Assignment - Numerical Intergration Methods

Devesh Khandelwal (72576)

V.1 - Algorithms for Computational Mathematics: Numerical Methods

B. Tech. (Information Technology and Mathematical Innovations)

Cluster Innovation Centre, University of Delhi

Trapezoidal rule

```
#include<iostream>
#include<vector>
#include<cmath>
#include<iomanip>

using namespace std;

double func(vector<double> poly, double x,int degree,double exp)
{
 for(int i=0; i<degree; i++)
 {
   exp += ((poly.at(i))*(pow(x,(degree-i-1))));
 }
 return exp;
}

int main()
{
 char y;
 do
 {

  int degree;
  double exp=0.0,x,n,a,b,temp,val=0,h,segments,answer=0;
  vector<double> poly;
  cout<<"Enter the degree of polynomial\n";
  cin>>degree;
  degree++;
  cout<<"Enter coefficient\n";
    while(poly.size()<degree)  // input the coefficients
    {
     cin>>n;
```

```
      poly.push_back(n);
    }
    for(int i=0; i<degree;i++)   //  display the equation
    {
     if(i!= (degree-1))
       cout<<poly.at(i)<<"x^"<<(degree-i-1)<<" + ";
     else
       cout<<poly.at(i)<<endl;
   }

   cout<<"enter the limits and number of segments \n";
   cin>>a>>b>>segments;
   h= (b-a)/segments;
   cout<<"The value of h is : "<<h<<endl;
   // cin>>a>>b;
   cout<<"Value at a is : "<<func(poly, a, degree, exp)<<endl;
   cout<<"Value at b is : "<<func(poly, b, degree, exp)<<endl;

   answer += func(poly, a, degree, exp) + func(poly, b, degree, exp);

   if (segments > 0)
   {

     for (int i = 0; i < segments-1; ++i)
     {
       val += h;
       answer += 2*func(poly, val, degree, exp);
       //cout<<"Now the answer value is : "<<answer<<endl;
     }
   }
   //cout<<"Before dividing : "<<answer<<endl;
   answer *= (b-a)/(2*segments);
   cout<<"Final answer is  : "<<answer<<endl;
   cout<<"Want to solve more equations\n Press 'y' or 'Y' for yes and any other key
for no\n";
   cin>>y;
}while(y=='y' || y=='Y');
}
```

Output

```
$ ./trapezoidal < input.txt
Enter the degree of polynomial
Enter coefficient
2x^4 + 0x^3 + 3x^2 + 25x^1 + 0.2
enter the limits and number of segments
The value of h is : 0.5
Value at a is : 0.2
Value at b is : 94.2
Final answer is  : 72.775
```

```
Want to solve more equations
Press 'y' or 'Y' for yes and any other key for no
```

Simpson 1/3 rule

```
#include<iostream>
#include<vector>
#include<cmath>
#include<iomanip>

using namespace std;

double func(vector<double> poly, double x,int degree,double exp)
{
 for(int i=0; i<degree; i++)
 {
   exp += ((poly.at(i))*(pow(x,(degree-i-1))));
 }
 return exp;
}

 int main()
 {
   char y;
   do
   {

    int degree;
    double exp=0.0,x,n,a,b,temp;
    double val=0;
    double h,x2, segments,answer=0;
    vector<double> poly;
    cout<<"Enter the degree of polynomial\n";
    cin>>degree;
    degree++;
    cout<<"Enter coefficient\n";
    while(poly.size()<degree)  // input the coefficients
    {
     cin>>n;
     poly.push_back(n);
   }
   for(int i=0; i<degree;i++)   //  display the equation
   {
    if(i!= (degree-1))
      cout<<poly.at(i)<<"x^"<<(degree-i-1)<<" + ";
    else
      cout<<poly.at(i)<<endl;
  }
```

```
  cout<<"enter the limits and number of segments \n";
  cin>>a>>b>>segments;
  h= (b-a)/segments;
  cout<<"The value of h is : "<<h<<endl;
  // cin>>a>>b;
  cout<<"Value at a is : "<<func(poly, a, degree, exp)<<endl;
  cout<<"Value at b is : "<<func(poly, b, degree, exp)<<endl;

  answer += func(poly, a, degree, exp) + func(poly, b, degree, exp);

  if (segments > 0)
  {

    for (int i = 1; i <= segments-1; ++i)
    {
      val += h;
      if (i%2 != 0)
      {
        answer += 4*func(poly, val, degree, exp);
      }

      else
        answer += 2*func(poly, val, degree, exp);
    }
  }
  answer *= (b-a)/(3*segments);
  cout<<"Final answer is  : "<<answer<<endl;
  cout<<"Want to solve more equations\n Press 'y' or 'Y' for yes and any other key
for no\n";
  cin>>y;
}while(y=='y' || y=='Y');
}
```

Output

```
Enter the degree of polynomial
Enter coefficient
2x^4 + 0x^3 + 3x^2 + 25x^1 + 0.2
enter the limits and number of segments
The value of h is : 0.5
Value at a is : 0.2
Value at b is : 94.2
Final answer is  : 71.2333
Want to solve more equations
Press 'y' or 'Y' for yes and any other key for no
```

Simpson 3/8 rule

The 3/8 rule is almost similar only changes in the following function.

```
answer += func(poly, a, degree, exp) + func(poly, b, degree, exp);

  if (segments > 0)
  {

    for (int i = 1; i <= segments-1; ++i)
    {
      val += h;
      if (i%3 == 0)
      {
        answer += 2*func(poly, val, degree, exp);
      }
      else
        answer += 3*func(poly, val, degree, exp);
    }
  }
  answer *= (3*h)/8;
```

Gauss-Quadture 1 point 1D

```
answer = (a-b)*func(poly, (a+b)/2 , degree, exp);
```

Gauss-Quadture 1 point 2D

```
#include<iostream>
#include<iomanip>
#include<math.h>

using namespace std;

float function(float x, float y){
    float value = x+y+5;
    return value;
}

float new_function(float x, float y, float ulx, float llx, float uly, float lly){
    float value=4*((ulx-llx)/2.0)*((uly-lly)/2.0)*(function( (((ulx-llx)/2.0)*x)+
((llx+ulx)/2.0), (((uly-lly)/2.0)*y)+((lly+uly)/2.0) ));
    return value;
}
```

```
int main(){
    float ulx=7, llx=-1, uly=6, lly=-1;
    float ans = new_function(0, 0, ulx, llx, uly, lly);
    cout<<ans;
}
```

Output

```
588
```

Gauss-Quadture 2 point 1D

```
#include<iostream>
#include<iomanip>
#include<math.h>

using namespace std;

float function(float x){
    float value = 1/x;
    return value;
}

float new_function(float x, float ul, float ll){
    float value=((ul-ll)/2.0)*(function((((ul-ll)/2.0)*x)+((ll+ul)/2.0)));
    return value;
}

int main(){
    float ul=2.0, ll=1.0;
    float ans = new_function(-pow(1.0/3.0, 1/2.0), ul, ll) +
new_function(pow(1.0/3.0, 1/2.0), ul, ll);
    cout<<ans;
}
```

Output

```
0.692308
```

Gauss-Quadture 2 point 2D

```
#include<iostream>
```

```
#include<iomanip>
#include<math.h>

using namespace std;

float function(float x, float y){
    float value = pow(x,2)+y+5;
    return value;
}

float new_function(float x, float y, float ulx, float llx, float uly, float lly){
    float value=((ulx-llx)/2.0)*((uly-lly)/2.0)*(function((((ulx-llx)/2.0)*x)+
((llx+ulx)/2.0), (((uly-lly)/2.0)*y)+((lly+uly)/2.0)));
    return value;
}

int main(){
    float ulx=7, llx=-1, uly=6, lly=-1;
    float ans = new_function(pow(1.0/3.0, 1/2.0), pow(1.0/3.0, 1/2.0), ulx, llx,
uly, lly) + new_function(-pow(1.0/3.0, 1/2.0), pow(1.0/3.0, 1/2.0), ulx, llx, uly,
lly) + new_function(pow(1.0/3.0, 1/2.0), -pow(1.0/3.0, 1/2.0), ulx, llx, uly, lly)
+ new_function(-pow(1.0/3.0, 1/2.0), -pow(1.0/3.0, 1/2.0), ulx, llx, uly, lly);
    cout<<ans;
}
```

Output

```
1222.67
```

THE END. *Namaste.*