龙贝格公式的实现

姓名：岳宇轩　　序号：14　　专业：19 慧与

指导老师：高云　　日期：2021.10.30

一．实验目的：

通过编程实验，熟练掌握龙贝格公式及复合梯形公式，并比较二者的精度。

二．实验步骤

1．验证龙贝格公式的积分效果

2．验证龙贝格公式对复合梯形公式精度的提高

三．实验原理

1.复合梯形公式：

复合梯形公式的基础是梯形公式

$$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a)+f(b)] \tag{1}$$

为了提高求积精度，实际计算时可以将步长 h 逐次分半

$$\frac{h}{4}[f(x_k)+2f(x_{k+\frac{1}{2}})+f(x_{k+1})] \tag{2}$$

注意，这里 $h=\frac{b-a}{2^{n-1}}$ 代表二分前的步长。将每个子区间上的积分值相加得

$$T_n=\frac{h}{4}\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}}) \tag{3}$$

从而导出下列递推公式

$$T_{n+1}=\frac{1}{2}T_n+\frac{h}{2}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}}) \tag{4}$$

其中

$$T_1=\frac{h}{2}[f(a)+f(b)] \tag{5}$$

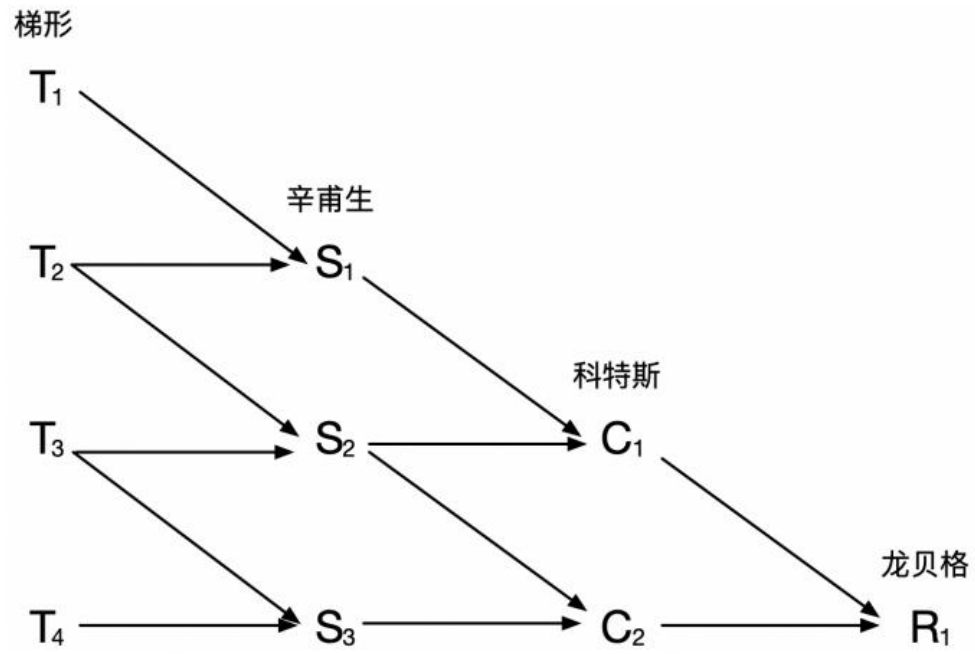
利用 T_1 和公式 (4)，我们可以递推地计算 $T_2, T_3, \cdots$

2. 龙贝格公式：

梯形法的算法虽然简单，但是精度低，收敛的速度缓慢，所以我们可以通过将精度较低的梯形法加工成高精度的方法，利用梯形公式的余项，并加以补偿我们可以得到

$$
\begin{aligned}
S_n &= T_{n+1} + \frac{1}{3}(T_{n+1} - T_n) = \frac{4}{3}T_{n+1} - \frac{1}{3}T_n \\
C_n &= S_{n+1} + \frac{1}{15}(S_{n+1} - S_n) = \frac{16}{15}S_{n+1} - \frac{1}{15}S_n \\
R_n &= C_{n+1} + \frac{1}{63}(C_{n+1} - C_n) = \frac{64}{63}C_{n+1} - \frac{1}{63}C_n
\end{aligned}
\tag{6}
$$

其中，Sn 为辛甫生公式，Cn 为柯特斯公式，Rn 为龙贝格公式。



我们在步长二分的过程中运用公式 (6) 加工，就能将粗糙的积分值逐步加工成精度较高的龙贝格值，或者说，将收敛缓慢的梯形值序列加工成收敛迅速的龙贝格值序列 。这种加速方法称龙贝格算法。

四．实验过程

1.定义一个积分函数

这里使用的是 $\int_0^1 \sqrt{2x - x^2}\,dx = \frac{\pi}{4} \approx 0.785398$

```
/// 被积函数 f(x)
double func(double x) {
    return sqrt(2 * x - x * x);
}
```

3. 求二分点函数值之和

```
/// 二分点的函数值之和
double sum(double a, double b, double h) {
    double result = 0.0;

    double x = a;
    while (x < b) {
        result += func(x + h / 2);
        x += h;
    }

    return result;
}
```

4. 加工计算

使用龙贝格算法计算积分值，需要首先计算复合梯形公式的 T1,T2,T3,T4，进而计算出辛甫生公式的 S1,S2,S3 和科特斯公式的 Ct,C2，才能计算 R。

```
/// 龙贝格算法
void Romberg() {
    double a = 0.0;
    double b = 1.0;

    double h = b - a;
    T[1] = h / 2.0 * (func(a) + func(b));

    T[2] = T[1] / 2.0 + h / 2.0 * sum(a, b, h);
    h = h / 2.0;
    T[3] = T[2] / 2.0 + h / 2.0 * sum(a, b, h);
    h = h / 2.0;
    T[4] = T[3] / 2.0 + h / 2.0 * sum(a, b, h);
    h = h / 2.0;

    S[1] = T[2] + (T[2] - T[1]) / 3;
    S[2] = T[3] + (T[3] - T[2]) / 3;
    S[3] = T[4] + (T[4] - T[3]) / 3;

    C[1] = S[2] + (S[2] - S[1]) / 15;
    C[2] = S[3] + (S[3] - S[2]) / 15;
```

```
    R[1] = C[2] + (C[2] - C[1]) / 63;

    while (fabs(R[k - 3] - REAL_VALUE) >=
0.00000005) {
        k++;
        T[k] = T[k - 1] / 2.0 + h / 2.0 * sum(a, b, h);
        h = h / 2;
        S[k - 1] = T[k] + (T[k] - T[k - 1]) / 3;
        C[k - 2] = S[k - 1] + (S[k - 1] - S[k - 2]) / 15;
        R[k - 3] = C[k - 2] + (C[k - 2] - C[k - 3]) / 63;
    }
}
```

在这里使用 T 数组记录复化梯形公式的计算结果，使用 S 数组存储辛甫生公式计算的结果，使用 C 数组记录科特斯公式的计算结果，使用 R 数组记录龙贝格公式的计算结果。由于需要进行计算到 T4 才可有 R1，因此在这里设置 k=4。这里要求 6 位有效数字，即误差不大于 5×10^-6。

```
double T[20];
double S[20];
double C[20];
double R[20];
int k = 4;
```

5. 验证龙贝格公式积分效果

```
/// 验证龙贝格公式的积分效果
printf("i\t R[i]\t\t 误差\n");
for (int i = 1; i < k - 2; i++) {
    printf("%d\t %f\t %f\n", i, R[i], fabs(R[i] -
REAL_VALUE));
}
```

这里定义真值如下

```
#define REAL_VALUE 0.785398
```

6. 验证龙贝格公式对复合梯形公式精度的提高

取出 T 数组和 R 数组中相应的数字进行比较

```
///验证龙贝格公式对复合梯形公式精度的提高
printf("k\t T[k]\t\t S[k]\t\t C[k]\t\t R[k]\n");
for (int i = 1; i < k + 1; i++) {
    if (i < 2) {
        printf("%d\t %f\n", i, T[i]);
    } else if (i < 3) {
        printf("%d\t %f\t %f\n", i, T[i], S[i - 1]);
    } else if (i < 4) {
        printf("%d\t %f\t %f\t %f\t\n", i, T[i], S[i - 1],
C[i - 2]);
    } else {
        printf("%d\t %f\t %f\t %f\t %f\n", i, T[i], S[i -
```

```
1], C[i - 2], R[i - 3]);
    }
}
```

五．实验结果

a. 验证龙贝格公式的积分效果；

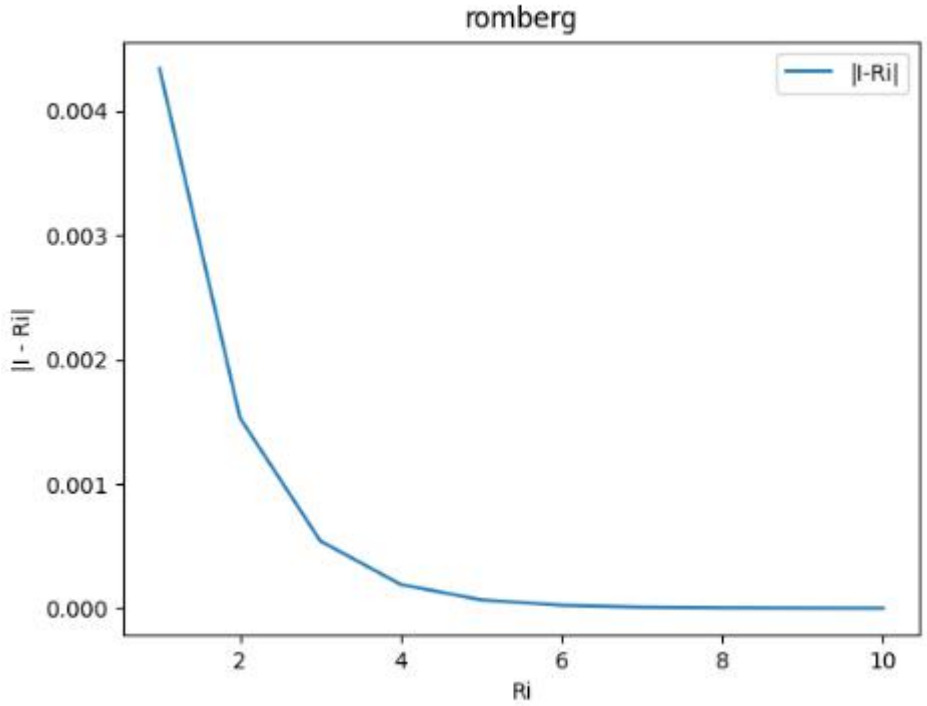
```
i       R[i]            误差
1       0.781055        0.004343
2       0.783866        0.001532
3       0.784857        0.000541
4       0.785207        0.000191
5       0.785331        0.000067
6       0.785374        0.000024
7       0.785390        0.000008
8       0.785395        0.000003
9       0.785397        0.000001
10      0.785398        0.000000
```

b. 验证龙贝格公式对复合梯形公式精度的提高。

```
Microsoft Visual Studio 调试控制台
k       T[k]            S[k]            C[k]            R[k]
1       0.500000
2       0.683013        0.744017
3       0.748927        0.770899        0.772691
4       0.772455        0.780297        0.780924        0.781055
5       0.780813        0.783599        0.783820        0.783866
6       0.783776        0.784763        0.784841        0.784857
7       0.784824        0.785174        0.785201        0.785207
8       0.785195        0.785319        0.785329        0.785331
9       0.785326        0.785370        0.785374        0.785374
10      0.785373        0.785388        0.785389        0.785390
11      0.785389        0.785395        0.785395        0.785395
12      0.785395        0.785397        0.785397        0.785397
13      0.785397        0.785398        0.785398        0.785398
```

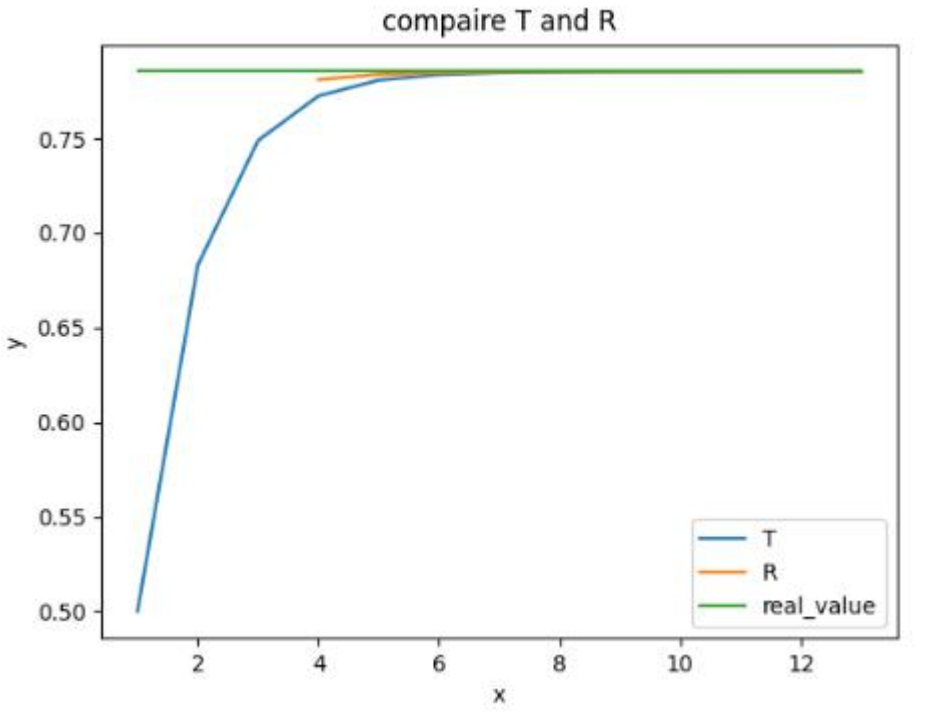
六．实验结果分析

a. 验证龙贝格公式的积分效果；

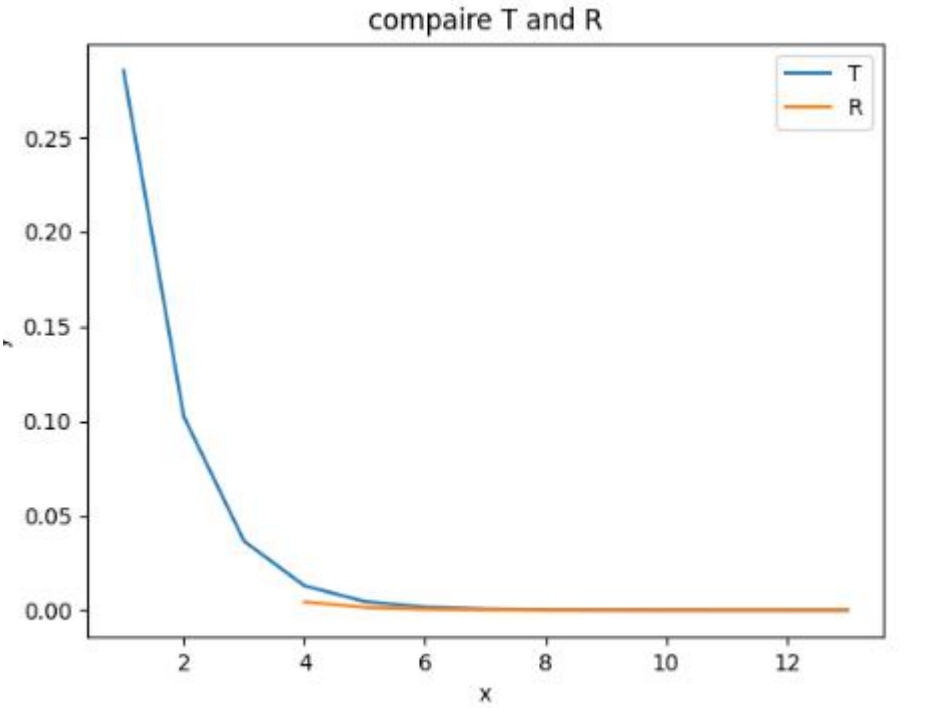


可以看出龙贝格公式的精度还是很高的，在使用一次龙贝格公式是精度就已经达到 2 位有效数字了。并且精度上升得很快，再递推调用数次龙贝格公式后得出的数值已经很接近真实值了。

b. 验证龙贝格公式对复合梯形公式精度的提高。



可以看到龙贝格公式比复化梯形公式更快接近真值



我们可以发现龙贝格公式能够很好的加速数值积分收敛到积分真实值，并且龙贝格公式的收敛速度大于复合梯形公式的收敛速度。所以使用龙贝格公式能够很好的计算数值积分，获得较高精度的数值。

七．实验心得

1.龙贝格算法比较简单，只需要计算需要首先计算复合梯形公式的 Ti,T2,T,T4、辛甫生公式的 S1,S2,S3、科特斯公式的 Ct,C2 和龙贝格公式的 R，就能够递推地使用龙贝格算法计算数值积分，直到精度符合要求。

2.本次实验通过使用 C++编程实现了龙贝格算法，发现龙贝格公式能够在有限的计算次数快速收敛到积分的真实值，计算速度快而且精度高。

3.龙贝格算法通过递推地补偿复合梯形公式的余项误差，使得误差较小。所以说在有限的等距的函数值时，通过使用龙贝格公式能够获得精度最高的积分值。