**ASSIGNMENT- 6: ROOTS OF NON-LINEAR EQUATIONS**

1. Write a menu-driven program for finding roots of a nonlinear equation using Bisection, Regula Falsi and Newton-Raphson method.

Code:

#include<stdio.h>

#include<math.h>

float \_y(float a){

float res = pow(a,2)-11\*a+30;

return res;

}

float \_f1(float a){

float res=2\*a-11;

return res;

}

void bisection(){

float x,y,e,y0=0,y1=0,x2=0,y2=0;

int n;

printf("Enter the boundary limits: ");

scanf("%f %f",&x,&y);

printf("Enter precision: ");

scanf("%f",&e);

printf("Enter max. number of iterations: ");

scanf("%d",&n);

y0 = \_y(x);

y1 = \_y(y);

for (int i=0;i<n;i++){

x2=(x+y)/2;

y2=\_y(x2);

if (fabs(y2)<=e){

printf("Converegent solution x=%lf f(x)=%lf\n",x2,y2);

return;

}

else if ((y0\*y2)>0){

x=x2;

y0=y2;

}

else{

y=x2;

y1=y2;

}

}

printf("Solution does not converge in %d iterations\n",n);

printf("%lf %lf\n",x2,y2);

}

void regular\_falsi(){

float x0,x1,e,x2,y2,y0,y1;

int n;

printf("Enter the boundary limits: ");

scanf("%f %f",&x0,&x1);

printf("Enter precision: ");

scanf("%f",&e);

printf("Enter number of iterations: ");

scanf("%d",&n);

y0 = \_y(x0);

y1 = \_y(x1);

for (int i=0;i<n;i++){

x2=(x0\*y1-x1\*y0)/(y1-y0);

y2=\_y(x2);

if (fabs(y2)<=e){

printf("Convergent Solution x=%lf f(x)=%lf\n",x2,y2);

return;

}

if (y0\*y2<0){

x1=x2;

y1=y2;

}

else{

x0=x2;

y0=y2;

}

}

printf("Solution does not converge\n");

printf("%lf %lf",x2,y2);

}

void newton(){

float x0,ep,d,y0,y0d,x1;

int n;

printf("Enter root's neighbour: ");

scanf("%f",&x0);

printf("Enter slope minima: ");

scanf("%f",&d);

printf("Enter precision: ");

scanf("%f",&ep);

printf("Enter number of iterations: ");

scanf("%d",&n);

for (int i=0;i<n;i++){

y0=\_y(x0);

y0d=\_f1(x0);

if (fabs(y0d)<=d){

printf("Slope too small %lf %lf\n",x0,y0d);

return;

}

x1=x0-(y0/y0d);

if (fabs((x1-x0)/x1)<=ep){

printf("Converegent Solutions x=%lf f(x)=%lf",x1,\_y(x1));

return;

}

x0=x1;

}

printf("Solution does not converge in %d iterations\n",n);

printf("%lf %lf %lf %lf",y0,y0d,x0,x1);

}

int main(){

printf("1. Bisection Method\n2. Regula Falsi Method\n3.Newton Raphson Method\n");

int n;

printf("Enter your choice: ");

scanf("%d",&n);

switch(n){

case 1 :bisection();break;

case 2 :regular\_falsi();break;

case 3 :newton(); break;

default: printf("Select valid choice\n");

}

return 0;

}

Output:

1. Bisection Method

2. Regula Falsi Method

3.Newton Raphson Method

Enter your choice: 1

Enter the boundary limits: 5.5 8

Enter precision: 0.001

Enter max. number of iterations: 100

Convergent solution x=6.000488 f(x)=0.000489

Enter your choice: 2

Enter the boundary limits: 5.5 8

Enter precision: 0.001

Enter number of iterations: 100

Convergent Solution x=5.999324 f(x)=-0.000675

Enter your choice: 3

Enter root's neighbour: 5.7

Enter slope minima: 0.001

Enter precision: 0.00001

Enter number of iterations: 100

Convergent Solutions x=6.000004 f(x)=0.000000

1. Use the above program to find 3 roots of the equation xtan(x)=c where c is a user-input constant. Use both the bisection method and Newton-Raphson method.

Code:

#include<stdio.h>

#include<math.h>

float c,ep,sl\_min;

int max\_iter;

float f(float x){

float res=x\*tan(x)-c;

return res;

}

float f\_prime(float x){

float res=tan(x)+x/pow(cos(x),2);

return res;

}

void bisection(float x0,float x1){

float x2,y0,y1,y2;

y0=f(x0);

y1=f(x1);

for (int i=0;i<max\_iter;i++){

x2=(x0+x1)/2;

y2=f(x2);

if (fabs(y2)<=ep){

printf("Convergent solution at %f, value %f\n",x2,y2);

return;

}

if ((y0\*y2)>0){

x0=x2;

y0=y2;

}

else{

x1=x2;

y1=y2;

}

}

printf("Solution does not converge in the given iterations\n");

}

void newtown\_raphson(float x){

for(int i=0;i<max\_iter;i++){

float y=f(x);

float y\_prime=f\_prime(x);

if (fabs(y\_prime)<sl\_min){

printf("Slope too small, diverging");

return;

}

if (fabs(y)<=ep){

printf("Convergent solution at %f, value %f\n",x,y);

return;

}

float x\_new=x-y/y\_prime;

x=x\_new;

}

printf("Solution not coverging in the given iterations\n");

}

int main(){

printf("Enter the value of c");

scanf("%f",&c);

printf("Enter tolerance: ");

scanf("%f",&ep);

printf("Enter Maximum number of iterations");

scanf("%d",&max\_iter);

getchar();

printf("1. Bisection Method\n2. Newtown Raphson Method\n");

int choice;

choice=getchar();

switch(choice){

case '1':

bisection(0,2);

bisection(3,4.4);

bisection(6,8);

break;

case '2':

printf("Enter minimum slope : ");

scanf("%f",&sl\_min);

newtown\_raphson(2);

newtown\_raphson(4);

newtown\_raphson(8);

break;

default: printf("Invalid Choice");break;

}

}

Output:

Enter the value of c10

Enter tolerance: 0.001

Enter Maximum number of iterations100

1. Bisection Method

2. Newtown Raphson Method

1

Convergent solution at 1.428864, value -0.000509

Convergent solution at 4.305835, value 0.000996

Convergent solution at 7.228149, value 0.000890

2

Enter minimum slope : 0

Convergent solution at 7.228110, value 0.000001

Convergent solution at 4.305807, value 0.000156

Convergent solution at 10.200264, value 0.000029

1. There are three real roots of the equation x 3 – 2.5x2 – 2.46x + 3.96 = 0 in the domain [-4, +4]. Write a program to first find out the disjoint subintervals in the given domain those cover the roots. Hence find the roots by Newton-Raphson method.

Code:

#include <stdio.h>

#include <math.h>

double epsilon;

int max\_iter;

double f(double x) {

return pow(x, 3) - 2.5 \* pow(x, 2) - 2.46 \* x + 3.96;

}

double f\_prime(double x) {

return 3 \* pow(x, 2) - 5 \* x - 2.46;

}

double newton\_raphson(double x0) {

double x = x0;

for (int i = 0; i < max\_iter; i++) {

double x\_new = x - f(x) / f\_prime(x);

if (fabs(x\_new - x) < epsilon) {

return x\_new;

}

x = x\_new;

}

return x;

}

void find\_roots(double domain[2], double subintervals[3][2], double roots[3]) {

int N = 100;

double delta = (domain[1] - domain[0]) / N;

int count = 0;

for (int i = 0; i < N; i++) {

double a = domain[0] + i \* delta;

double b = domain[0] + (i + 1) \* delta;

if (f(a) \* f(b) <= 0) {

subintervals[count][0] = a;

subintervals[count][1] = b;

roots[count] = newton\_raphson((a + b) / 2);

count++;

}

}

}

int main() {

double domain[2] = {-4, 4};

double subintervals[3][2];

double roots[3];

printf("enter precision and no. of iterations : ");

scanf("%lf %d",&epsilon,&max\_iter);

find\_roots(domain, subintervals, roots);

printf("Disjoint subintervals that cover the roots:\n");

for (int i = 0; i < 3; i++) {

printf("[%f, %f]\n", subintervals[i][0], subintervals[i][1]);

}

printf("\nRoots:\n");

for (int i = 0; i < 3; i++) {

printf("%f\n", roots[i]);

}

return 0;

}

Output:

enter precision and no. of iterations : 0.001 100

Disjoint subintervals that cover the roots:

[-1.440000, -1.360000]

[0.960000, 1.040000]

[2.800000, 2.880000]

Roots:

-1.376617

1.000000

2.876617