作业五：数值积分

英才 1701 赵鹏威 U201710152

2019 年 10 月 22 日

目录

1 引言

科学计算中常常会碰到求积分的问题，但是能够求出不定积分的情况非常有限。如果要在不求出不定积分的情况下求出定积分，就可以使用数值积分方法. 本次作业使用两种数值积分方法：复合梯形积分和复合辛普森积分.

2 问题描述

问题 1. 使用复合梯形积分和复合辛普森积分计算

$$\int_1^5 \sin(x)\mathrm{d}x,$$

要求步长为 $h = 0.1$，并估计误差.

这个积分可以求出真实值

$$\int_1^5 \sin(x)\mathrm{d}x = \cos(1) - \cos(5) \approx 0.25664012.$$

由于问题要求使用的步长为 0.1，因此需要将 1 和 5 等分成 40 段，进行复合积分.

3 程序实现

3.1 复合梯形积分

给定函数 f，积分上下限 a、b 和整数 n，复合梯形积分可以表示为

$$T_n(f) = \frac{h}{2}\left(f(a) + 2\sum_{i=1}^{n-1} f\left(x_i\right) + f(b)\right),$$

其中 $h = (b-a)/n$，$x_i = a + ih$. 复合梯形积分的误差用下式计算

$$E_n = \frac{4}{3}(T_{2n} - T_n).$$

复合梯形积分的流程图如图1所示. 代码见 Listing.1

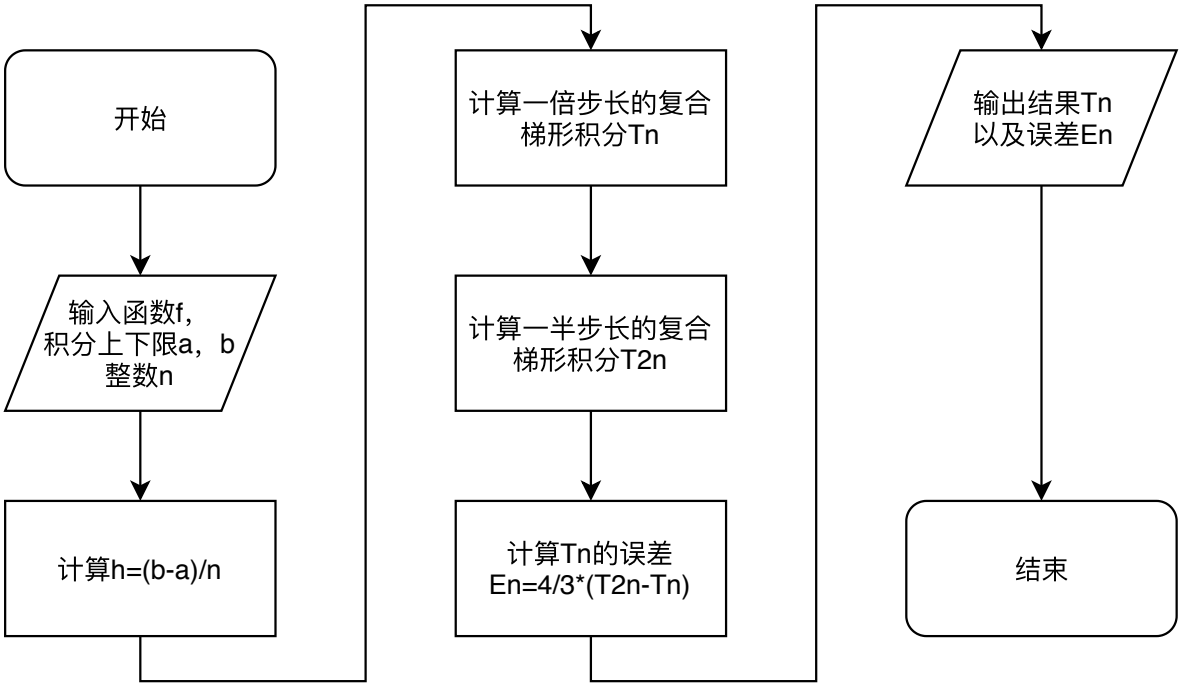

图 1: 复合梯形积分流程图

Listing 1: **trapezoid.f90**

```
1  subroutine trapezoid(f, a, b, n, I)
2      implicit none
3      real(8), external :: f
4      real(8), intent(in) :: a, b
5      integer, intent(in) :: n
6      type(result), intent(out) :: I
7
```

```
    real(8) :: h, I_hh
    integer :: j

    h = (b - a) / n
    I%value = (f(a) + f(b)) * h/2.0d0
    do j = 1, n-1
        I%value = I%value + 2.0d0*f(a+j*h) * h/2.0d0
    end do

    h = (b - a) / (2*n)
    I_hh = (f(a) + f(b)) * h/2.0d0
    do j = 1, 2*n-1
        I_hh = I_hh + 2.0d0*f(a+j*h) * h/2.0d0
    end do

    I%error = 4.0d0/3.0d0 * (I_hh - I%value)

    return
end subroutine
```

3.2 复合辛普森积分

给定函数 f，积分上下限 a、b 和偶数 $n = 2m$，复合梯形积分可以表示为

$$S_n(f) = \frac{h}{3}\left(f(a) + 4\sum_{i=0}^{m-1} f\left(x_{2i+1}\right) + 2\sum_{i=1}^{m-1} f\left(x_{2i}\right) + f(b)\right).$$

误差用下式计算

$$E_n = \frac{16}{15}(S_{2n} - S_n).$$

注意复合辛普森积分要求区间被分成偶数段，也就是 n 必须为偶数. 复合辛普森积分的流程图如图2所示. 代码见 Listing.2.

Listing 2: **simpson.f90**

```
subroutine simpson(f, a, b, n, I)
    implicit none
    real(8), external :: f
    real(8), intent(in) :: a, b
    integer, intent(in) :: n
    type(result), intent(out) :: I

    real(8) :: h, I_hh
    integer :: j

```

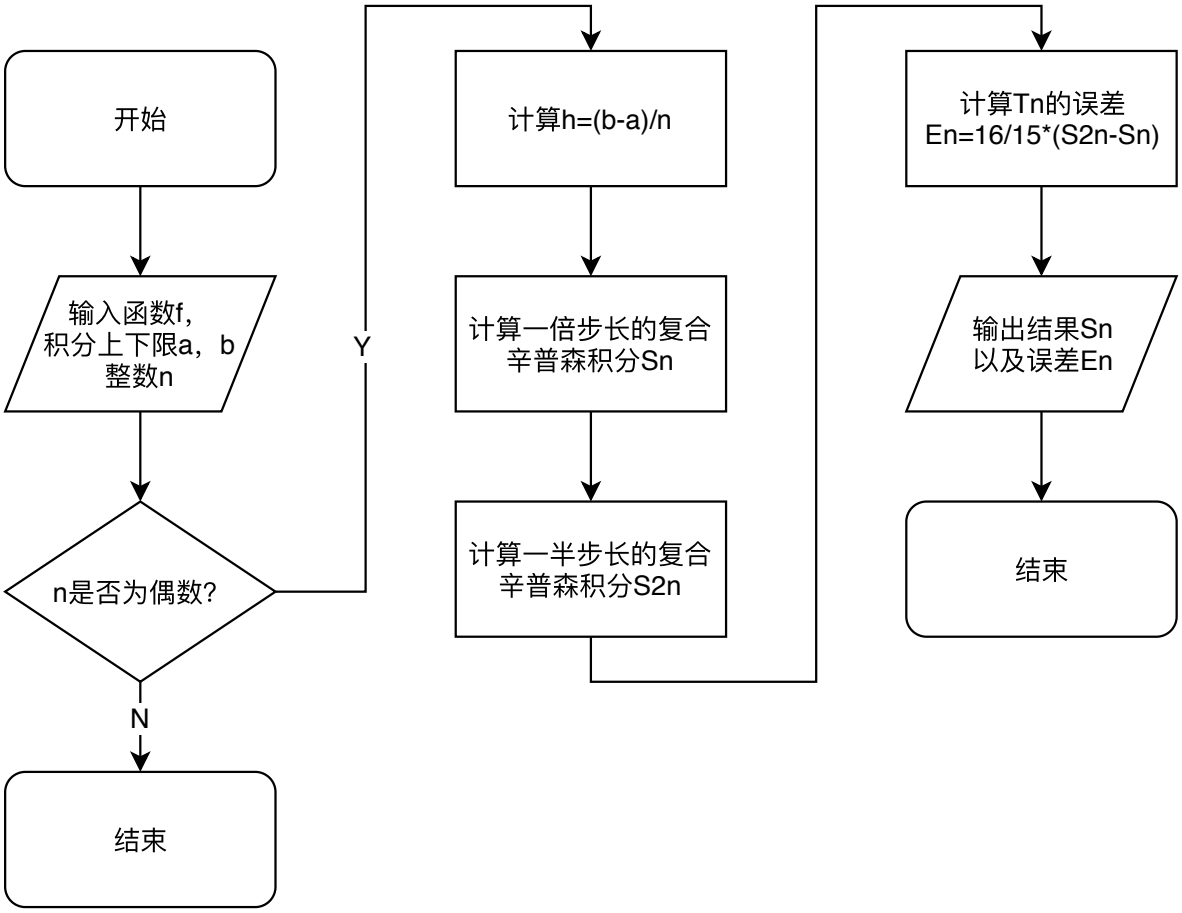

图 2: 复合辛普森积分流程图

```
11      if (mod(n, 2) == 1) stop "ERROR (Simpson): n must be even!"
12
13      h = (b - a) / n
14      I%value = (f(a) + f(b)) * h/3.0d0
15      do j = 0, n/2-1
16          I%value = I%value + 4.0d0*f(a+(2*j+1)*h) * h/3.0d0
17      end do
18      do j = 1, n/2-1
19          I%value = I%value + 2.0d0*f(a+2*j*h) * h/3.0d0
20      end do
21
22      h = (b - a) / (2*n)
23      I_hh = (f(a) + f(b)) * h/3.0d0
24      do j = 0, n-1
25          I_hh = I_hh + 4.0d0*f(a+(2*j+1)*h) * h/3.0d0
26      end do
```

```
27      do j = 1, n-1
28          I_hh = I_hh + 2.0d0*f(a+2*j*h) * h/3.0d0
29      end do
30
31      I%error = 16.0d0/15.0d0 * (I_hh - I%value)
32
33      return
34  end subroutine
```

4 运行时结果

程序的运行时结果如图3所示. 第 1 列数字是积分结果, 第 2 列数字是误差估计. 可以看到, 与真值相比, 复合梯形积分能够精确到小数点后第 3 位, 而复合辛普森积分能够精确到小数点后第 6 位. 另外, 可以发现, 如果将两种方法计算出来的数值积分结果加上对于的误差估计, 就能够很接近真值, 可见误差估计非常准确.

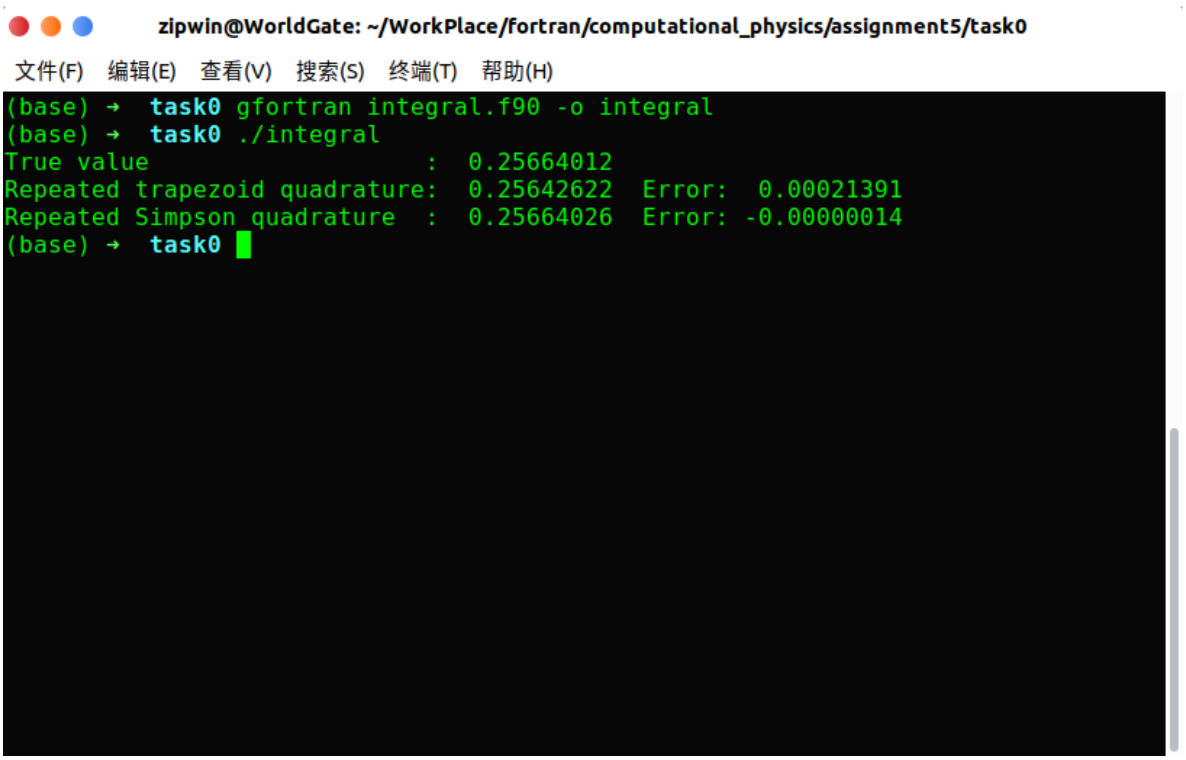

图 3: 运行时结果

附录

代码可在https://github.com/ZipWin/computational_physics/tree/master/assignments/assignment5找到.

Listing 3: **integral.f90**

```
module utils
    implicit none
    type result
        real(8) :: value, error
    end type

end module

module numerical_integral
    use utils
    implicit none
    contains
    subroutine trapezoid(f, a, b, n, I)
        implicit none
        real(8), external :: f
        real(8), intent(in) :: a, b
        integer, intent(in) :: n
        type(result), intent(out) :: I

        real(8) :: h, I_hh
        integer :: j

        h = (b - a) / n
        I%value = (f(a) + f(b)) * h/2.0d0
        do j = 1, n-1
            I%value = I%value + 2.0d0*f(a+j*h) * h/2.0d0
        end do

        h = (b - a) / (2*n)
        I_hh = (f(a) + f(b)) * h/2.0d0
        do j = 1, 2*n-1
            I_hh = I_hh + 2.0d0*f(a+j*h) * h/2.0d0
        end do

        I%error = 4.0d0/3.0d0 * (I_hh - I%value)

        return
    end subroutine
```

```
39
40      subroutine simpson(f, a, b, n, I)
41          implicit none
42          real(8), external :: f
43          real(8), intent(in) :: a, b
44          integer, intent(in) :: n
45          type(result), intent(out) :: I
46
47          real(8) :: h, I_hh
48          integer :: j
49
50          if (mod(n, 2) == 1) stop "ERROR (Simpson): n must be even!"
51
52          h = (b - a) / n
53          I%value = (f(a) + f(b)) * h/3.0d0
54          do j = 0, n/2-1
55              I%value = I%value + 4.0d0*f(a+(2*j+1)*h) * h/3.0d0
56          end do
57          do j = 1, n/2-1
58              I%value = I%value + 2.0d0*f(a+2*j*h) * h/3.0d0
59          end do
60
61          h = (b - a) / (2*n)
62          I_hh = (f(a) + f(b)) * h/3.0d0
63          do j = 0, n-1
64              I_hh = I_hh + 4.0d0*f(a+(2*j+1)*h) * h/3.0d0
65          end do
66          do j = 1, n-1
67              I_hh = I_hh + 2.0d0*f(a+2*j*h) * h/3.0d0
68          end do
69
70          I%error = 16.0d0/15.0d0 * (I_hh - I%value)
71
72          return
73      end subroutine
74
75  end module
76
77  program main
78      use utils
79      use numerical_integral
80      implicit none
81      real(8), external :: f
82      type(result) :: I
```

```
83
84      print "('True value               :', f12.8)", cos(1.0d0)-cos(5.0d0)
85      call trapezoid(f, 1.0d0, 5.0d0, 40, I)
86      print "('Repeated trapezoid quadrature:', f12.8, ' Error:', f12.8)", I%value, I%error
87      call simpson(f, 1.0d0, 5.0d0, 40, I)
88      print "('Repeated Simpson quadrature :', f12.8, ' Error:', f12.8)", I%value, I%error
89
90  end program
91
92  real(8) function f(x)
93      implicit none
94      real(8), intent(in) :: x
95
96      f = sin(x)
97
98      return
99  end function
```