实验报告3

姓名：吴毅龙 学号：PB19111749

1. 问题

- 分别编写用**复化Simpson**积分公式和**复化梯形**积分公式计算积分的通用程序。
- 用如上程序计算积分$\int_0^8 \frac{x}{1+x^2}$取等距积分节点，记为 $0=x_0<x_1<\ldots<x_N=8$,其中 N分别取值为2^k, $k=0,1,\ldots,10$；分别计算相应的数值积分误差 (用科学计数形式)，并求相应的误差阶（用浮点形式，比如1.8789）。
- 比较并分析两种方法的**优劣**。

2. 计算过程及计算结果

2.1 算法与程序实现

复化Simpson积分公式和复化梯形积分公式分别是：

对于积分区间$[a,b]$，取n个等距间隔，则每个间隔的长度为$h=\frac{b-a}{n}$，令$x_i=a+ih$，$i=0,1,\ldots,n$，$n=2m$

复化梯形：$I(f)=\sum_{i=0}^{n-1}\{\frac{h}{2}(f(x_i)+f(x_{i+1}))-\frac{h^3}{12}f'(\xi_i)\}$

复化Simspon：$I(f)=\sum_{i=0}^{m-1}\{\frac{h}{3}(f(x_{2i})+f(x_{2i+1})+f(x_{2i+2}))-\frac{(2h)^5}{2880}f^{(4)}(\xi_i)\}$

误差：$E(f)=|\int_0^8 \frac{x}{1+x^2}-I(f)|$

使用C语言对上述算法进行程序实现

```c
#include<stdio.h>
#include<math.h>

double low = 0;
double high = 8;

double function(double x)
{
    return x / (1 + pow(x, 2));
}

double CompoundTrapezoid(double n)
{
    double result = 0;
    double length = high - low;
    double h = length / n;
    int k;
    for (k = 0; k < n; k++)
    {
        result += function(low + k * h) + function(low + (k + 1) * h);
    }
    result = result * h / 2;
    return result;
}
```

```
double CompoundSimpson(double n)
{
    double result = 0;
    double length = high - low;
    double h = length / n;
    double m = n / 2;

    for (int k = 0; k < m; k++)
    {
        result += function(low + (2 * k) * h) + 4 * function(low + (2 * k + 1) *
h) + function(low + (2 * k + 2) * h);
    }
    result = result * h / 3;
    return result;
}

int main()
{
    double trueval = 0.5 * log(65);
    double curr;
    double result;
    double error;

    printf("CompoundTrapezoid\n");
    curr = CompoundTrapezoid(1);
    error = fabs(trueval - curr);
    printf("k=0 result=%.12E error=%.12E\n", curr, error);
    for (int k = 1; k <= 10; k++)
    {
        result = curr;
        curr = CompoundTrapezoid(pow(2,k));
        double error1 = fabs(trueval - result);
        double error2 = fabs(trueval - curr);
        double d = -1 * (log(error2) - log(error1)) / log(2);

        printf("k=%d result=%.12E error=%.12E d=%.4E\n", k, curr, error2, d);
    }

    printf("CompoundSimpson\n");
    curr = CompoundSimpson(1);
    printf("k=0 result=%.12E error=%.12E\n", curr, error);
    for (int k = 1; k <= 10; k++)
    {
        result = curr;
        curr = CompoundSimpson(pow(2, k));
        double error1 = fabs(trueval - result);
        double error2 = fabs(trueval - curr);
        double d = -1 * (log(error2) - log(error1)) / log(2);

        printf("k=%d result=%.12E error=%.12E d=%.4E\n", k, curr, error2, d);
    }
    return 0;
}
```

2.2 计算结果

2.2.1 复化梯形积分

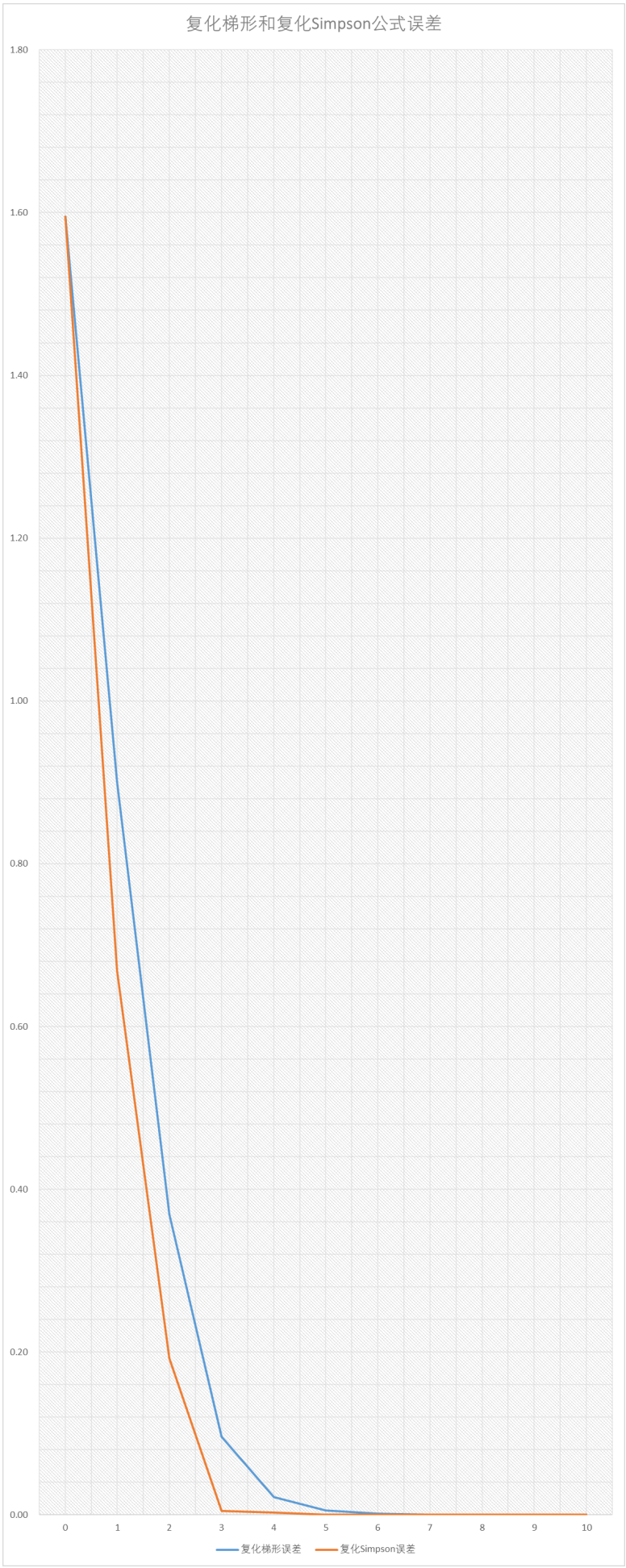
k	公式计算结果	误差	误差阶
0	4.923076923077E-01	1.594885942640E+00	
1	1.187330316742E+00	8.998633182057E-01	8.2568E-01
2	1.717989482695E+00	3.692041522525E-01	1.2853E+00
3	1.991302433655E+00	9.589120129244E-02	1.9449E+00
4	2.065438757416E+00	2.175487753153E-02	2.1401E+00
5	2.081874052193E+00	5.319582755276E-03	2.0320E+00
6	2.085870086459E+00	1.323548489293E-03	2.0069E+00
7	2.086863132827E+00	3.305021210829E-04	2.0017E+00
8	2.087111033310E+00	8.260163769558E-05	2.0004E+00
9	2.087172986029E+00	2.064891875087E-05	2.0001E+00
10	2.087188472811E+00	5.162136559544E-06	2.0000E+00

2.2.2 复化Simpson积分

k	公式计算结果	误差	误差阶
1	1.419004524887E+00	6.681891100609E-01	
2	1.894875871346E+00	1.923177636014E-01	1.7968E+00
3	2.082406750642E+00	4.786884305774E-03	5.3283E+00
4	2.090150865337E+00	2.957230388777E-03	6.9484E-01
5	2.087352483785E+00	1.588488368074E-04	4.2185E+00
6	2.087202097881E+00	8.462932700848E-06	4.2304E+00
7	2.087194148283E+00	5.133349878506E-07	4.0432E+00
8	2.087193666805E+00	3.185676655804E-08	4.0102E+00
9	2.087193636935E+00	1.987565667605E-09	4.0025E+00
10	2.087193635072E+00	1.241682312525E-10	4.0006E+00

3. 计算结果分析

如图所示为复化梯形与复化Simpson公式积分在不同k值下的误差



由此可见，复化Simpson积分公式向精确值的收敛速度比复化梯形积分公式更快。且随着k值的增加，复化梯形积分公式的误差阶逐渐收敛到2，复化Simpson积分公式的误差阶逐渐收敛到4.

综合来看，复化Simpson公式的算法表现要优于复化梯形公式。

4. 实验总结

通过这个实验，我再一次体会到算法从理论落地到实践的过程，这其中需要解决许多在理论推演过程中无法想象到的问题。例如在编程的过程中对于下标的处理是一个比较容易出错的地方，在认真检查程序之后，终于发现了这个细微的错误。同时这次实验也让我再次体会到了一个优秀的算法的重要性和先进性，优秀的算法在相同的硬件和软件条件下可以极大的提高生产效率，这不仅仅是简单的时间缩短，在实际的工作生产场合中还有会有更加巨大的实践意义。