

|  |
| --- |
| C.P.N.M LAB REPORT |
|  |
| April 2  BCSE 1ST YEAR 1ST 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 9

1. 1. Find 𝑑𝑦 𝑑𝑥 using forward, backward and central differencing schemes for 𝑦 = 𝑆𝑖𝑛 𝑥 using (i) ∆𝑥 = 0.1 (ii) ∆𝑥 = 0.01 for 0 ≤ 𝑥 ≤ 𝜋 and determine the relative percentage error, defined as | 𝑑𝑦 𝑑𝑥|𝑛𝑢𝑚𝑒𝑟𝑖𝑐𝑎𝑙− 𝑑𝑦 𝑑𝑥|𝑒𝑥𝑎𝑐𝑡 𝑑𝑦 𝑑𝑥|𝑒𝑥𝑎𝑐𝑡 | × 100 in each case with the exact derivative given by 𝑑𝑦 𝑑𝑥 |𝑒𝑥𝑎𝑐𝑡 = 𝑐𝑜𝑠(𝑥). Comment on the result.

PROGRAM:

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define PI 3.14

#define f(x) sin(x)

#define df(x) cos(x)

float forward\_difference(float x0,float dx)

{

    float dfx;

    dfx=(f(x0+dx)-f(x0))/dx;

    return dfx;

}

float backward\_difference(float x0,float dx)

{

    float dfx;

    dfx=(f(x0)-f(x0-dx))/dx;

    return dfx;

}

float centre\_difference(float x0,float dx)

{

    float dfx;

    dfx=(f(x0+dx)-f(x0-dx))/(2\*dx);

    return dfx;

}

void percentage\_error(float x0,float dfx)

{

    float f1x;

    f1x=df(x0);

    float error\_percent;

    error\_percent=(fabs((dfx-f1x)/f1x)\*100);

    printf("the relative percentage error is=%f",error\_percent);

}

int main()

{

    float x0,f1,f0,dx,dfx;

    int choice;

    printf("enter the value of x(in degrees)=");

    scanf("%f",&x0);

    x0=x0\*PI/180;

    printf("1.forward difference\n2.backward diffrence\n3.central difference\nenter the choice=");

    scanf("%d",&choice);

    printf("enter the value of dx(in degrees)=");

    fflush(stdin);

    scanf("%f",&dx);

    dx=dx\*PI/180;

    switch(choice)

    {

        case 1:

        dfx=forward\_difference(x0,dx);

        printf("the value is=%f\n",dfx);

        percentage\_error(x0,dfx);

        break;

        case 2:

        dfx=backward\_difference(x0,dx);

        printf("the value is=%f\n",dfx);

        percentage\_error(x0,dfx);

        break;

        case 3:

        dfx=centre\_difference(x0,dx);

        printf("the value is=%f\n",dfx);

        percentage\_error(x0,dfx);

        break;

        default:

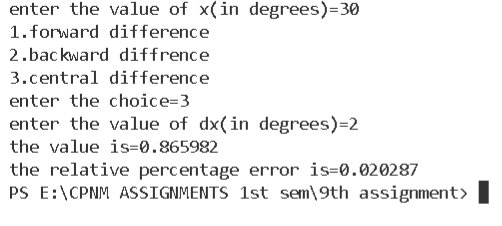
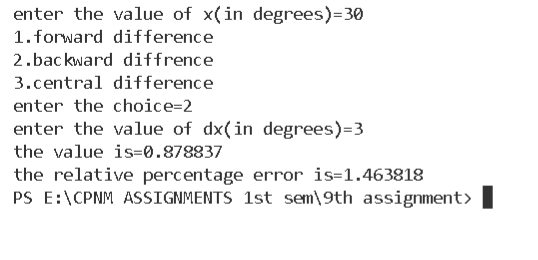
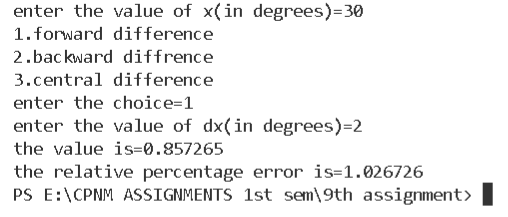
        printf("invalid choice");

    }

    return 0;

}

OUTPUT:



1. Write a function to find the integration of a function f within the limits a and b by Trapezoidal and Simpson's 1/3rd integration method. f, a and b should be provided as arguments of the function integrate. Write the main function to take the name of the integrand function and the limits of integration as command line arguments.

PROGRAM:

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define f(x) x\*x

void trapezodial\_rule(double a,double b,double n)

{

    double h,sum=0,fx,c;

    h=(b-a)/n;

    sum=f(a)+f(b);

    for(int i=1;i<=n-1;i++)

    {

        c=a+i\*h;

        sum=sum+2\*f(c);

    }

    sum=(sum\*((b-a)/n))/2;

    printf("the area under the curve x\*x from %.2f to %.2lf is=%lf",a,b,sum);

}

void simpsons\_1\_3\_method(double a,double b,double n)

{

    double h,sum=0,c;

    h=fabs(b-a)/n;

    for(int i=0;i<=n;i++)

    {

        c=a+i\*h;

        if(i==0)

        sum+=f(c);

        else if(i%2==0)

        sum+=2\*f(c);

        else

        sum+=4\*f(c);

    }

    sum=h\*sum/3;

    printf("the area under the curve x\*x from %.2f to %.2lf is=%lf",a,b,sum);

}

void integrate(double a,double b,double choice,double n)

{

   if(choice==1.00)

   trapezodial\_rule(a,b,n);

   else if(choice==2.00)

   simpsons\_1\_3\_method(a,b,n);

   else

   printf("invalid choice");

}

int main(int argc,const char\* argv[])

{

    double choice,n,a,b;

    printf("1.trapezoidal rule\n2.simpsons 1/3 rd rule\nenter the choice=");

    scanf("%lf",&choice);

    printf("enter the number of divisions=");

    scanf("%lf",&n);

    a=atoi(argv[1]);

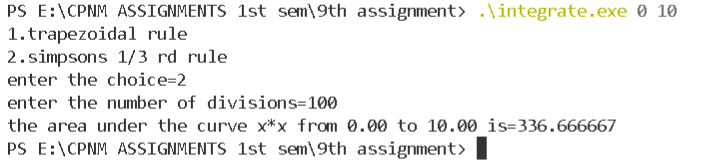
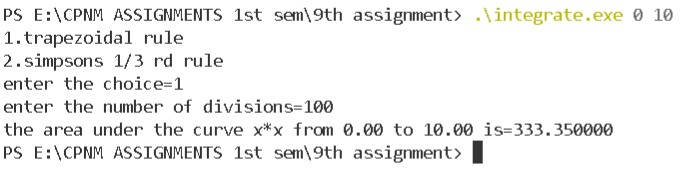
    b=atoi(argv[2]);

    integrate(a,b,choice,n);

    return 0;

}

OUTPUT:



1. Write a program to solve the following differential equations by (i) Euler method, (ii) Runge-Kutta second order method. Compare your solutions. In each method estimate the truncation error and choose an appropriate step size. 𝑑𝑦 𝑑𝑥 = 2𝑥𝑦, 𝑦(0) = 0.5, 𝑠𝑜𝑙𝑢𝑡𝑖𝑜𝑛 𝑓𝑜𝑟 1 ≥ 𝑥 ≥ 0 𝑑𝑦 𝑑𝑥 = 𝑥 2 + 𝑦 2 , 𝑦(0) = 1, 𝑠𝑜𝑙𝑢𝑡𝑖𝑜𝑛 𝑓𝑜𝑟 1 ≥ 𝑥 ≥ 0

PROGRAM:

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define f(x,y) 2\*x\*y

#define g(x,y) x\*x+y\*y

void eulers\_method(int choice,int n,float val)

{

    double h;

    double y[100],f[100],x[100];

    if(choice==1)

    {

        h=fabs(val)/n;

        x[0]=0;

        y[0]=0.5;

        f[0]=f(x[0],y[0]);

        for(int i=1;i<=n;i++)

        {

            y[i]=y[i-1]+h\*f(x[i-1],y[i-1]);

            x[i]=x[0]+(double)i\*h;

            f[i]=f(x[i],y[i]);

        }

    }

    if(choice==2)

    {

        h=fabs(val)/n;

        x[0]=0;

        y[0]=1.0;

        f[0]=g(x[0],y[0]);

        for(int i=1;i<=n;i++)

        {

            y[i]=y[i-1]+h\*f[i-1];

            x[i]=x[0]+(double)i\*h;

            f[i]=g(x[i],y[i]);

        }

    }

    printf("the value of y(%f)=%lf",val,y[n]);

}

void runge\_kutta\_2nd\_order\_method(int choice,int n,float val)

{

    double f[100],x[100],y[100],k1[100],k2[100],h,g[100],x1[100],y1[100];

    if(choice==1)

    {

        x[0]=0;

        y[0]=0.5;

        f[0]=f(x[0],y[0]);

        h=fabs(val)/n;

        k1[0]=h\*f[0];

        k2[0]=h\*f((x[0]+h),(y[0]+k1[0]));

        for(int i=1;i<=n;i++)

        {

            y[i]=y[i-1]+(k1[i-1]+k2[i-1])/2;

            x[i]=x[0]+(double)i\*h;

            f[i]=f(x[i],y[i]);

            k1[i]=h\*f[i];

            k2[i]=h\*f((x[i]+h),(y[i]+k1[i]));

        }

        printf("the value of y(%f)=%lf",val,y[n]);

    }

    if(choice==2)

    {

        x[0]=0.00;

        y[0]=1.00;

        g[0]=g(x[0],y[0]);

        h=fabs(val)/n;

        k1[0]=h\*g[0];

        x1[0]=x[0]+h;

        y1[0]=y[0]+k1[0];

        f[0]=g(x1[0],y1[0]);

        k2[0]=h\*f[0];

        for(int i=1;i<=n;i++)

        {

            y[i]=y[i-1]+(k1[i-1]+k2[i-1])/2;

            x[i]=x[0]+(double)i\*h;

            g[i]=g(x[i],y[i]);

            k1[i]=h\*g[i];

            x1[i]=x[i]+h;

            y1[i]=y[i]+k1[i];

            f[i]=g(x1[i],y1[i]);

            k2[i]=h\*f[i];

        }

        printf("the value of y(%f)=%lf",val,y[n]);

    }

}

int main()

{

    int choice;

    printf("1.dy/dx=2xy\n2.dy/dx=x^2+y^2\nenter the choice=");

    scanf("%d",&choice);

    int choice2,n;

    printf("1.eulers method\n2.runge kutta 2nd order\nenter the choice=");

    scanf("%d",&choice2);

    float x;

    printf("enter the value of x to be found out=");

    scanf("%f",&x);

    fflush(stdin);

    printf("enter the number of divisions=");

    scanf("%d",&n);

    switch(choice)

    {

        case 1:

        switch(choice2)

        {

            case 1:

            eulers\_method(1,n,x);

            break;

            case 2:

            runge\_kutta\_2nd\_order\_method(1,n,x);

            break;

            default:

            printf("invalid choice");

        }

        break;

        case 2:

        switch(choice2)

        {

            case 1:

            eulers\_method(2,n,x);

            break;

            case 2:

            runge\_kutta\_2nd\_order\_method(2,n,x);

            break;

            default:

            printf("invalid choice");

        }

        break;

        default:

        printf("invalid choice");

    }

    return 0;

}

OUTPUT:

