**20K-274 Assignment 2 BCS-4B**

**Numerical Differentiation**

**a)3 point Formula**

#include<iostream.h>

#include<conio.h>

#include<math.h>

float funct(float a);

int main()

{

char choice='y';

float f1,f2,x,h;

clrscr();

cout<<"X ? ";cin>>x;

do{

cout<<"Enter value of h ? ";cin>>h;

cout<<endl<<"The derivative is: "<<endl;

f1=(funct(x+h)-funct(x))/h;

f2=(funct(x+h)-funct(x-h))/(2\*h);

cout<<"2 point derivative : "<<f1;

cout<<endl<<"3 points derivative: "<<f2;

cout<<endl<<"wanna continue (y/n) ? ";cin>>choice;

}while(choice=='y');

getch();

return 0;

}

float funct(float x)

{

return exp(x)\*sin(x);

}

**b) 5 point Formula**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int main()

{

float x[20], y[20][20], xp, h, sum=0.0, term, first\_derivative;

int i,j, n, index, flag = 0, sign=1;

/\* Reading number of data \*/

printf("Enter number of data: ");

scanf("%d", &n);

/\* Reading actual data of x and y \*/

printf("Enter data:\n");

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

{

printf("x[%d] = ", i);

scanf("%f", &x[i]);

printf("y[%d] = ", i);

scanf("%f", &y[i][0]);

}

/\* Reading calculation point \*/

printf("Enter at what value of x you want to calculate derivative: ");

scanf("%f", &xp);

/\* Checking whether given point (xp) is

a valid point in x data or not \*/

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

{

if (fabs(xp - x[i])< 0.0001)

{

/\* index of calculation point \*/

index = i;

flag = 1;

break;

}

}

/\* if flag is still 0 then given point (xp)

is not a valid point (not in list of x data)

for this calculation \*/

if (flag==0)

{

printf("Invalid calculation point. Program exiting...");

exit(0);

}

/\* Generating Forward Difference Table \*/

for(i = 1; i < n; i++)

{

for(j = 0; j < n-i; j++)

{

y[j][i] = y[j+1][i-1] - y[j][i-1];

}

}

/\* Calculate finite difference \*/

h = x[1] - x[0];

/\* Applying formula for calculating sum of

different terms in formula to find derivatives

using forward difference formula \*/

for(i=1; i< n-index; i++)

{

term = pow(y[index][i], i)/i;

sum = sum + sign\*term;

sign = -sign;

}

/\* Dividing by h \*/

first\_derivative = sum/h;

/\* Displaying final result \*/

printf("First derivative at x = %0.2f is %0.2f", xp, first\_derivative);

return 0;

}

**Numerical Integration**

1. **Newtons open cotes formula**

from scipy.integrate import newton\_cotes

def f(x):

return np.sin(x)

a = 0

b = np.pi

exact = 2

for N in [2, 4, 6, 8, 10]:

x = np.linspace(a, b, N + 1)

an, B = newton\_cotes(N, 1)

dx = (b - a) / N

quad = dx \* np.sum(an \* f(x))

error = abs(quad - exact)

print('{:2d} {:10.9f} {:.5e}'.format(N, quad, error))

**b) Composite Trapezoidal, Simpson’s and Midpoint formula**

double f(double x){

  return x\*x;

}

/\*Function definition to perform integration by Trapezoidal Rule \*/

double trapezoidal(double f(double x), double a, double b, int n){

  double x,h,sum=0,integral;

  int i;

  h=fabs(b-a)/n;

  for(i=1;i<n;i++){

    x=a+i\*h;

    sum=sum+f(x);

  }

  integral=(h/2)\*(f(a)+f(b)+2\*sum);

  return integral;

}

/\*Program begins\*/

main(){

  int n,i=2;

  double a,b,h,eps,sum=0,integral,integral\_new;

  /\*Ask the user for necessary input \*/

  printf("\nEnter the initial limit: ");

  scanf("%lf",&a);

  printf("\nEnter the final limit: ");

  scanf("%lf",&b);

  printf("\nEnter the desired accuracy: ");

  scanf("%lf",&eps);

  integral\_new=trapezoidal(f,a,b,i);

  /\* Perform integration by trapezoidal rule for different number of sub-intervals until they converge to the given accuracy:\*/

  do{

    integral=integral\_new;

    i++;

    integral\_new=trapezoidal(f,a,b,i);

  }while(fabs(integral\_new-integral)>=eps);

  /\*Print the answer \*/

  printf("The integral is: %lf\n with %d intervals",integral\_new,i);

}

**a)Divided difference table five values**

def proterm(i, value, x):

pro = 1;

for j in range(i):

pro = pro \* (value - x[j]);

return pro;

# Function for calculating

# divided difference table

def dividedDiffTable(x, y, n):

for i in range(1, n):

for j in range(n - i):

y[j][i] = ((y[j][i - 1] - y[j + 1][i - 1]) /

(x[j] - x[i + j]));

return y;

# Function for applying Newton's

# divided difference formula

def applyFormula(value, x, y, n):

sum = y[0][0];

for i in range(1, n):

sum = sum + (proterm(i, value, x) \* y[0][i]);

return sum;

# Function for displaying divided

# difference table

def printDiffTable(y, n):

for i in range(n):

for j in range(n - i):

print(round(y[i][j], 4), "\t",

end = " ");

print("");

# Driver Code

# number of inputs given

n = 4;

y = [[0 for i in range(10)]

for j in range(10)];

x = [ 5, 6, 9, 11 ];

# y[][] is used for divided difference

# table where y[][0] is used for input

y[0][0] = 12;

y[1][0] = 13;

y[2][0] = 14;

y[3][0] = 16;

# calculating divided difference table

y=dividedDiffTable(x, y, n);

# displaying divided difference table

printDiffTable(y, n);

# value to be interpolated

value = 7;

# printing the value

print("\nValue at", value, "is",

round(applyFormula(value, x, y, n), 2))

**b)Central difference table five values**

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int main()

{

double y[50][50],x[50];

int n,i,j,st,v=0;

double p,e,s,d,sum=0,r,sum1=0,r1,g1,g2,g3,g4;

cout<<"Enter n="<<endl;

cin>>n;

cout<<"Starting Point="<<endl;

cin>>s;

cout<<"End Point="<<endl;

cin>>e;

cout<<"Difference="<<endl;

cin>>d;

p=(e-s)/d;

st=n/2;

cout<<"X"<<"\t"<<"Y"<<endl;

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

{

cin>>x[i]>>y[0][i];

}

cout<<"X"<<"\t"<<"Y"<<endl;

for(i=1;i<n;i++)

{

for(j=0;j<n-i;j++)

{

y[i][j]=y[i-1][j+1]-y[i-1][j];

}

}

cout<<"Forward Difference Table"<<endl;

for(j=0;j<n;j++)

{

printf("%lf",x[j]);

for(i=0;i<n-j;i++)

{

if(y[i][j]>0.0001)

{

printf("\t%lf",y[i][j]);

}

}

printf("\n");

}

g1=p;

g2=(p\*(p-1))/2;

g3=((p+1)\*p\*(p-1))/6;

g4=((p+1)\*p\*(p-1)\*(p-2))/24;

r=y[v][st]+g1\*y[v+1][st]+g2\*y[v+2][st]+g3\*y[v+3][st]+g4\*y[v+4][st];

cout<<"Root is="<<r<<endl;

return 0;

}