

数值积分的实现

第七小组：刘阳 任浩辰

科目：数值分析

日期：2020 年 4 月 18 日

1 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。

2 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

3 实验内容

3.1 复合梯形公式

复合梯形公式的基础是梯形公式

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

为了提高求积精度，实际计算时可以将步长 h 逐次分半

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

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

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

从而导出下列递推公式

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

其中

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

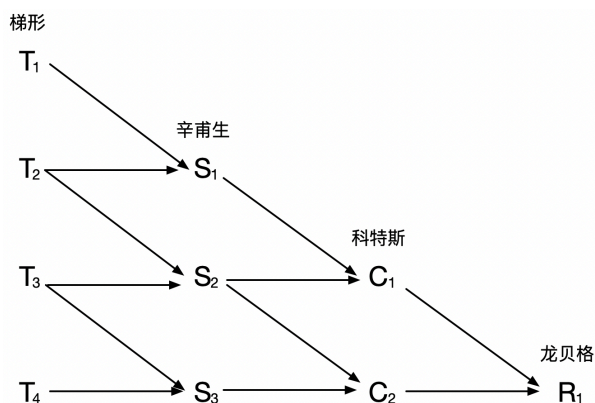
利用 T_1 和公式 (4)，我们可以递推地计算 T_2, T_3, \dots

3.2 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

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

其中， S_n 为辛甫生公式， C_n 为柯特斯公式， R_n 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 T_n 逐步加工成精度较高的龙贝格值 R_n ，或者说，将收敛缓慢的梯形值序列 T_n 加工成收敛迅速的龙贝格值序列 R_n 。这种加速方法称龙贝格算法。

4 实验过程及关键语句描写

4.1 验证龙贝格公式的积分效果

这里所求的定积分是

$$I = \int_0^1 \sqrt{2x - x^2} dx = \frac{\pi}{4} \approx 0.785398$$

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

求二分点的函数值之和

$$\sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}})$$

```

/// 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;
}

```

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 T_1, T_2, T_3, T_4 ，进而计算出辛甫生公式的 S_1, S_2, S_3 和科特斯公式的 C_1, C_2 ，才能计算 R_1 。

```

#define REAL_VALUE 0.785398 // 真实值

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

/// 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) >= 0.0000005) {
    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;
}
}

```

这里要求 6 位有效数字，即误差不大于 5×10^{-6} 。

4.2 验证龙贝格公式对复合梯形公式精度的提高

因为计算龙贝格公式需要复合梯形公式的数值，所以对比龙贝格公式和复合梯形公式，只要将储存在数组的数值取出即可。

5 实验结果

5.1 验证龙贝格公式的积分效果

通过调用龙贝格算法计算积分值并计算与真实值的误差。

```

/Users/leo/items/CLion/NumericalAnalysis/cmake-build-debug
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

Process finished with exit code 0

```

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。

5.2 验证龙贝格公式对复合梯形公式精度的提高

将储存在数组中的复合梯形公式、辛甫生公式、科特斯公式以及龙贝格公式的计算数值打印成下三角数表，比较它们的精度。

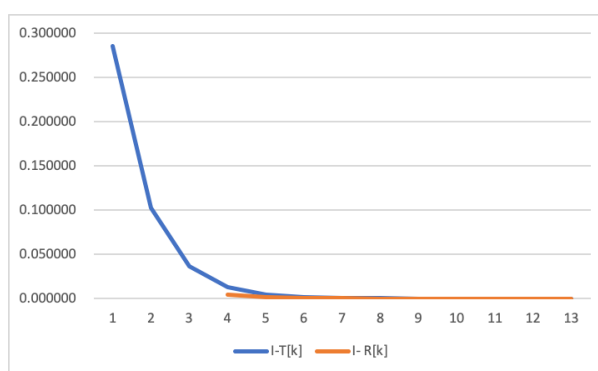
```

/Users/leo/items/CLion/NumericalAnalysis/cmake-build-debug
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

Process finished with exit code 0

```

进一步，对比复合梯形公式和龙贝格公式对于真实值的误差。



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。

6 实验体会

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 T_1, T_2, T_3, T_4 、辛甫生公式的 S_1, S_2, S_3 、科特斯公式的 C_1, C_2 和龙贝格公式的 R_1 ，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。