

|  |
| --- |
| C.P.N.M. LAB REPORT |
|  |
| ASSIGNMENT 6  BCSE FIRST YEAR FIRST SEMESTER  Authored by: SOHAM CHOWDHURY |



**CPNM LAB ASSIGNMENT REPORT**

BCSE FIRST YEAR FIRST SEMESTER 2021-2022

NAME-SOHAM CHOWDHURY

DEPARTMENT-COMPUTER SCIENCE AND ENGINEERING

SECTION-A3.

ROLL NO-002110501145.

# ASSIGNMENT 6

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

I initially take the upper limit and lower limit from the user then perform the bisection or newton raphson method or the false position method of numerical methods.for newton raphson only one nearest root is required.

Program:

#include<stdio.h>

#include<math.h>

#define f(x) x\*x\*x-4\*x+1

#define df(x) 3\*x\*x-4

#define e 0.001

int main()

{

    float x0,x1,x2,f1,f0,f2,df0;

    int choice,i=0,n;

    printf("enter the choice=");

    scanf("%d",&choice);

    switch(choice)

    {

        case 1:

        printf("enter x0=");

        scanf("%f",&x0);

        fflush(stdin);

        printf("enter x1=");

        scanf("%f",&x1);

        if(f(x0)\*f(x1)>0)

        {

            printf("invalid inputs");

        }

        else

        {

            do

            {

                f0=f(x0);

                f1=f(x1);

                x2=(x0+x1)/2;

                f2=f(x2);

                if(f0\*f2<0)

                {

                    x1=x2;

                }

                else

                {

                    x0=x2;

                }

                i++;

                } while (fabs(f2)>e);

                printf("root=%f",x2);

                printf("\nnumber of iterrations=%d",i);

        }

        break;

        case 2:

        printf("enter x0=");

        scanf("%f",&x0);

        fflush(stdin);

        printf("enter x1=");

        scanf("%f",&x1);

        if(f(x0)\*f(x1)>0)

        {

            printf("invalid inputs");

        }

        else

        {

            do

            {

                f0=f(x0);

                f1=f(x1);

                x2=((x0\*f1)-(x1\*f0))/(f1-f0);

                f2=f(x2);

                if(f0\*f2<0)

                {

                    x1=x2;

                }

                else

                {

                    x0=x2;

                }

                i++;

            } while (fabs(f2)>e);

            printf("root=%f",x2);

            printf("\nnumber of iterrations=%d",i);

        }

        break;

        case 3:

        printf("enter x0=");

        scanf("%f",&x0);

        printf("enter the maximum number of iterations=");

        scanf("%d",&n);

        do

        {

            f0=f(x0);

            df0=df(x0);

            if(df0==0.0)

            {

                printf("math error");

                break;

            }

            if(i<n)

            {

                x1=x0-f0/df0;

                i++;

                x0=x1;

                f1=f(x1);

            }

        } while (fabs(f1)>e);

        printf("root=%f",x1);

        printf("\nnumber of iterrations=%d",i);

        break;

        default:

        printf("invalid choice");

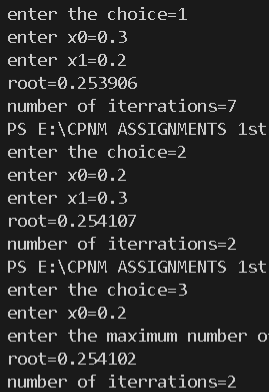
        break;

    }

    return 0;

}

Output:



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.

The equation to be solved is xtan(x)-c=0,then appy the newton raphson and bisection method using these we find three roots for this equation by getting three ranges or nearest roots from the user.

Program: #include<stdio.h>

#include<stdlib.h>

#include <math.h>

#include<limits.h>

#define e 0.001

double f(double x,double c)

{

    return x\*tan(x)-c;

}

double df(double x)

{

    return tan(x)+x\*(1/(cos(x)\*cos(x)));

}

void bisection\_method(double x0,double x1,double c)

{

    double f0,f1,f2,x2,a,b;

    a=x0;

    b=x1;

    int k=0;

    printf("\n  f(x0)\t         f(x1)\t      f(x2)\n");

    do

    {

        f0=f(a,c);

        f1=f(b,c);

        if(f0\*f1>0)

        {

           printf("no root can be found between %lf & %lf",x0,x1);

           k=1;

        }

        x2=(a+b)/2;

        f2=f(x2,c);

        printf("\n%lf\t%lf\t%lf",f0,f1,f2);//printing f1,f0.

        if(f0\*f2>0)

        {

            a=x2;

        }

        else

        {

            b=x2;

        }

    } while (fabs(f2)>e);

    if(k==1)

    printf("no root could be found between %lf and %lf",x0,x1);

    else

    printf("\nthe root using newton ralphson methos is=%lf\n",x2);

}

void newton\_ralphson\_method(double x0,double c,int n)

{

    double g,f0,df0,f1,df1;

    double x1;

    int i=0,k=0;

    g=x0;

    printf("\n  x1\t          f(x1)\t          df(x1)\n");

    do

    {

        f0=f(g,c);

        df0=df(g);

        if(df0==0)

        {

            printf("math error");

            k=1;

            break;

        }

        if(i<n)

        {

            x1=g-(f0/df0);

            i++;

            f1=f(x1,c);

            df1=df(x1);

            printf("%lf\t%lf\t%lf\n",x1,f1,df1);//printing x1,f1,df1

            g=x1;

        }

    } while (fabs(f1)>e);

    if(k==0)

    {

        printf("\nthe root using newton ralphson method =%lf\n",x1);

    }

    else

    printf("\nthe root could not be found\n");

}

int main()

{

    double c;

    int n;

    printf("enter the value of c=");

    scanf("%lf",&c);

    fflush(stdin);

    double x0,x1,f0,f1;

    double root;

    int choice;

    for(int i=0;i<3;i++)

    {

        printf("\nenter the details of the root number %d:\n",i+1);

        printf("enter the choice:=\n1:bisection method\n2:newton ralphson method\n=");

        scanf("%d",&choice);

        if(choice==1)

        {

            printf("enter the value of x0=");

            scanf("%lf",&x0);

            fflush(stdin);

            printf("enter the value of x1=");

            scanf("%lf",&x1);

            bisection\_method(x0,x1,c);

        }

        else if(choice==2)

        {

            printf("enter the value of x0=");

            scanf("%lf",&x0);

            fflush(stdin);

            printf("enter the maximum number of iterations=");

            scanf("%d",&n);

            newton\_ralphson\_method(x0,c,n);

        }

            else

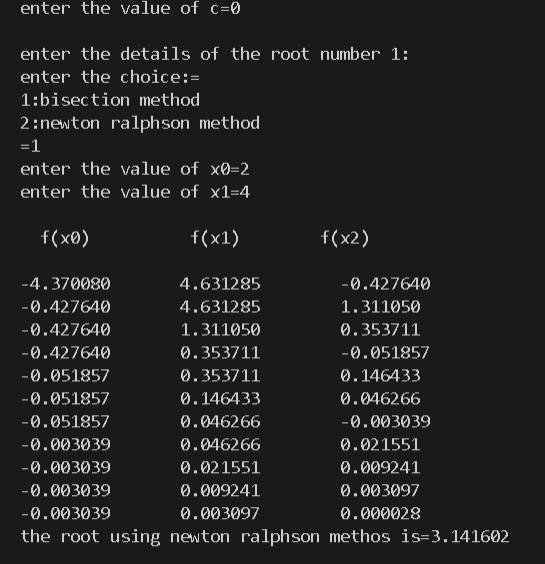
            printf("invalid choice");

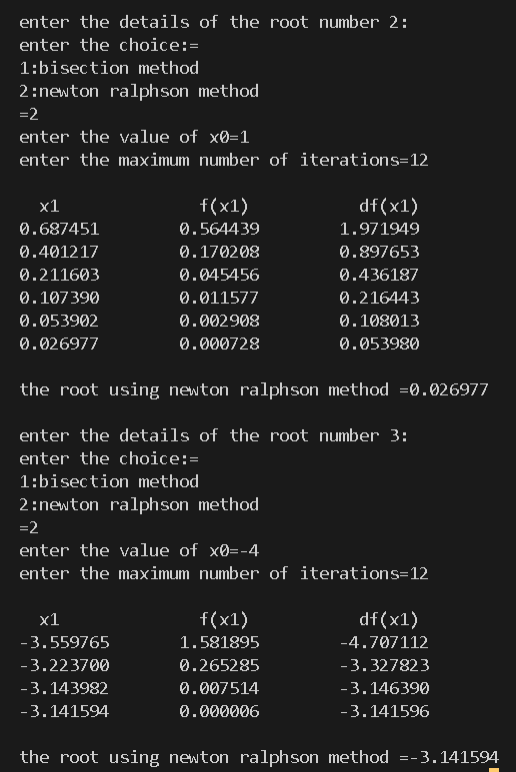
    }

    return 0;

}

Output:





1. There are three real roots of the equation x 3 – 2.5x 2 – 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.

Start x0=-4 and x1=-3 and if a root exists between them then we apply newton raphson method else increase x1/upper limit by 0.01 .

Program:

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define f(x) x\*x\*x-2.5\*x\*x-2.46\*x+3.96

#define df(x) 3\*x\*x-2.5\*2\*x-2.46

#define e 0.001

void newton\_ralphson(float x0,int n,float r1,float r2)

{

    float f0,df0,f1,df1,x1,a;

    int i=0;

    a=x0;

    do

    {

       f0=f(a);

       df0=df(a);

       if(df0==0)

       {

           printf("mathematical error");

           break;

       }

       if(i<n)

       {

           x1=a-f0/df0;

           i++;

           a=x1;

           f1=f(a);

           df1=df(a);

       }

    }while(fabs(f1)>e);

    printf("the root using newton ralphson method within [%f,%f)is=%0.3f\n\n\n",r1,r2,x1);

}

int main()

{

    float x0,x1,root1,root2,root3;

    int n,no\_of\_root=0;

    float f0,f1;

    x0=-4.0;

    x1=-3.0;

    printf("enter the number of iterations(maximum)=");

    scanf("%d",&n);

    while(no\_of\_root<3)

    {

        f0=f(x0);

        f1=f(x1);

        if(f0\*f1<0)

        {

            printf("the root %d exists between [%f,%f)\n",no\_of\_root+1,x0,x1);

            newton\_ralphson(x1,n,x0,x1);

            no\_of\_root++;

            x0=x1;

            if(x1<4)

            x1=x1+1;

            if(no\_of\_root==2)

            x1=4.0;

        }

        else

        {

            x1=x1+0.01;

        }

    }

    return 0;

}

Output:

