Report of Assignment 10

-----OpenMP

Meng Wei

Results:

My choice is N = 100000000.

Thread Number	Total Time(sec.)
1	10.405875
2	5.992353
4	4.063664
8	4.383599

PS: Codes as follows:

1. OpenMP Trapezoidal Rule:

```
// OpenMPTrapezoidal.cpp : Defines the entry point for the console application.
    //

    #include "stdafx.h"
    #include <stdio.h>
    #include <stdlib.h>
    #include <math.h>
    #include <omp.h>

    //define the integrand f(x)=sin(x)
    double f(double x)
    {
     return (sin(x));
    }
    //compute f''(x) in order to get error bound
    double g(double x)
    {
     return (-sin(x));
    }

    int main()
```

```
{
 double a = 0.0;
 double b = acos(-1);
 int i;
 double wtime;
 int N = 100000000;
 double h = (b - a) / N;
 double s;
 double E;
 s = 0.5*h*(f(a) + f(b));
 wtime = omp_get_wtime();
# pragma omp parallel num_threads(8) shared (a, b, N, h, E) private ( i )

# pragma omp for reduction ( + : s )

 for ( i = 1; i < N; i++)
 {
     s = s + h*f(a + i*h);
 }
    E = (b - a)*h*h / 12;
 wtime = omp_get_wtime() - wtime;// the time for Trapezoidal rule

 printf("%12f\n", wtime);
 printf("integration of sin(x), from 0 to PI, with Trapezoidal Rule is: %0.8f\n",
s);
 printf("error bound in Trapezoidal Rule is: %0.8f\n", E);
 return 0;
}
```

2. Simpson Rule:

```
// OpenMPSimpson.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <omp.h>

//define the integrand f(x)=sin(x)
double f(double x)
{
    return (sin(x));
```

```
}
//compute f''''(x) in order to get error bound
double h(double x)
{
    return (sin(x));
}

int main()
{
    double a = 0.0;
    double b = acos(-1);
    int i;
    int N = 100000000;
    double h = (b - a) / N;
    double s;
    double E;
    double wtime;

    s = f(a) + f(b);
    wtime = omp_get_wtime();
# pragma omp parallel num_threads(8)  shared (a, b, N, h, E ) private ( i )
# pragma omp for reduction ( + : s )


    for (i = 1; i < N; i++)
    {
        if (i % 2 != 0)
            s = s + 4 * f(a + i*h);
        else s = s + 2 * f(a + i*h);
    }
    s = h*s / 3;
    E = (b - a)*h*h*h*h / 180;
    wtime = omp_get_wtime() - wtime;// the time for Simpson rule

    printf("%12f\n", wtime);
    printf("integration of sin(x), from 0 to PI, with Simpson Rule is: %0.8f\n",
s);
    printf("error bound in Simpson Rule is: %0.8f\n", E);
    return 0;
}
```