

目 录

1	设计目的	1
2	算法概述	1
3	程序设计	2
4	问题求解	5
5	结果检验	8
6	实验总结	9

1 设计目的

1. 学会提高数值积分精度的技巧;
2. 掌握变步长的梯形算法;
3. 理解理查德森 (Richardson) 外推加速方法;
4. 使用龙贝格 (Romberg) 算法计算积分;
5. 加深对 MATLAB 中循环和矩阵寻访的理解.

2 算法概述

变步长的梯形法

使用复合求积公式时, 可将步长逐次分半. 每个子区间 $[x_k, x_{k+1}]$ 经过二分只增加了一个分点 $x_{k+\frac{1}{2}} = \frac{1}{2}(x_k + x_{k+1})$, 用复合梯形公式求得该子区间上的积分值为

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

这里 h 代表每次将区间二分前一次的步长. 将每个子区间上的积分值相加得

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

从而导出变步长梯形法的递推公式:

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

外推加速与龙贝格算法

理查德森 (Richardson) 外推加速公式为

$$T_m(h) = \frac{4^m}{4^m - 1} T_{m-1}\left(\frac{h}{2}\right) - \frac{1}{4^m - 1} T_{m-1}(h). \quad (2)$$

设以 $T_0^{(k)}$ 表示二分 k 次后求得的梯形值, 以 $T_m^{(k)}$ 表示序列 $\{T_0^{(k)}\}$ 的 m 次加速值, 则根据公式 2 可得龙贝格求积公式:

$$T_m^{(k)} = \frac{4^m}{4^m - 1} T_{m-1}^{(k+1)} - \frac{1}{4^m - 1} T_{m-1}^{(k)}, \quad k = 1, 2, \dots \quad (3)$$

龙贝格算法的计算过程如下:

1. 取二分次数 $k = 0, h = b - a$, 求 $T_0^{(0)} = \frac{h}{2}[f(a) + f(b)]$. 令 $1 \rightarrow k$.
2. 求梯形值 $T_0(\frac{b-a}{2^k})$, 即按公式 1 计算 $T_0^{(k)}$.
3. 求加速值. 按公式 3 逐个算出 T 表的其余各元素.
4. 若 $|T_k^{(0)} - T_{k-1}^{(0)}| < \varepsilon$ (预先给定精度), 则终止计算. 并取 $T_k^{(0)} = I$; 否则令 $k + 1 \rightarrow k$ 转步 2 继续计算.

3 程序设计

针对变步长梯形公式编写函数程序 `trapestep()` 如下:

```

1 function [ ] = trapestep(a, b, k)
2 % Step Changing Trapezoid Method
3 % Coded by Nan.Xiao 2010-05-23
4 % Step 1. Calc T_1
5 % Step 2. Calc T_2 ~ T_n
6 % Step 3. Output Result
7 format long
8 %     function y = f(x)
9 %         y=(2/sqrt(pi))*exp(-x);
10 %     end
11 %     Examples Can Also Be:
12 %     function y = f(x)
13 %         y=x.*sin(x);
14 %     end
15 %     Or
16     function y = f(x)
17         y=x*sqrt(1+x.^2);
18     end
19 temp=zeros(k,1);
20 t=zeros(k+1,1);
21 for i=1:k
22     for j=1:(2^(i-1))
23         temp(i,1)=temp(i,1)+f(a+(((2*j-1)*(b-a))/(2^i
           )))

```

```

24     end
25 end
26 mysum=temp;
27 t(1,1)=0.5*(f(a)+f(b));
28 for q=2:k+1
29     t(q,1)=(0.5*t(q-1,1))+(((b-a)*mysum(q-1,1))
        /(2^(q-1)));
30 end
31 disp('Step□Changing□Trapezoid□Results:');
32 disp(t);
33 end

```

针对龙贝格算法编写函数程序 romberg() 如下:

```

1 function [ ] = romberg(aa, bb, kk, iter)
2 % Romberg Integral Method
3 % Coded by Nan.Xiao 2010-05-23
4 % Step 1. Calc TrapeStep
5 % Step 2. Calc Romberg Method
6 % Step 3. Extract Diagonal & Output
7 function [ ] = trapestep(a, b, k)
8 format long
9     function y = f(x)
10         y=(2/sqrt(pi))*exp(-x);
11     end
12 %     Examples Can Also Be:
13 %     function y = f(x)
14 %         y=x.*sin(x);
15 %     end
16 %     Or
17 %     function y = f(x)
18 %         y=x*sqrt(1+x.^2);
19 %     end
20 temp=zeros(k,1);
21 t=zeros(k+1,1);
22 for i=1:k

```

```

23     for j=1:(2^(i-1))
24         temp(i,1)=temp(i,1)+f(a+(((2*j-1)*(b-a))/(2^i
           )));
25     end
26 end
27 mysum=temp;
28 t(1,1)=0.5*(f(a)+f(b));
29 for q=2:k+1
30     t(q,1)=(0.5*t(q-1,1))+(((b-a)*mysum(q-1,1))
           /(2^(q-1)));
31 end
32 end
33     format long
34 trapestep(aa, bb, kk);
35 diagonal=zeros(kk,1);
36 tt=zeros(kk,kk);
37 for v=1:kk
38     tt(v,1)=t(v,1);
39 end
40 for r=2:kk
41     for s=2:r
42         tt(r,s)=(((4^(s-1)*tt(r,s-1))-tt(r-1,s-1))
           /(4^(s-1)-1));
43     end
44 end
45 save('tt.txt', 'tt', '-ascii', '-double'); % Save
           the Table
46 % Extract Table Diagonals
47 for w=1:kk
48     diagonal(w,1)=tt(w,w);
49 end
50 disp('Romberg Table Diagonals:')
51 disp(diagonal);
52 % Output Final Result
53 cc=2;

```

```

54 while abs(diagonal(cc,1)-diagonal(cc-1,1))>=iter
      || abs(diagonal(cc+1,1)-diagonal(cc,1))>=iter
55     cc=cc+1;
56 end
57 disp('The Final Result is:');
58 disp(diagonal(cc,1));
59 end

```

4 问题求解

用龙贝格求积方法计算下列积分, 误差不超过 10^{-5} .

1. $\frac{2}{\sqrt{\pi}} \int_0^1 e^{-x} dx$;
2. $\int_0^{2\pi} x \sin x dx$;
3. $\int_0^3 x\sqrt{1+x^2} dx$.

解: 将上述程序保存为 `trapestep.m` 及 `romberg.m`, 根据题意, 将 `romberg.m` 的函数 `f` 修改为相应的被积函数. 在 MATLAB 中分别执行

```

>> romberg(0, 1, 25, 1e-6)
>> romberg(0, 2*pi, 25, 1e-6)
>> romberg(0, 3, 25, 1e-6)

```

即可得到如下计算结果:

```

Romberg Table Diagonals:
    0.771743332258054
    0.713512151168973
    0.713272026445579
    0.713271669814180
    0.713271669674931
    0.713271669674918
    0.713271669674918
    0.713271669674918
    0.713271669674918

```

```
0.713271669674918
0.713271669674918
0.713271669674919
0.713271669674919
0.713271669674917
0.713271669674920
0.713271669674918
0.713271669674913
0.713271669674918
0.713271669674917
0.713271669674925
0.713271669674924
0.713271669674935
0.713271669674912
0.713271669674885
0.713271669674887
```

The Final Result is:
0.713271669814180

Romberg Table Diagonals:

```
-0.0000000000000001
0.0000000000000001
-7.018385351885766
-6.266954014124826
-6.283266463460164
-6.283185210826781
-6.283185307207264
-6.283185307179584
-6.283185307179585
-6.283185307179586
-6.283185307179587
-6.283185307179588
-6.283185307179584
-6.283185307179580
```

-6.283185307179578
-6.283185307179603
-6.283185307179576
-6.283185307179590
-6.283185307179598
-6.283185307179588
-6.283185307179702
-6.283185307179473
-6.283185307179557
-6.283185307179426
-6.283185307179457

The Final Result is:
-6.283185307207264

Romberg Table Diagonals:

4.743416490252569
6.989465743280173
8.728844851089482
9.481516434622373
9.845941132884642
10.026942696106440
10.117289514478745
10.162443613495572
10.185018251490048
10.196305269082211
10.201948740203774
10.204770471055273
10.206181335892373
10.206886768237325
10.207239484400619
10.207415842481126
10.207504021521231
10.207548111041278
10.207570155801234

```
10.207581178181259
10.207586689371405
10.207589444966269
10.207590822763965
10.207591511662431
10.207591856111920

The Final Result is:
10.207591511662431
```

5 结果检验

使用 MATLAB 原生函数 `quad()`(自适应辛普森法) 检验上述 3 个定积分的计算结果:

```
1 f1 = @(x) (2/sqrt(pi))*exp(-x);
2 y1 = quad(f1 ,0, 1);
3 disp(y1);
4 f2 = @(x) x.*sin(x);
5 y2 = quad(f2, 0, 2*pi);
6 disp(y2);
7 f3 = @(x) x.*sqrt(1+x.^2);
8 y3 = quad(f3, 0, 3);
9 disp(y3);
```

即可得到如下计算结果:

```
0.713271671228492

-6.283185228932334

10.207592195132435
```

与程序的计算结果比较, 易知龙贝格算法的计算结果是正确的.

6 实验总结

1. 数值积分的核心问题是对函数值的平均高度 $h(\zeta)$ 提供一种算法. 就此衍生出了机械求积的概念. 对求积精度的改良也围绕着这个问题展开.
2. 龙贝格 (Romberg) 算法是在区间逐次分半过程中, 对用变步长梯形公式所获得的近似值进行多级“加工”, 以获得精度更高的积分近似值的方法.
3. 为了获得良好的可移植性, 在设计程序时仅使用了最基本的矩阵操作语句, 未使用 MATLAB 中的 `feval()` 函数. 这样做带来的问题是: 一旦改变被积函数, 就必须修改函数体.
4. 误差不超过 10^{-5} , 参数 `iter` 应设定为 `1e-6`.
5. 使用 `while` 语句判断积分结果的过程中, 不能仅与前一项作比较, 还要与后一项比较, 否则当遇到前两项相差较小, 但与真实值相差很大的情况时将判断错误.
6. 值得注意的是, 这两个程序的具体实现方式与前几个程序不同. 这两个程序是在 `for` 循环中每次累加改变和式的值, 即进行内存重用, 用时间换空间. 而非计算和式中的每一项存入矩阵然后直接相加. 若按照后法计算, 变步长梯形公式中的和式将需要生成阶数为 2 的指数次的矩阵, 矩阵达到一定阶数 (约四百万阶) 时将会出现问题. 而使用前一种方式, 理论上只要有足够的计算能力, 基本上可以得出计算结果.

参考文献

- [1] 李庆扬, 王能超, 易大义. 数值分析 (第 5 版). 北京: 清华大学出版社, 2008
- [2] 王能超. 计算方法——算法设计及其 MATLAB 实现. 北京: 高等教育出版社, 2005