MA 322: Lab Assignment #6

Due on Sunday, September 13, 2015

Jiten Chandra Kalita

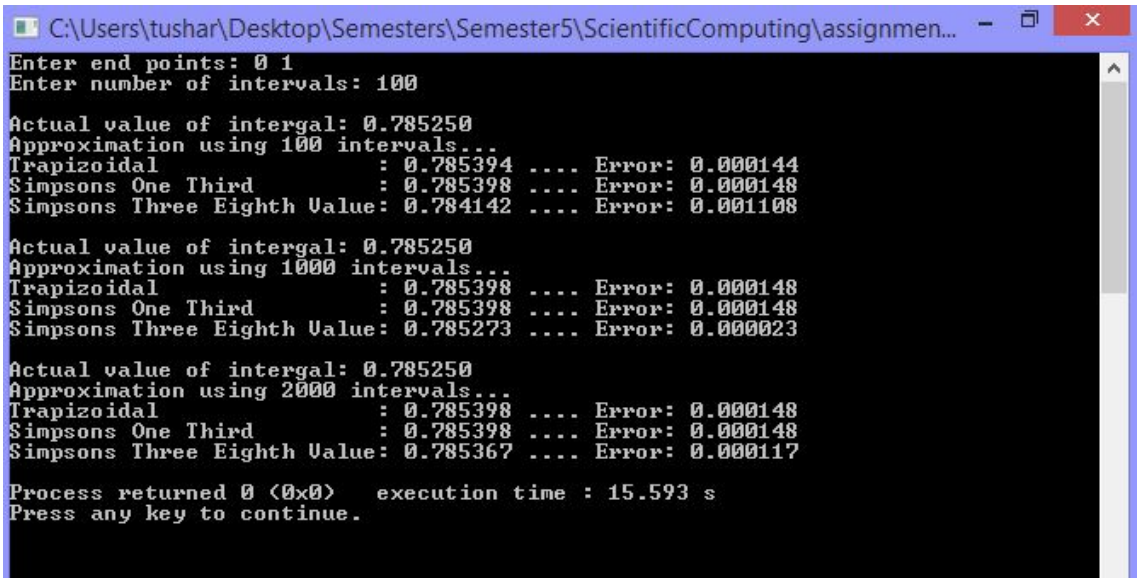
Tushar Sircar - 130123038

PROBLEM 1

```
#include<iostream>
#include<stdio.h>
#include<cmath>
using namespace std;

double getF(double x)
{
    return 1/(1 + (x*x));
}

double trapizoidalValue(double *x,int n,double h)
{
    double ans = 0;
    for(int i=1; i<n; i++)
        ans += getF(x[i]);
    ans = ans*2;
    ans += getF(x[0]) + getF(x[n]);
    return (h*ans)/2;
}

double simpsonsOneThirdValue(double *x,int n,double h)
{
    double ans = 0;
    ans = getF(x[0]) + getF(x[n]);
    for(int i=1; i<=n-1; i++)
    {
        if(i%2 == 0)
            ans += 2*getF(x[i]);
        else ans += 4*getF(x[i]);
    }
    return (ans*h)/3;
}

double simpsonsThreeEighthValue(double *x,int n,double h)
{
    double ans = 0;
    ans = getF(x[0]) + getF(x[n]);
    for(int i=1; i<n; i++)
    {
        if(i%3 == 0)
            ans += 2*getF(x[i]);
        else ans += 3*getF(x[i]);
    }
    return (3*h*ans)/8;
}

void printValuesAndErrors(double *x,int n,double h)
{
    cout<<endl;
    double actual = 3.141/4;
    printf("Actual value of intergal: %f\nApproximation using %d intervals...\n",actual,n);
```

```
    double trapizoidal = trapizoidalValue(x,n,h);
    double sonethird = simpsonsOneThirdValue(x,n,h);
    double sthreeeighth = simpsonsThreeEighthValue(x,n,h);
    printf("Trapizoidal                : %f .... Error: %f\n",trapizoidal,abs(actual-trapizoidal));
    printf("Simpsons One Third         : %f .... Error: %f\n",sonethird,abs(actual-sonethird));
    printf("Simpsons Three Eighth Value: %f .... Error: %f\n",sthreeeighth,abs(actual-sthreeeighth));
}

int main()
{
    double left,right;
    cout<<"Enter end points: ";
    cin>>left>>right;

    int n;
    cout<<"Enter number of intervals: ";
    cin>>n;

    double *x = new double[(20*n)+1];
    double h = (right-left)/n;

    x[0] = left;
    for(int i=1; i<=n; i++)
        x[i] = x[i-1] + h;
    printValuesAndErrors(x,n,h);


    x[0] = left;
    h = h/(double)10;
    for(int i=1; i<=10*n; i++)
        x[i] = x[i-1] + h;
    printValuesAndErrors(x,10*n,h);

    x[0] = left;
    h = h/(double)2;
    for(int i=1; i<=20*n; i++)
        x[i] = x[i-1] + h;
    printValuesAndErrors(x,20*n,h);




    return 0;
}
```

OUTPUT



RESULTS

(a) Number of intervals taken are 100,1000 and 2000.

(b) Error decreases as the number of intervals are increased.

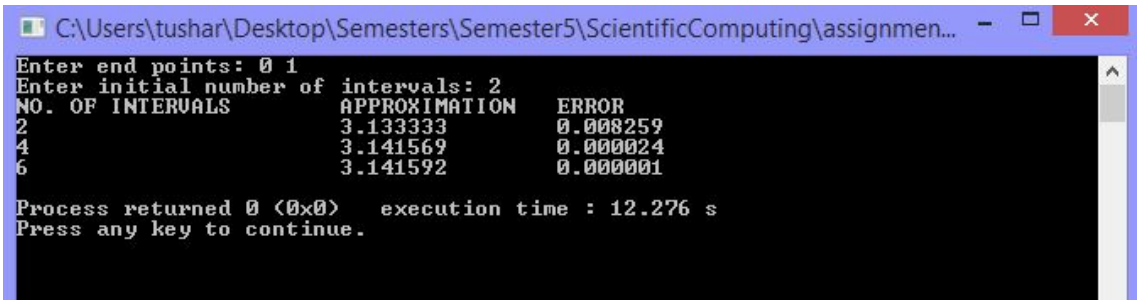
(c) In terms of accuray Simpsons Three-Eighth > Simpsons One-Third > Trapizoidal, as expected based on theory.

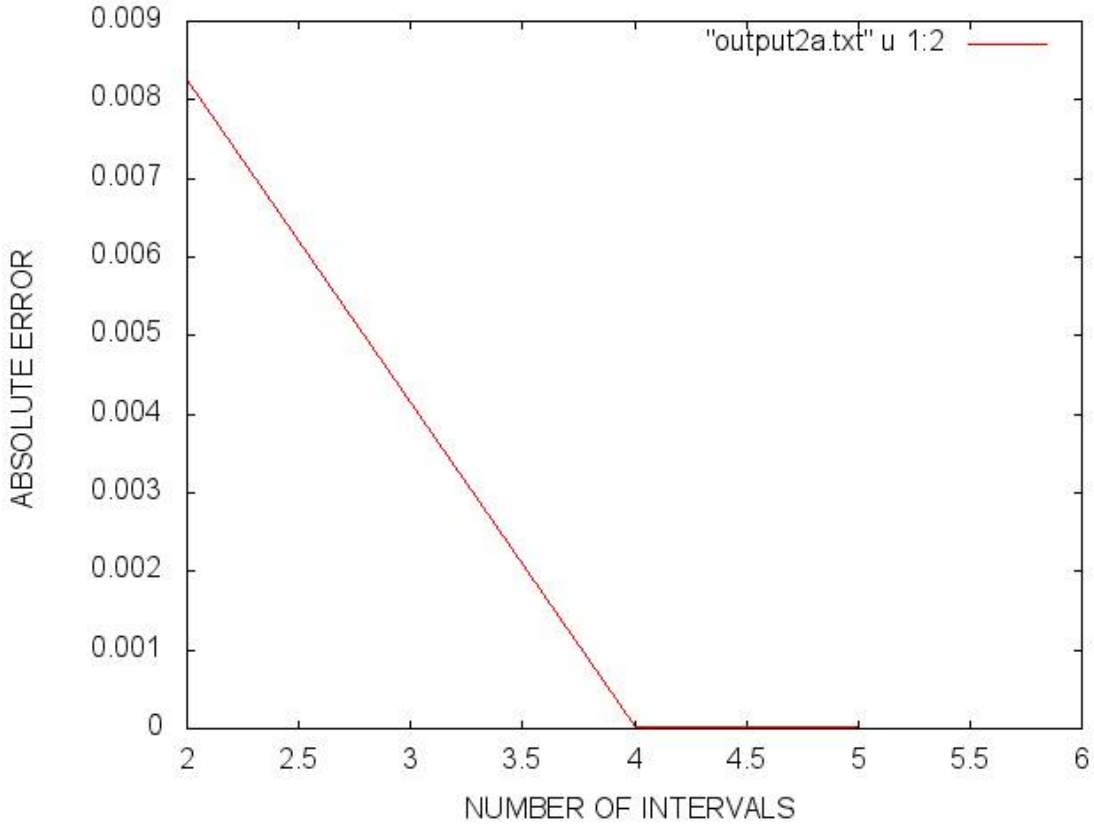
PROBLEM 2(a)

```
#include<iostream>
#include<stdio.h>
#include<cmath>
using namespace std;

double getF(double x)
{
    return (double)4/(1 + (x*x));
}


double simpsonsOneThirdValue(double *x,int n,double h)
{
    double ans = 0;
    ans = getF(x[0]) + getF(x[n]);
    for(int i=1; i<=n-1; i++)
    {
        if(i%2 == 0)
            ans += 2*getF(x[i]);
        else ans += 4*getF(x[i]);
    }
    return (ans*h)/3;
}



double printValuesAndErrors(double *x,int n,double h)
{
    double actual = 4*atan(1);
    double sonethird = simpsonsOneThirdValue(x,n,h);
    printf("%d\t\t\t%f\t%f\n",n,sonethird,abs(sonethird-actual));
    return abs(sonethird-actual);
}


double printIntervalsVsError(double left,double right,int n)
{
    double h = (right-left)/n;
    double *x = new double[n+1];
    x[0] = left;
    for(int i=1; i<=n; i++)
        x[i] = x[i-1] + h;
    return printValuesAndErrors(x,n,h);
}

int main()
{
    double left,right;
    cout<<"Enter end points: ";
    cin>>left>>right;
```

```
    int n;
    cout<<"Enter initial number of intervals: ";
    cin>>n;

    double *x = new double[(50*n)+1];
    double h = (right-left)/n;

    printf("NO. OF INTERVALS\tAPPROXIMATION\tERROR\n");
    int i = n;
    while(1)
    {
       double error = printIntervalsVsError(left,right,i);
       if(error < 0.5*pow(10,-5))
            break;
        i+=2;
    }

    return 0;
}
```

OUTPUT





RESULTS

(a) Error decreases as we increase the number of intervals.

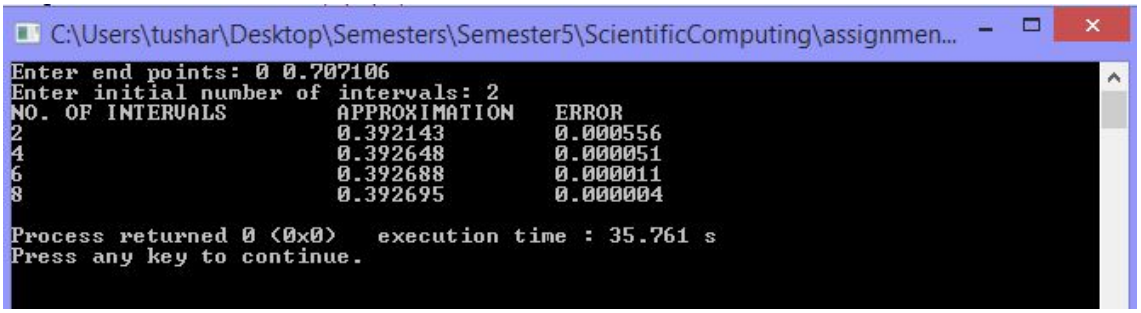
(b) Number of intervals required is 6 for desired accuracy.

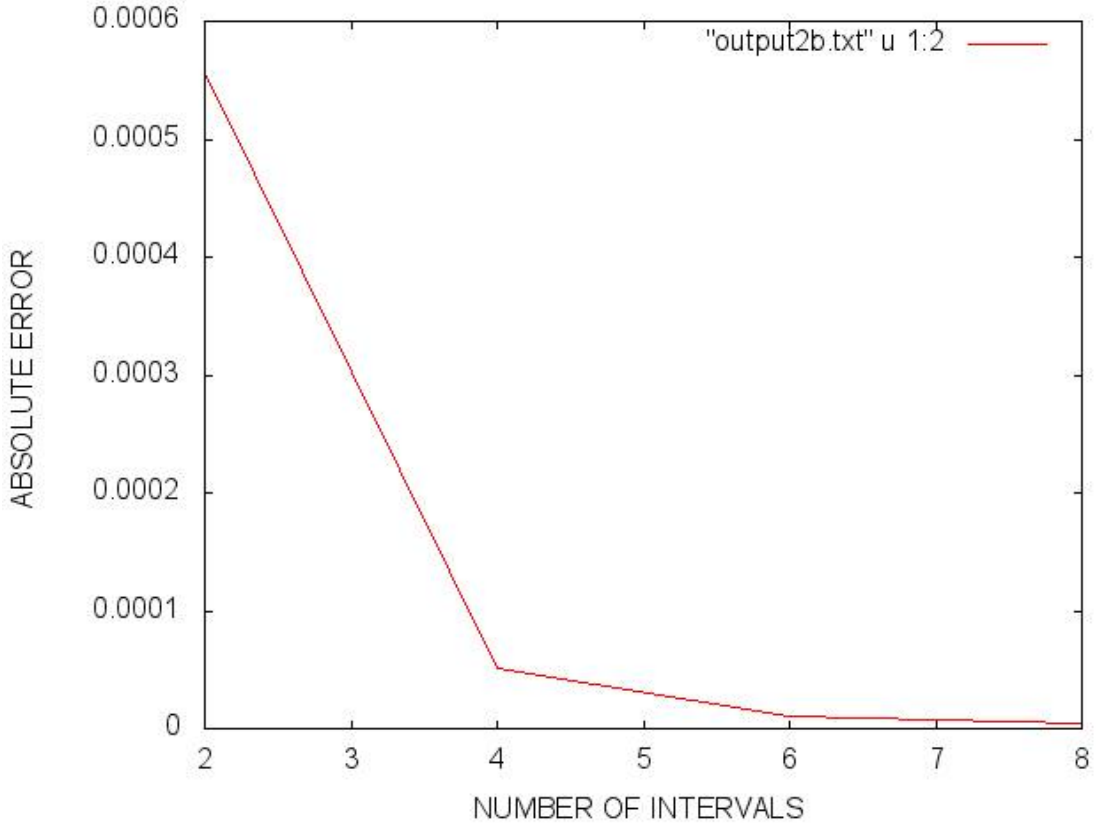
PROBLEM 2(b)

```
#include<iostream>
#include<stdio.h>
#include<cmath>
using namespace std;

double getF(double x)
{
    return sqrt(1 - (x*x)) - x;
}


double simpsonsOneThirdValue(double *x,int n,double h)
{
    double ans = 0;
    ans = getF(x[0]) + getF(x[n]);
    for(int i=1; i<=n-1; i++)
    {
        if(i%2 == 0)
            ans += 2*getF(x[i]);
        else ans += 4*getF(x[i]);
    }
    return (ans*h)/3;
}



double printValuesAndErrors(double *x,int n,double h)
{
    double actual = 0.5*atan(1);
    double sonethird = simpsonsOneThirdValue(x,n,h);
    printf("%d\t\t\t%f\t%f\n",n,sonethird,abs(sonethird-actual));
    return abs(sonethird-actual);
}


double printIntervalsVsError(double left,double right,int n)
{
    double h = (right-left)/n;
    double *x = new double[n+1];
    x[0] = left;
    for(int i=1; i<=n; i++)
        x[i] = x[i-1] + h;
    return printValuesAndErrors(x,n,h);
}

int main()
{
    double left,right;
    cout<<"Enter end points: ";
    cin>>left>>right;
```

```
    int n;
    cout<<"Enter initial number of intervals: ";
    cin>>n;

    double *x = new double[(50*n)+1];
    double h = (right-left)/n;

    printf("NO. OF INTERVALS\tAPPROXIMATION\tERROR\n");
    int i = n;
    while(1)
    {
        double error = printIntervalsVsError(left,right,i);
        if(error < 0.5*pow(10,-5))
            break;
        i+=2;
    }

    return 0;
}
```

OUTPUT





RESULTS

(a) Error decreases as we increase the number of intervals.

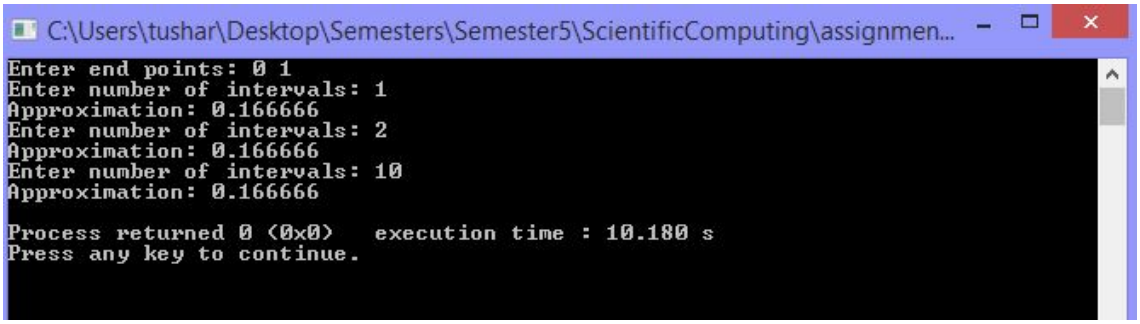
(b) Number of intervals required is 8 for desired accuracy.

PROBLEM 3(b) - Part 1

```
#include<iostream>
#include<stdio.h>
#include<cmath>
using namespace std;

double getF(double x)
{
    return pow(x,(double)5);
}

double convertValue(double x,double a,double b)
{
    return ((b-a)*x + (b+a))/(double)2;
}

double threePointGaussian(double left,double right)
{
    double x1,x2,x3;
    double c1,c2,c3;

    c1 = 0.555555;
    c2 = 0.888888;
    c3 = 0.555555;

    x1 = convertValue(0.7745966692,left,right);
    x2 = convertValue(0,left,right);
    x3 = convertValue(-0.7745966692,left,right);

    return (c1*getF(x1) + c2*getF(x2) + c3*getF(x3));


}

void evaluateByDividing(double left,double right,int n)
{
    double h = (right-left)/n;
    double ans = 0;
    for(int i=0; i<n; i++)
        ans += (h/(double)2)*threePointGaussian(left + (i*h),left + ((i+1)*h));
    printf("Approximation: %f\n",ans);
}

int main()
{
    double left,right;
    cout<<"Enter end points: ";
    cin>>left>>right;

    int n;

    int i =1;
```

```
    while((i++)<4)
    {
        cout<<"Enter number of intervals: ";
        cin>>n;
        evaluateByDividing(left,right,n);
    }


    return 0;
}
```

OUTPUT



PROBLEM 3(b) - Part 2

```
#include<iostream>
#include<stdio.h>
#include<cmath>
using namespace std;

double getF(double x)
{
    return sin(x)/x;
}

double convertValue(double x,double a,double b)
{
    return ((b-a)*x + (b+a))/2;
}

double threePointGaussian(double left,double right)
{
    double x1,x2,x3;
    double c1,c2,c3;

    c1 = 0.555555;
    c2 = 0.888888;
    c3 = 0.555555;

    x1 = convertValue(0.7745966692,left,right);
    x2 = convertValue(0,left,right);
    x3 = convertValue(-0.7745966692,left,right);

    return (c1*getF(x1) + c2*getF(x2) + c3*getF(x3));


}

void evaluateByDividing(double left,double right,int n)
{
    double h = (right-left)/n;
    double ans = 0;
    for(int i=0; i<n; i++)
        ans += (h/(double)2)*threePointGaussian(left + (i*h),left + ((i+1)*h));
    printf("Approximation: %f\n",ans);
}

int main()
{
    double left,right;
    cout<<"Enter end points: ";
    cin>>left>>right;

    int n;

```

```
    int i = 1;
    while((i++)<5)
    {
        cout<<"Enter number of intervals: ";
        cin>>n;
        evaluateByDividing(left,right,n);
    }


    return 0;
}
```