数值分析第二次作业

杨弦昊

中国机械科学研究总院

【题目一】 已知积分

$$\int_0^1 \frac{4}{1+x^2}\mathrm{d}x = \pi$$

成立，所以我们可以通过对上面给定被积函数的数值积分来计算 π 的近似值.

(1) 分别使用复合中点公式、复合梯形公式和复合 Simpson 公式计算 π 的近似值。选择不同的 h，对每种求积公式，试将误差刻画成 h 的函数，并比较各方法的精度。是否存在某个 h 值，当小于这个值之后再继续减小 h 的值，计算不再有所改进？为什么？

(2) 实现 Romberg 求积方法，并重复上面的计算。

(3) 使用自适应求积方法重复上面的计算。(建议自学自适应求积方法)

解：(1) 对于被积函数，编写函数名为 f_4runge 的函数，输入为自变量 i，输出为函数值：

```c
// 定义被积函数 4*runge
double f_4runge(double i) {
    return 4 / (1 + i * i);
}
```

根据复合中点求积公式：

$$\int_a^b f(x)\mathrm{d}x \approx h\sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) \triangleq M(h) \tag{1}$$

编写函数名为 remix_center_point 的 C 语言函数如下用于计算积分：

```c
// 构造复合中点求积公式函数
double remix_center_point(double a, double b, int n, FunctionType func1) {
    double h;
    double x_frac;
    double f_value;
    double result_sum = 0;
    h = (b - a) / n;
    for (int i = 0; i < n; i++) {
        x_frac = a + ((i + 0.5) * h);
        f_value = func1(x_frac);
        result_sum += f_value;
    }
    return h * result_sum;
}
```

其中函数的输入为积分起点 a，积分终点 b，积分精度 n，被积函数 func1；

调用方法如下：

```c
#define N_center 1e6
double a = 0;
a = remix_center_point(0, 1, N_center, f_4runge);
```

同理根据复合梯形求积公式：

$$\int_a^b f(x)\mathrm{d}x \approx \frac{h}{2}\sum_{i=0}^{n-1}[f(x_i)+f(x_{i+1})] \triangleq T(h) \tag{2}$$

编写函数名为 remix_trapezoid 的求积函数，函数输入为积分起点 a，积分终点 b，积分精度 n，被积函数 func1；

```
//构造复合梯形求积公式函数
double remix_trapezoid(double a, double b, int n, FunctionType func1) {
    double h;
    double x_i;
    double x_i_plus;
    double f_value;
    double f_value_plus;
    double result_sum = 0;
    h = (b - a) / n;
    for (int i = 0; i < n; i++) {
        x_i = a + (i * h);
        x_i_plus = a + ((i+1) * h);
        f_value = func1(x_i);
        f_value_plus = func1(x_i_plus);
        result_sum += (f_value + f_value_plus);
    }
    return (h / 2.0000) * result_sum;
}
```

调用方法如下所示：

```
#define N_trapezoid 1e6
double b = 0;
b = remix_trapezoid(0, 1, N_trapezoid, f_4runge);
```

同理根据复合 Simposon 求积公式：

$$\int_a^b f(x)\mathrm{d}x \approx \frac{h}{6}\sum_{i=0}^{n-1}\left[f(x_i)+4f(x_{i+\frac{1}{2}})+f(x_{i+1})\right] \triangleq S(h) \tag{3}$$

编写求复合 Simpson 求积公式函数，函数名为 remix_simposon，函数输入为积分起点 a，积分终点 b，积分精度 n，被积函数 func1；

```
//构造Simpson求积公式函数
double remix_simposon(double a, double b, int n, FunctionType func1) {
    double h;
    double x_i;
    double x_i_frac;
    double x_i_plus;
    double f_value;
    double f_value_frac;
    double f_value_plus;
    double result_sum = 0;
    h = (b - a) / n;
    for (int i = 0; i < n; i++) {
        x_i = a + (i * h);
        x_i_frac = a + ((i + 0.5) * h);
```

```
        x_i_plus = a + ((i + 1) * h);
        f_value = func1(x_i);
        f_value_frac = func1(x_i_frac);
        f_value_plus = func1(x_i_plus);
        result_sum += (f_value + 4 * f_value_frac + f_value_plus);
    }
    return (h / 6.0000) * result_sum;
}
```

调用方法与上述过程类似，这里不再赘述。

下面构造误差函数，思路是利用 for 循环计算不同的 n 次积分值，将 n 对应的 h 值和计算得到的积分值与标准 π 值做差得到误差，并将 h 与误差保存至 txt 文件中，并绘制误差值与 h 的图像。同时判断误差的绝对值是否小于 0.5×10^{-4}，若满足条件则打印当前的积分步距 $1/h$; 以下是计算误差并生成 txt 文件的函数，函数名为 handleCoordinates。

```
//创建用于绘制误差函数的txt坐标数据文件以便用tikz宏包绘制图像
void handleCoordinates(const char *filename, int numPoints, int begin_point,
    int end_point,double (*func)(double, double, int, double(*)(double))) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("无法创建文件 %s\n", filename);
        return;
    }
    fprintf(file, "h y\n"); // 列名
    int flag = 0;  //用于输出达到定义精度的标志符
    for (double i = 1; i < numPoints; i++) {
        double x = i;
        double y = fabs(M_PI - func(0, 1, i, f_4runge));
        fprintf(file, "%.15lf %.15lf\n", x, y);
        //判断计算结果是否达到所需精度
        if (y < 0.5e-4 && flag == 0)
        {
            printf("达到所需精度的h为: %f\n",i+1);
            flag = 1;
        }
    }
    fclose(file);
}
```

调用方式如下所示：

```
const char *filename1 = "中点公式误差坐标.txt";
handleCoordinates(filename1,25,0,1,remix_center_point);//25表示积分次数将从1到24
```

重复上述调用方法我们可以得到与 h 有关的误差函数，对自变量 h 对数处理后得到如图1所示的曲线：

根据图像我们可以发现：

当以 0.5×10^{-4} 为标准时，采用复合 Simpson 公式的求积方法的误差可以更快的收敛，同时复合中点求积公式方法的误差比复合梯形公式计算得到的误差收敛的略快。因此我们可以得出结论：

$$\text{复合 Simpson 公式求积精度} > \text{复合中点公式求积精度} > \text{复合梯形公式求积精度} \tag{4}$$

采用复合中点公式的求积方法的误差在 $h \in \left[\frac{1}{42}, 1\right]$ 上改进较快，继续往后则精度变化不大。

采用复合梯形公式的求积方法的误差在 $h \in \left[\frac{1}{59}, 1\right]$ 上改进较快，继续往后则精度变化不大。

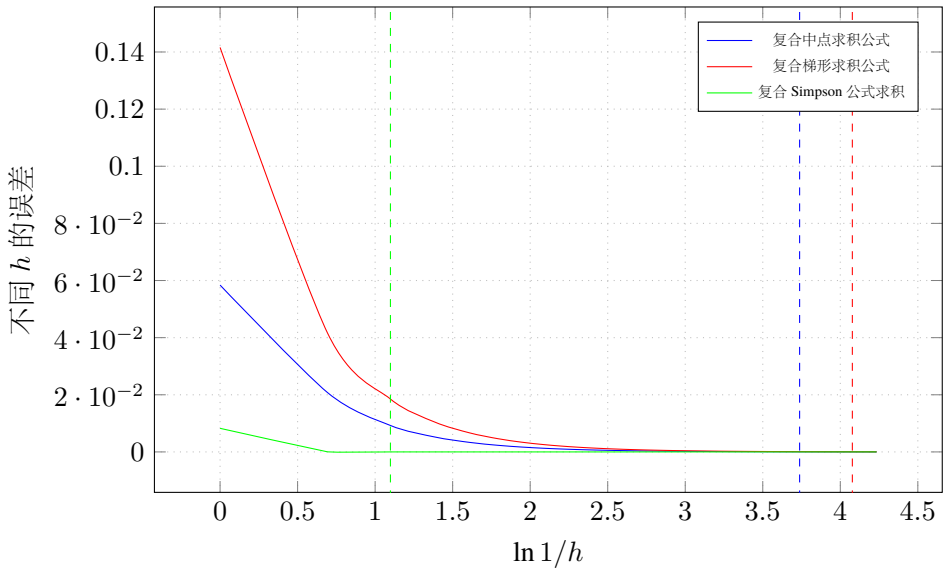

图 1: 复合方法误差函数

采用复合 Simpson 公式的求积方法的误差在 $h \in \left[\frac{1}{3}, 1\right]$ 上改进较快，继续往后则精度变化不大。

当积分步距 h（即每个子区间的宽度）足够小时，结果会越来越接近真实积分值。但一旦 h 变得足够小，进一步减小 h 将不再显著改进结果的精确度。这是因为数值积分方法的误差分为两部分：截断误差和舍入误差。截断误差是由于使用数值方法而不是解析方法来估算积分引入的误差。它通常随着 h 的减小而减小。当 h 足够小时，截断误差会逼近零，但进一步减小 h 不会显著改进结果，因为截断误差已经非常小了。舍入误差是由于计算机浮点数精度限制引入的误差。即使截断误差为零，舍入误差仍然存在。随着 h 的减小，舍入误差可能会变得更显著，因为计算机必须处理更多的小数位数。这也是为什么当 h 足够小时，进一步减小 h 可能会引入舍入误差，而不再改进结果的原因。

（2）Romberg 求积方法：

龙贝格积分法即在复化梯形积分公式的基础上使用 Richardson 外推，逐步提高积分的精度。

根据 Romberg 求积序列：

$$\begin{aligned} T_2(h) &= \frac{T_1(h/2) - 4^{-1}T_1(h)}{1 - 4^{-1}} \\ T_3(h) &= \frac{T_2(h/2) - 4^{-2}T_2(h)}{1 - 4^{-2}} \\ T_{k+1}(h) &= \frac{T_k(h/2) - 4^{-k}T_k(h)}{1 - 4^{-k}}, \quad k = 1, 2, \cdots \end{aligned} \tag{5}$$

【题目二】 已知积分

$$\int_{-\infty}^{+\infty} e^{-x^2} \cos x \mathrm{d}x = \sqrt{x} e^{\left(\frac{-1}{4}\right)} \tag{6}$$

使用如下三种方法进行数值积分：(1) 截断积分区间并使用复合求积公式进行计算. 通过选取不同的积分区间和复合公式中的步长，考察极限情况。并与精确积分值作比较。

(2) 使用自适应求积方法重复 (1) 中的计算。

(3) Gauss-Hermite 求积公式是为积分区间 $(-\infty, +\infty)$ 构造的，取权函数为 $e^(-x^2)$，所以用该方法来近似本体的积分是比较理想的。查找 Gauss-Hermite 求积公式在不同阶对应的积分节点和权数，计算本体的积分。