***Name of the Experiment:*** Numerical Integration.

***Theory:***

Let the interval [a, b] be divided into n equal subintervals such that a = x0< x1 < x2 <……. Xn = b. Clearly, xn = x0 +nh. Hence the integral becomes

I =

Approximating y by Newton’s forward difference formula, we obtain

I =

Since, xn = x0 +ph, dx = h dp and hence the above integral becomes

I = h

Which gives on simplification,

= nh

From the general formula, we can obtain different intrigation formulae by putting n = 1, 2, 3,…, etc. We derive here a few of this formulae but it should be remarked that the trapezoidal and simpson’s 1/3-rules are found to give sufficient accuracy for use in practical problems.

**Trapezoidal Rule:**

Setting n = 1 in the general formula, all differences higher than the first will become zero and we obtain,

= h( y0 + ∆y0) = h [y0 + (y1 – y0)] = (y0 + y1).

For the next interval [x1 ,x2] we deduce similarly

= (y1 + y2).

And so on. For the last interval [xn-1 ,xn], we have

= (yn-1 + yn).

Combining all these expressions, we obtain the rule

= [y0 + 2(y1 + y2 +…………+ yn-1 ) + yn]

Which is known as trapezoidal rule.

**Simpson’s 1/3-Rule:**

This rule is obtained by putting n = 2 in the general equation by replacing the curve by n/2 arcs of second-degree polynomials of parabola. We have then

= 2h(y0 + ∆y0 + ∆2y0 ) = (y0 + 4 y1 + y2).

Similarly

= (y2 + 4 y3 + y4)

And finally

= (yn-2 + 4 yn-1 + yn).

Summing up we obtain

= [y0 + 4 (y1 + y3 + y5 +…………+ yn-1) + 2(y2 + y4 + y6 +…………+ yn-2) + yn ]

Which is known as simpson’s 1/3 rule.

***Source Code:***

#include<bits/stdc++.h>

using namespace std;

int main()

{

float u,x[10],y[10],h,T,S;

cout<<"Set upperlimit=";

cin>>u;

cout<<"Set lowerlimit=";

cin>>x[0];

while(1)

{

int count=0;

float sum=0,sum2=0,sum3=0;

cout<<"\nSet h(dif)=";

cin>>h;

for(int i=0;; i++)

{

x[i+1]= h+x[i];

count++;

if(x[i+1]>u)

break;

}

for(int i=0; i<count; i++)

{

y[i]=1/(1+x[i]);

}

cout<<" X Y"<<endl;

cout<<".................."<<endl;

for(int i=0; i<count; i++)

{

printf("%.4f\t%.4f\n",x[i],y[i]);

}

for(int i=1; i<count-1; i++)

{

sum=sum+y[i];

}

T=(y[0]+2\*sum+y[count-1])/(2/h);

cout<<"\nBy trapezoidal="<<T<<endl;

for(int i=1; i<count-1; i=i+2)

{

sum2=sum2+y[i];

}

for(int i=1; i<count-2; i=i+2)

{

sum3=sum3+y[1+i];

}

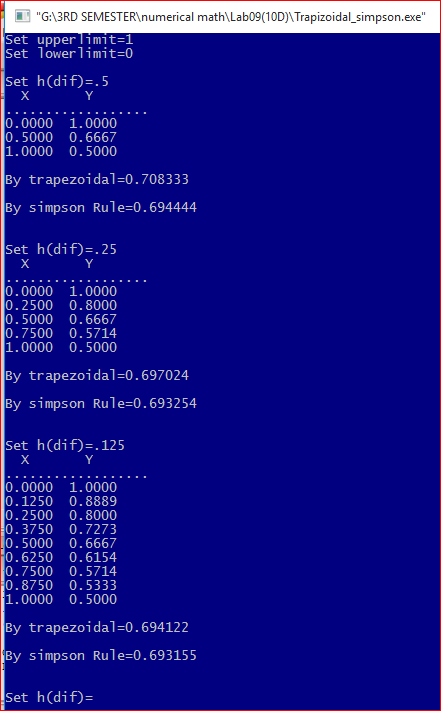
S=(y[0]+4\*sum2+2\*sum3+y[count-1])/(3/h);

//cout<<"\nS=("<<y[0]<<"+"<<4\*sum2<<"+"<<2\*sum3<<"+"<<y[count-1]<<")/"<<(3/h);

cout<<"\nBy simpson Rule="<<S<<"\n\n";

}

***Input/output:***

**

***Discussion:***

Using Newton’s forward difference formula, general formula for numerical integration was derived. Assigning upper limit and lower limit, we see the less the interval we use, the more accurate value will be found. Here the integration of dx=ln2=0.69314.We got more accurate value when h=.125 then h=.5