y0+k3);

y1 = y0 + (k1+2\*k2+2\*k3+k4)/6

y0 = y1;

x0 = x0 + h;

end

y = @(x) 1/(1+x^2);

fprintf("Exact solutions");

y(0.2)

y(0.4)

**Output:**

>>runge

%y(0.2)

y1 =

0.9615

%y(0.4)

y1 =

0.8621

Exact solutions

%y(0.2)

ans =

0.9615

%y(0.4)

ans =

0.8621

**Q12) Square mesh**

clc;

h = 1/3;

n = 1/h;

x = 0:h:1;

y = 0:h:1;

u(4,1) = 0;

% Setting Boundary Conditions

for i=1:n+1

u(i,1) = -y(i);

u(1,i) = 2\*x(i);

u(i,n+1) = 2-y(i);

u(n+1,i) = 2\*x(i)-1;

end

% Filling the remaining mesh points

% By using standard 5-point formula

for j = 2:n

for i=2:n

u(i,j) = 1/4\*(u(i-1,j)+u(i,j+1)+u(i+1,j)+u(i,j-1));

end

end

% the points are printed from bottom to top if you see from axis point of

% view i.e., 1st row in u are points on x-axis

disp(u)

**Output:**

% the interior points are initial approximations

>>mesh\_grid

0 0.6667 1.3333 2.0000

-0.3333 0.0833 0.7708 1.6667

-0.6667 -0.2292 0.5521 1.3333

-1.0000 -0.3333 0.3333 1.0000

**Q13) Crank-Nikolson Implicit Finite Difference Method**

clc;

% Parameters

dx=0.5;

dt=0.25;

La= dt/(dx)^2; % lambda

L = 2; % upper limit of x

nx= ((L-0)/dx) + 1;

nt=2; % to compute 1 time steps k=[1,2]

% --- Constant Coefficients of the tridiagonal Matrix

b = La; % Super diagonal: coefficients of u(i+1)

c = b; % Subdiagonal: coefficients of u(i-1)

a = 2\*(1+La); % Main Diagonal: coefficients of u(i)

% Boundary conditions and Initial Condition

Uo(1)=0; Uo(nx)=0;

x=linspace(0,L,nx);

f0 = sin((pi.\*x)/2);

Uo(1:nx)= (f0)';

Un(1)=0; Un(nx)=0;

% Store results for future use

UUU(1,:)=Uo;

% Loop over time

for k=2:nt

for ii=1:nx-2

if ii==1

d(ii)=c\*Uo(ii)+2\*(1-c)\*Uo(ii+1)+b\*Uo(ii+2)+c\*Un(1);

elseif ii==nx-2

d(ii)=c\*Uo(ii)+2\*(1-c)\*Uo(ii+1)+b\*Uo(ii+2)+b\*Un(nx);

else

d(ii)=c\*Uo(ii)+2\*(1-c)\*Uo(ii+1)+b\*Uo(ii+2);

end

end % d is a row vector

% Transform a, b, c scalar constants in column vectors:

bb=b\*ones(nx-3,1);

cc=bb;

aa=a\*ones(nx-2,1);

% Use column vectors to construct diagonal matrices

AA=diag(aa)+ diag(-bb,1)+ diag(-cc,-1); %AA is one triadiagonal Matrix

% Find the solution for interior nodes i=2,3,4,5

UU=AA\d'; % UU is temp at interior nodes only

% Build the whole solution by including BCs

Un=[Un(1),UU',Un(nx)]; % row vector

% Store results for future use

UUU(k,:)=Un;

% to start over

Uo=Un;

end

UUU % Output

**Output:**

UUU =

0 0.7071 1.0000 0.7071 0.0000

0 0.3867 0.5469 0.3867 0