MATLAB数值积分法

作者：凯鲁嘎吉 - 博客园
http://www.cnblogs.com/kailugaji/

一、实验目的

许多工程技术和数学研究中要用到定积分，如果无法直接算不出精确值（如含在积分方程中的积分）或计算困难但可用近似值近似时，就用数值积分法方法加以解决。常用的算法有：复化梯形、辛甫生（Simpson）、柯特斯（Cotes）求积法; 龙贝格(Romberg)算法；高斯（Gauss）算法。

二、实验原理

由Lagrange插值公式，记$A_i=\int_a^b l_i dx, R[f]=\int_a^b R_n(x)dx$，得 $I=\int_a^b f(x)dx=\sum_{i=0}^{n}A_i f(x_i)+R[f]$.

取$x_i=a+ih, h=\frac{b-a}{n}, i=0,1,2,\cdots,n$，得牛顿－柯特斯(Newton-Cotes)求积公式

$$I_n\approx\sum_{i=0}^{n}A_i f(x_i),\qquad 其中A_i=\frac{(-1)^{n-i}h}{i!(n-i)!}\int_0^n\prod_{\substack{0\le j\le n\\ j\ne i}}(s-j)ds,$$

(1)复化梯形公式 $$T_n=\sum_{k=0}^{n-1}\frac{h}{2}[f(x_k)+f(x_{k+1})]=\frac{h}{2}[f(a)+2\sum_{k=1}^{n-1}f(x_k)+f(b)].$$

(2)复化辛甫生公式 $$S_n=\frac{h}{6}[f(a)+4\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})+2\sum_{k=1}^{n-1}f(x_k)+f(b)]$$

(3)复化柯特斯公式 $$C_n=\frac{h}{90}[7f(a)+32\sum_{k=0}^{n-1}f(x_{k+\frac{1}{4}})+12\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})+32\sum_{k=0}^{n-1}f(x_{k+\frac{3}{4}})+14\sum_{k=1}^{n-1}f(x_k)+7f(b)]$$

(4)龙贝格公式 $$R_n=\frac{64}{63}C_{2n}-\frac{1}{63}C_n.$$

(5)高斯-勒让德公式 $$\int_{-1}^{1}f(x)dx\approx\frac{5}{9}f(-\frac{\sqrt{15}}{5})+\frac{8}{9}f(0)+\frac{5}{9}f(\frac{\sqrt{15}}{5}).$$

注： $$\int_a^b f(x)dx=\frac{b-a}{2}\int_{-1}^{1}f(\frac{b-a}{2}t+\frac{b+a}{2})dt.$$

三、实验程序

下面给出复化 S impson求积法程序（梯形及柯特斯复化求积分程序可比照编制）：

输入　端点a,b,正整数m

输出　I的近似值SI

$step\ 1\quad h \leftarrow \dfrac{b-a}{2m}$

$step\ 2\quad SI0 \leftarrow f(a)+f(b)$

$\qquad\qquad SI1 \leftarrow 0$

$\qquad\qquad SI2 \leftarrow 0$

$step\ 3$　对$i=1,\cdots,2m-1$做$step4-5$

$step\ 4\quad x \leftarrow a+ih$

$step\ 5$　若i是偶数，则$SI2 \leftarrow SI2+f(x)$

　　　　　　否则$SI1 \leftarrow SI1+f(x)$

$step\ 6\quad SI \leftarrow h\cdot(SI0+4\cdot SI1+2\cdot SI2)/3$

$step\ 7$　输出(SI)　停机

四、实验内容

选一可精确算值的定积分，用复化的梯形法及复化 S impson求积法作近似计算，并比较结果。

五、解答

1.(程序)

xps.m:

```
function y=xps(x)
y=x^(3/2);
```

复化梯形公式：

trap.m:

```
function [T,Y,esp]=trap(a,b,n)
h=(b-a)/n;
T=0;
for i=1:(n-1)
    x=a+h*i;
```

```
    T=T+xps(x);
end
T=h*(xps(a)+xps(b))/2+h*T;
syms x
Y=vpa(int(xps(x),x,a,b),8);
esp=abs(Y-T);
```

复化辛甫生（Simpson）公式：

simpson.m:

```
function [SI,Y,esp]=simpson(a,b,m)
%a,b为区间左右端点，xps(x)为求积公式，m*2等分区间长度
h=(b-a)/(2*m);
SI0=xps(a)+xps(b);
SI1=0;
SI2=0;
for i=1:((2*m)-1)
    x=a+i*h;
    if mod(i,2)==0
        SI2=SI2+xps(x);
    else
        SI1=SI1+xps(x);
    end
end
SI=h*(SI0+4*SI1+2*SI2)/3;
syms x
Y=vpa(int(xps(x),x,a,b),8);
esp=abs(Y-SI);
```

2.(运算结果)

```
>> [T,Y,esp]=trap(1,2,8)

T =

    1.8636


Y =

1.8627417


esp =
```

```
0.0008089288247354886607354274019599
>> [SI,Y,esp]=simpson(1,2,8)

SI =

    1.8627


Y =

1.8627417


esp =

0.000000020499792974248975951923057436943
```

从计算结果看：复化辛普森公式更精确。

3.(拓展（方法改进、体会等））

MATLAB中有一些内置函数，用于实施自适应求积分，都是根据Gander和Gautschi构造的算法编写的。

quad：使用辛普森求积，对于低精度或者不光滑函数效率更高

quadl：该函数使用了称为洛巴托求积（Lobatto Quadrature）的算法，对于高精度和光滑函数效率更高

使用方法：

I=quad(func,a,b,tol);

func是被积函数，a，b是积分限，tol是期望的绝对误差（如果不提供，默认为1e-6）

例如对于函数f=xe^x在[0,3]上求积分，显然可以通过解析解知道结果是2e^3+1=41.171073846375336

先创建一个M文件xex.m

内容如下：

function f=xex(x)

```
f=x.*exp(x);
```

然后调用：

```
>> format long
>> format compact
>> quad(@xex,0,3)
ans =
  41.171073850902332
```

可见有9位有效数字，精度还是蛮高的。

如果使用quadl：

```
>> quadl(@xex,0,3)
ans =
  41.171074668001779
```

反而只有7位有效数字