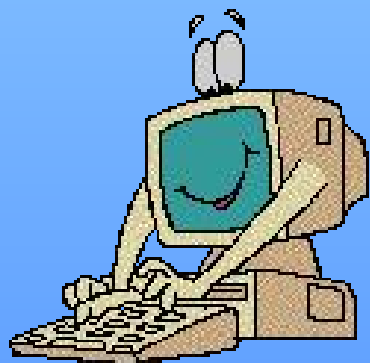


数值积分与数值微分

2023.07



本章要点

本章将用插值多项式 $P(x)$ 代替被积函数 $f(x)$,从而导出计算定积分 $\int_a^b f(x)dx$ 近似值的几个基本求积公式:

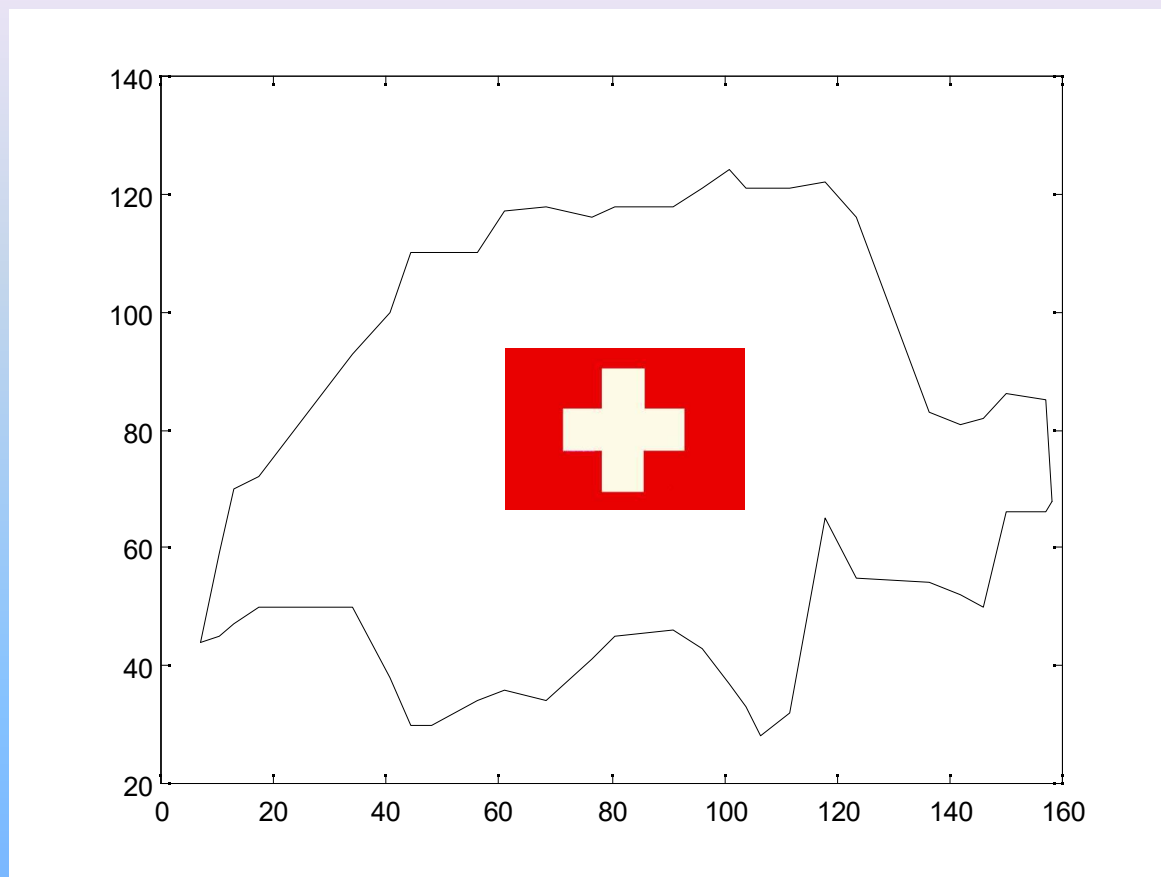
- (1) 等距节点下的:Newton-Cotes公式和Romberg公式
- (2) 数值微分公式

本章应用题:

为了计算瑞士国土的面积,首先对地图作了如下测量:以西向东方向为 x 轴,由南向北方向为 y 轴,选择方便的原点,并将从最西边界到最东边界在 x 轴上的区间适当地划分为若干段,在每个分点的 y 方向测出南边界点和北边界点的 y 坐标,数据如表(单位mm):

x	7.0	10.5	13.0	17.5	34.0	40.5	44.5	48.0	56.0
y1	44	45	47	50	50	38	30	30	34
y2	44	59	70	72	93	100	110	110	110
x	61.0	68.5	76.5	80.5	91.0	96.0	101.0	104.0	106.5
y1	36	34	41	45	46	43	37	33	28
y2	117	118	116	118	118	121	124	121	121
x	111.5	118.0	123.5	136.5	142.0	146.0	150.0	157.0	158.0
y1	32	65	55	54	52	50	66	66	68
y2	121	122	116	83	81	82	86	85	68

瑞士地图的外形如图(比例尺18mm:40km)



试由测量数据计算瑞士国土的近似面积,并与其精确值41288平方公里比较

§ 5.1 Newton-Cotes公式

对于积分 $I(f) = \int_a^b f(x) dx$

如果知道 $f(x)$ 的原函数 $F(x)$,则由 $Newton - Leibniz$ 公式有

$$\int_a^b f(x) dx = F(x) \Big|_a^b = F(b) - F(a)$$

但是在工程技术和科学研究中,常会见到以下现象:

- (1) $f(x)$ 的解析式根本不存在,只给出了 $f(x)$ 的一些数值
- (2) $f(x)$ 的原函数 $F(x)$ 求不出来,如 $F(x)$ 不是初等函数
- (3) $f(x)$ 的表达式结构复杂,求原函数较困难

以上这些现象,Newton-Leibniz很难发挥作用
只能建立积分的近似计算方法

这类方法很多,但为方便起见,最常用的一种方法是利用插值多项式来构造数值求积公式,具体步骤如下:


在积分区间 $[a,b]$ 上取一组节点

$$a \leq x_0 < x_1 < \cdots < x_n \leq b$$

作 $f(x)$ 的 n 次插值多项式

$$L_n(x) = \sum_{k=0}^n f(x_k) l_k(x)$$

$l_k(x) (k = 0, 1, \cdots, n)$ 为插值基函数



不同的
插值方法
有不同的
基函数

用 $L_n(x)$ 作为被积函数 $f(x)$ 的近似,有

$$\begin{aligned}\int_a^b f(x)dx &\approx \int_a^b L_n(x)dx = \int_a^b \sum_{k=0}^n f(x_k)l_k(x)dx \\ &= \sum_{k=0}^n f(x_k) \int_a^b l_k(x)dx\end{aligned}$$

若计 $A_k = \int_a^b l_k(x)dx$, 则

$$I(f) = \int_a^b f(x)dx \approx \sum_{k=0}^n A_k f(x_k) = I_n(f)$$

这就是数值求积公式 其中 A_k 称为求积系数

为了使一个求积公式能对更多的积分具有较好的实际计算意义,就要求它对尽可能多的被积函数都准确地成立

因此定义代数精度的概念:

定义1. 若求积公式

$$I(f) = \int_a^b f(x)dx \approx \sum_{k=0}^n A_k f(x_k) = I_n(f)$$

对任意次数不超过 m 次的代数多项式 $P_i(x) (i \leq m)$ 都准确成立,即

$$\int_a^b P_i(x)dx = \sum_{k=0}^n A_k P_i(x_k) \quad i = 0, 1, \dots, m$$

但对 $m+1$ 次多项式却不能准确成立,即只要

$$\int_a^b x^{m+1}dx \neq \sum_{k=0}^n A_k x_k^{m+1}$$

则称该求积公式具有 m 次的代数精度

代数精度也称
代数精确度

一、Newton-Cotes数值求积公式

Newton-Cotes公式是指等距节点下使用Lagrange插值多项式建立的数值求积公式

设函数 $f(x) \in C[a, b]$

将积分区间 $[a, b]$ 分割为 n 等份

各节点为 $x_k = a + kh$, $k = 0, 1, \dots, n$

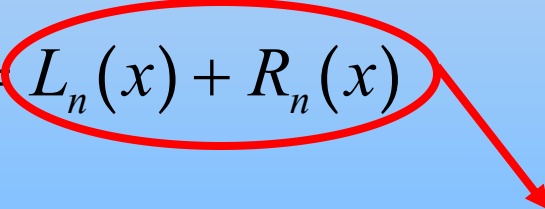
其中 $h = \frac{b-a}{n}$ 为步长

$f(x)$ 的Lagrange插值多项式及余项分别为

$$L_n(x) = \sum_{k=0}^n f(x_k) l_k(x) \quad R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$$

其中 $l_k(x) = \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j} \quad \xi \in [a, b] \quad \omega_{n+1}(x) = \prod_{i=0}^n (x - x_i)$

而 $f(x) = L_n(x) + R_n(x)$



因此对于定积分 $I(f) = \int_a^b f(x) dx$

有 $I(f) = \int_a^b f(x) dx = \int_a^b [L_n(x) + R_n(x)] dx$

$$\begin{aligned}
 I(f) &= \int_a^b f(x)dx = \int_a^b \sum_{k=0}^n f(x_k)l_k(x)dx + \int_a^b R_n(x)dx \\
 &= \sum_{k=0}^n A_k f(x_k) + \int_a^b R_n(x)dx
 \end{aligned}$$

其中 $A_k = \int_a^b l_k(x)dx = \int_a^b \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j} dx$

令 $I_n(f) = \sum_{k=0}^n A_k f(x_k)$

n阶Newton-Cotes求积公式

$$R(I_n) = \int_a^b R_n(x)dx$$

Newton-Cotes公式的余项(误差)

即有 $I(f) = I_n(f) + R(I_n)$

$$I(f) \approx I_n(f)$$

A_k 的计算:

注意是等距节点

$$A_k = \int_a^b l_k(x) dx = \int_a^b \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j} dx$$

假设 $x = a + th$ 由 $x \in [a, b]$ 可知 $t \in [0, n]$

$$A_k = \int_a^b \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j} dx = \int_0^n \left(\prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{(t - j)h}{(k - j)h} \right) \cdot h \cdot dt$$

$$= \frac{h \cdot (-1)^{n-k}}{k! \cdot (n-k)!} \int_0^n \prod_{\substack{0 \leq j \leq n \\ j \neq k}} (t - j) dt$$

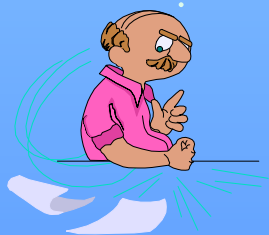
$$= (b-a) \cdot \frac{(-1)^{n-k}}{n \cdot k! \cdot (n-k)!} \int_0^n \prod_{\substack{0 \leq j \leq n \\ j \neq k}} (t-j) dt$$

$$A_k \triangleq (b-a) \cdot C_k^{(n)} \quad C_k^{(n)} \text{称为Cotes系数}$$

所以Newton-Cotes公式化为

$$I_n(f) = \sum_{k=0}^n A_k f(x_k) = (b-a) \sum_{k=0}^n C_k^{(n)} f(x_k)$$

思考



使用n次Lagrange插值多项式的Newton-Cotes公式至少具有n次代数精度,并且n为偶数时至少具有n+1次代数精度,试以n=1,2,4为例说明该结果

二、低阶Newton-Cotes公式及其余项

在Newton-Cotes公式中, $n=1,2,4$ 时的公式是最常用也最重要三个公式,称为低阶公式

1. 梯形(trapezia)公式及其余项

取 $n=1$, 则 $x_0=a$, $x_1=b$, $h=b-a$

Cotes系数为 $C_0^{(1)} = -\int_0^1 (t-1)dt = \frac{1}{2}$

$$C_1^{(1)} = \int_0^1 tdt = \frac{1}{2}$$

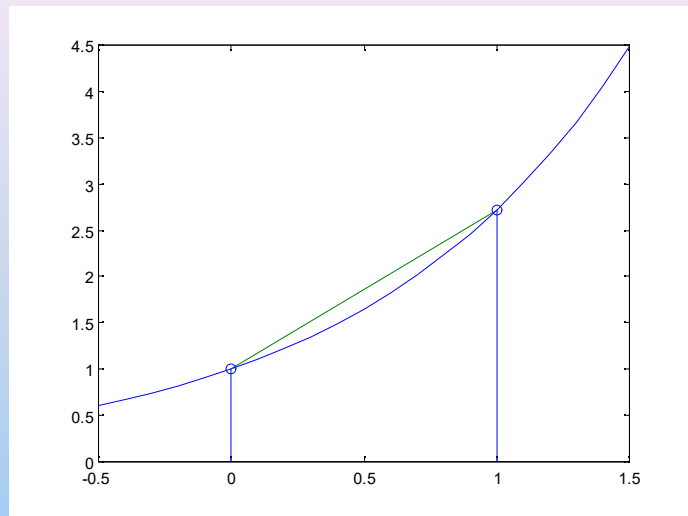
求积公式为

$$I_1(f) = (b-a) \sum_{k=0}^1 C_k^{(1)} f(x_k)$$

$$= \frac{b-a}{2} [f(x_0) + f(x_1)]$$

即

$$I_1(f) = \frac{b-a}{2} [f(a) + f(b)]$$



上式称为梯形求积公式,也称两点公式,记为

$$T = I_1(f) = \frac{(b-a)}{2} [f(a) + f(b)]$$

梯形公式的余项为

$$R(T) = R(I_1) = \int_a^b R_1(x) dx$$

$$R(T) = \int_a^b \frac{f''(\xi)}{2} (x-a)(x-b) dx$$

$$= \frac{f''(\eta)}{2} \int_a^b (x-a)(x-b) dx$$

$$= -\frac{f''(\eta)}{2} \frac{(b-a)^3}{6}$$

$$= -\frac{(b-a)^3}{12} f''(\eta)$$

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$$

第二积分
中值定理

$$\eta \in [a, b]$$

故

$$|R(T)| \leq \frac{(b-a)^3}{12} M_2$$

$$M_2 = \max_{x \in [a, b]} |f''(x)|$$

梯形(trapezia)公式具有1次代数精度

2.Simpson公式及其余项

取 $n = 2$, 则 $x_0 = a$, $x_1 = \frac{b+a}{2}$, $x_2 = b$, $h = \frac{b-a}{2}$

Cotes系数为 $C_0^{(2)} = \frac{1}{4} \int_0^2 (t-1)(t-2)dt = \frac{1}{6}$

$$C_1^{(2)} = \frac{-1}{2} \int_0^2 t(t-2)dt = \frac{4}{6}$$

$$C_2^{(2)} = \frac{1}{4} \int_0^2 (t-1)t dt = \frac{1}{6}$$

求积公式为

$$I_2(f) = (b-a) \sum_{k=0}^2 C_k^{(2)} f(x_k)$$

$$I_2(f) = (b-a) \left[\frac{1}{6} f(x_0) + \frac{4}{6} f(x_1) + \frac{1}{6} f(x_2) \right]$$

$$= \frac{b-a}{6} \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$

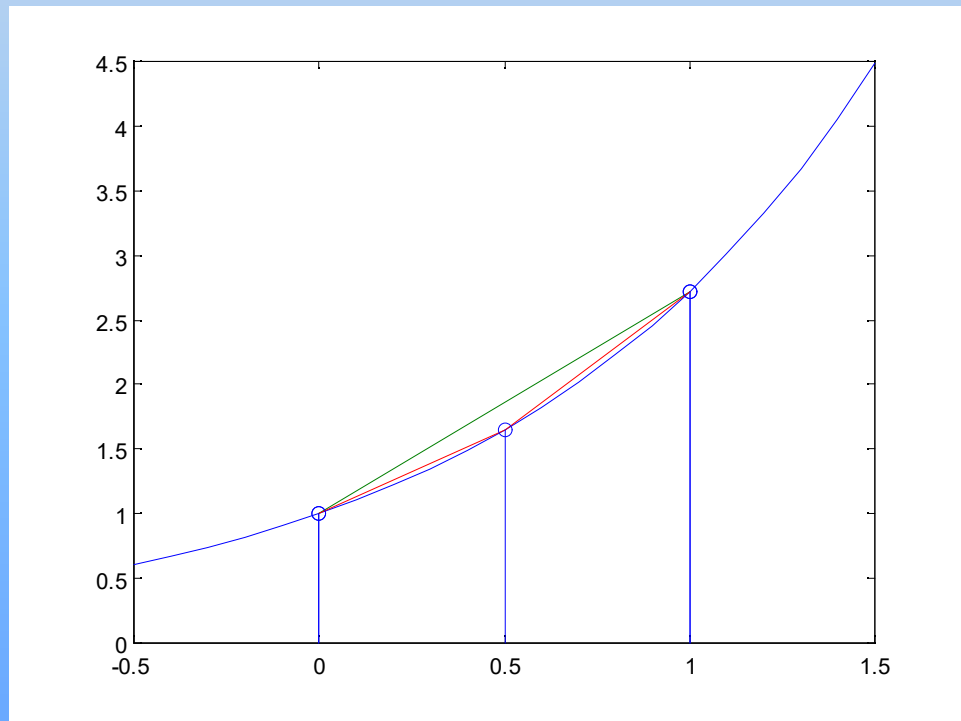
上式称为**Simpson求积公式**，也称**三点公式**或**抛物线公式**

记为 $S = I_2(f)$

Simpson公式的余项为

$$R(S) = R(I_2) = \int_a^b R_2(x) dx$$

$$= -\frac{b-a}{180} \left(\frac{b-a}{2}\right)^4 f^{(4)}(\eta)$$



Simpson公式具有3次代数精度

3.Cotes公式及其余项

取 $n = 4$, 则 $x_k = a + kh$, $k = 0, 1, \dots, 4$, $h = \frac{b-a}{4}$

Cotes系数为 $C_0^{(4)} = \frac{1}{4 \cdot 4!} \int_0^4 (t-1)(t-2)(t-3)(t-4)dt = \frac{7}{90}$

$$C_1^{(4)} = -\frac{1}{4 \cdot 3!} \int_0^4 t(t-2)(t-3)(t-4)dt = \frac{32}{90}$$

$$C_2^{(4)} = \frac{1}{4 \cdot 2! \cdot 2!} \int_0^4 t(t-1)(t-3)(t-4)dt = \frac{12}{90}$$

$$C_3^{(4)} = -\frac{1}{4 \cdot 3!} \int_0^4 t(t-1)(t-2)(t-4)dt = \frac{32}{90}$$

$$C_4^{(4)} = -\frac{1}{4 \cdot 4!} \int_0^4 t(t-1)(t-2)(t-3)dt = \frac{7}{90}$$

求积公式为

$$\begin{aligned} I_4(f) &= (b-a) \sum_{k=0}^4 C_k^{(4)} f(x_k) \\ &= (b-a) \left[\frac{7}{90} f(x_0) + \frac{32}{90} f(x_1) + \frac{12}{90} f(x_2) + \frac{32}{90} f(x_3) + \frac{7}{90} f(x_4) \right] \\ &= \frac{b-a}{90} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)] \end{aligned}$$

上式称为**Cotes求积公式**，也称**五点公式** 记为 $C = I_4(f)$

Cotes公式的余项为

$$R(C) = R(I_4) = \int_a^b R_4(x) dx = -\frac{2(b-a)}{945} \left(\frac{b-a}{4}\right)^6 f^{(6)}(\eta)$$

Cotes公式具有5次代数精度

例 用梯形公式和辛普森公式计算积分 $\int_0^1 e^{-x} dx$ ，并估计误差。

解 记 $I(f) = \int_0^1 e^{-x} dx$ ，则 $f'(x) = -e^{-x}$ ， $f''(x) = e^{-x}$

$$f'''(x) = -e^{-x}, f^{(4)}(x) = e^{-x} \implies |f''(x)| \leq 1, |f^{(4)}(x)| \leq 1$$

于是由梯形公式及其余项，有

$$I(f) = \int_0^1 e^{-x} dx \approx \frac{1-0}{2} (e^{-0} + e^{-1}) \approx 0.6839$$

$$|R_1[f]| = \left| \frac{-(b-a)^3}{12} f''(\eta) \right| \leq \frac{1}{12} \approx 0.08333$$

由辛普森公式及其余项，有

$$I(f) = \int_0^1 e^{-x} dx \approx \frac{1-0}{6} [e^{-0} + 4e^{-0.5} + e^{-1}] \approx 0.6323$$

$$|R_2[f]| = \left| -\frac{1}{90} \left(\frac{b-a}{2}\right)^5 f^{(4)}(\eta) \right| \leq \frac{1}{90} \left(\frac{1}{2}\right)^5 \approx 0.0003472$$

二者
精度？

三、Newton-Cotes公式的稳定性(舍入误差)

事实上,当 $n < 8$ 时, 公式都是稳定的

四、MATLAB命令

1、在MATLAB中，用梯形公式计算数值积分的库函数：

$Z=\text{trapz}(y)$ %输入数组为 y ，返回值为按梯形公式计算的积分值

$Z=\text{trapz}(x,y)$ % x 和 y 是维数相同的数组，返回值为 y 对 x 的积分

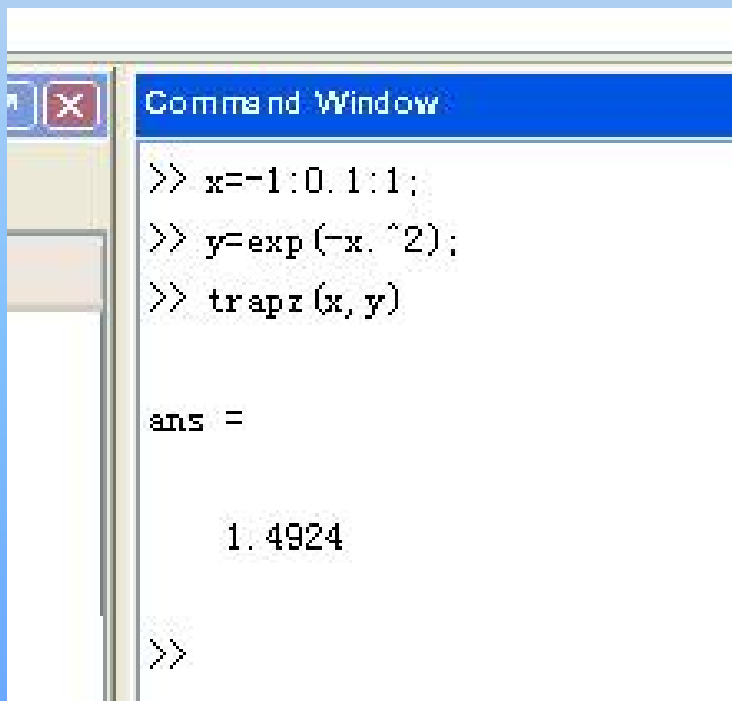
2、在MATLAB中，用辛普森公式计算数值积分的库函数：

$Z=\text{quad}('fun', a, b, tol)$ %计算以文件 $fun.m$ 命名的函数在区间 $[a, b]$ 上的积分，指定了相对误差为 tol

例 用 MATLAB 软件计算积分 $\int_{-1}^1 e^{-x^2} dx$.

$z=\text{trapz}(x,y)$ x 是表示积分区间的离散化向量;

y 与 x 同维数, 表示被积函数; z 返回积分的近似值.

A screenshot of the MATLAB Command Window. The window has a title bar with a close button. The text inside shows a sequence of commands: x=-1:0.1:1; y=exp(-x.^2); trapz(x,y). The output shows 'ans =' followed by '1.4924' on the next line. The prompt '>>' is visible at the bottom.

```
>> x=-1:0.1:1;  
>> y=exp(-x.^2);  
>> trapz(x,y)  
  
ans =  
  
1.4924  
  
>>
```

```
>> quad('exp(-x.^2)',-1,1)
```

```
ans =
```

```
1.4936
```

```
>>
```


§ 5.2 复合求积法

当积分区间 $[a, b]$ 的长度较大,而节点个数 $n + 1$ 固定时
直接使用Newton-Cotes公式的余项将会较大
而如果增加节点个数,即 $n + 1$ 增加时
公式的舍入误差又很难得到控制

为了提高公式的精度,又使算法简单易行,往往使用复合方法

即将积分区间 $[a, b]$ 分成若干个子区间

然后在每个小区间上使用低阶Newton-Cotes公式

最后将每个小区间上的积分的近似值相加

一、复合求积公式

将定积分 $\int_a^b f(x)dx$ 的积分区间 $[a, b]$ 分割为 n 等份

各节点为 $x_k = a + kh$, $k = 0, 1, \dots, n$ $h = \frac{b-a}{n}$

在子区间 $[x_k, x_{k+1}]$ ($k = 0, 1, \dots, n-1$) 上使用 *Newton - Cotes* 公式

将 $[x_k, x_{k+1}]$ 分割为 l 等份, 步长为 $\frac{h}{l}$, 节点为

$$x_k, x_k + \frac{h}{l}, x_k + \frac{2h}{l}, \dots, x_k + \frac{lh}{l} = x_{k+1}$$

记为 $x_k, x_{k+\frac{1}{l}}, x_{k+\frac{2}{l}}, \dots, x_{k+\frac{l}{l}} = x_{k+1}$

在 $[x_k, x_{k+1}]$ 上作 $f(x)$ 的 l 阶 $Newton - Cotes$ 求积公式

$$\begin{aligned}\int_{x_k}^{x_{k+1}} f(x)dx &\approx I_l^{(k)} = (x_{k+1} - x_k) \sum_{i=0}^l C_i^{(l)} f(x_{k+\frac{i}{l}}) \\ &= h \sum_{i=0}^l C_i^{(l)} f(x_{k+\frac{i}{l}})\end{aligned}$$

由积分的区间可加性,可得

$$\int_a^b f(x)dx = \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x)dx$$

复合求积公式

$$\approx \sum_{k=0}^{n-1} I_l^{(k)} = h \sum_{k=0}^{n-1} \sum_{i=0}^l C_i^{(l)} f(x_{k+\frac{i}{l}}) = I_n$$

$l = 1$ 时,可得复合梯形求积公式

$$\begin{aligned}\int_a^b f(x)dx &\approx T_n = h \sum_{k=0}^{n-1} \sum_{i=0}^1 C_i^{(1)} f(x_{k+i}) \\ &= h \sum_{k=0}^{n-1} \frac{1}{2} [f(x_k) + f(x_{k+1})]\end{aligned}$$

复合梯形公式

$$T_n = \frac{b-a}{2n} [f(a) + 2 \sum_{k=1}^{n-1} f(x_k) + f(b)]$$

$l = 2$ 时,可得复合 *Simpson* 求积公式

$$\int_a^b f(x)dx \approx S_n = h \sum_{k=0}^{n-1} \sum_{i=0}^2 C_i^{(2)} f(x_{k+\frac{i}{2}})$$

$$\int_a^b f(x)dx \approx S_n = h \sum_{k=0}^{n-1} \frac{1}{6} [f(x_k) + 4f(x_{k+\frac{1}{2}}) + f(x_{k+1})]$$

复合Simpson公式
复合抛物线公式

$$= \frac{b-a}{6n} [f(a) + 4 \sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}}) + 2 \sum_{k=1}^{n-1} f(x_k) + f(b)]$$

$l = 4$ 时,可得复合Cotes求积公式

复合Cotes公式

$$\int_a^b f(x)dx \approx C_n = h \sum_{k=0}^{n-1} \sum_{i=0}^4 C_i^{(4)} f(x_{k+\frac{i}{4}})$$

$$= \frac{h}{90} \sum_{k=0}^{n-1} [7f(x_k) + 32f(x_{k+\frac{1}{4}}) + 12f(x_{k+\frac{2}{4}}) + 32f(x_{k+\frac{3}{4}}) + 7f(x_{k+1})]$$

$$= \frac{b-a}{90n} [7f(a) + \sum_{k=0}^{n-1} [32f(x_{k+\frac{1}{4}}) + 12f(x_{k+\frac{2}{4}}) + 32f(x_{k+\frac{3}{4}})] + 14 \sum_{k=1}^{n-1} f(x_k) + 7f(b)]$$

$$T = \frac{(b-a)}{2} [f(a) + f(b)]$$

复合梯形公式分解

$$[x_0, x_1] \quad T^{(0)} = \frac{h}{2} [f(a) + f(x_1)]$$

$$[x_1, x_2] \quad T^{(1)} = \frac{h}{2} [f(x_1) + f(x_2)]$$

$$[x_2, x_3] \quad T^{(2)} = \frac{h}{2} [f(x_2) + f(x_3)]$$

$$[x_3, x_4] \quad T^{(3)} = \frac{h}{2} [f(x_3) + f(x_4)]$$

$$[x_{n-2}, x_{n-1}] \quad T^{(n-2)} = \frac{h}{2} [f(x_{n-2}) + f(x_{n-1})]$$

$$[x_{n-1}, x_n] \quad T^{(n-1)} = \frac{h}{2} [f(x_{n-1}) + f(b)]$$

$$T_n = \frac{h}{2} [f(a) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(b)]$$

$$S = \frac{b-a}{6} [f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

复合Simpson公式分解

$$[x_0, x_1] \quad S^{(0)} = \frac{h}{6} [f(a) + 4f(x_{0+\frac{1}{2}}) + f(x_1)]$$

$$[x_1, x_2] \quad S^{(1)} = \frac{h}{6} [f(x_1) + 4f(x_{1+\frac{1}{2}}) + f(x_2)]$$

$$[x_{n-1}, x_n] \quad S^{(n-1)} = \frac{h}{6} [f(x_{n-1}) + 4f(x_{n-1+\frac{1}{2}}) + f(b)]$$

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

例1. 使用各种复合求积公式计算定积分 $I = \int_0^1 \frac{\sin x}{x} dx$

解: 为简单起见,依次使用8阶复合梯形公式、4阶复合Simpson公式和2阶复合Cotes公式

可得各节点的值如右表

<i>Trapz</i>	<i>Simp.</i>	<i>Cotes</i>	x_i	$f(x_i)$
x_0	x_0	x_0	0	1
x_1	$x_{0+\frac{1}{2}}$	$x_{0+\frac{1}{4}}$	0.125	0.99739787
x_2	x_1	$x_{0+\frac{1}{2}}$	0.25	0.98961584
x_3	$x_{1+\frac{1}{2}}$	$x_{0+\frac{3}{4}}$	0.375	0.97672674
x_4	x_2	x_1	0.5	0.95885108
x_5	$x_{2+\frac{1}{2}}$	$x_{1+\frac{1}{4}}$	0.625	0.93615564
x_6	x_3	$x_{1+\frac{1}{2}}$	0.75	0.90885168
x_7	$x_{3+\frac{1}{2}}$	$x_{1+\frac{3}{4}}$	0.875	0.87719257
x_8	x_4	x_2	1	0.84147098

复合求积
公式的程序
newtoncotes.m

函数程序
func.m

分别由复合Trapz、Simpson、Cotes公式有

$$T_8 = \frac{1}{16} [f(0) + 2 \sum_{k=1}^7 f(x_k) + f(1)] = 0.94569086$$

$$S_4 = \frac{1}{24} [f(0) + 4 \sum_{k=0}^3 f(x_{k+\frac{1}{2}}) + 2 \sum_{k=1}^3 f(x_k) + f(1)]$$

$$= 0.94608331$$

$$C_2 = \frac{1}{180} [7f(0) + \sum_{k=0}^1 [32f(x_{k+\frac{1}{4}}) + 12f(x_{k+\frac{2}{4}}) + 32f(x_{k+\frac{3}{4}})] + 14 \sum_{k=1}^1 f(x_k) + 7f(1)]$$

$$= 0.94608307$$

比较三个
公式的结果

精度最低

$$T_8 = 0.94569086$$

精度次高

$$S_4 = 0.94608331$$

精度最高

$$C_2 = 0.94608307$$

原积分的精确值为 $I = \int_0^1 \frac{\sin x}{x} dx = 0.946083070367183 \dots$

那么哪个复合求积公式的收敛最快呢？

二、复合求积公式的余项和收敛的阶

我们知道,三个求积公式的余项分别为

单纯的求积公式

$$R(T) = -\frac{(b-a)^3}{12} f''(\eta)$$

$$R(S) = -\frac{b-a}{180} \left(\frac{b-a}{2}\right)^4 f^{(4)}(\eta)$$

$$R(C) = -\frac{2(b-a)}{945} \left(\frac{b-a}{4}\right)^6 f^{(6)}(\eta)$$

复合求积公式的每个小区间

$$= -\frac{h}{12} \cdot h^2 \cdot f''(\eta_k)$$

$$= -\frac{h}{180} \left(\frac{h}{2}\right)^4 f^{(4)}(\eta_k)$$

$$= -\frac{2h}{945} \left(\frac{h}{4}\right)^6 f^{(6)}(\eta_k)$$

1. 设被积函数 $f(x) \in C^2[a, b]$, 则复合梯形公式的余项为

$$I - T_n = \sum_{k=0}^{n-1} \left[-\frac{h^3}{12} f''(\eta_k) \right] = -\frac{h^3}{12} \sum_{k=0}^{n-1} f''(\eta_k)$$

由于

$$\min_{a \leq x \leq b} \{f''(x)\} \leq \sum_{k=0}^{n-1} \frac{f''(\eta_k)}{n} \leq \max_{a \leq x \leq b} f''(x)$$

由介值定理, $\exists \eta \in [a, b]$, 使得

$$\sum_{k=0}^{n-1} \frac{f''(\eta_k)}{n} = f''(\eta)$$

即有

$$\begin{aligned} I - T_n &= -\frac{nh^3}{12} \sum_{k=0}^{n-1} \frac{f''(\eta_k)}{n} = -\frac{nh^3}{12} f''(\eta) = -\frac{(b-a)}{12} h^2 f''(\eta) \\ &= R(T_n) \end{aligned}$$

又由

$$I - T_n = -\frac{h^3}{12} \sum_{k=0}^{n-1} f''(\eta_k)$$

$$\frac{I - T_n}{h^2} = -\frac{1}{12} \sum_{k=0}^{n-1} f''(\eta_k) h = -\frac{1}{12} \sum_{k=0}^{n-1} f''(\eta_k) \Delta x_k$$

$$\rightarrow -\frac{1}{12} \int_a^b f''(x) dx = -\frac{1}{12} [f'(b) - f'(a)] \quad \begin{pmatrix} h \rightarrow 0 \\ n \rightarrow \infty \end{pmatrix}$$

因此当 n 足够大时,复合梯形公式的余项为

$$I - T_n \approx -\frac{h^2}{12} [f'(b) - f'(a)] = -\frac{1}{12} h^2 [f'(b) - f'(a)]$$

2. 若被积函数 $f(x) \in C^4[a, b]$

则 n 足够大时, 复合 *Simpson* 公式的余项为

$$\begin{aligned} I - S_n &= \sum_{k=0}^{n-1} \left(-\frac{h^5}{180 \cdot 2^4} f^{(4)}(\eta_k) \right) = -\frac{b-a}{180} \left(\frac{h}{2} \right)^4 f^{(4)}(\eta) \\ &\approx -\frac{h^4}{180 \cdot 2^4} [f'''(b) - f'''(a)] = -\frac{1}{180} \left(\frac{h}{2} \right)^4 [f'''(b) - f'''(a)] \end{aligned}$$

3. 若 $f(x) \in C^6[a, b]$, 同样可得复合 *Cotes* 公式的余项

$$\begin{aligned} I - C_n &= \sum_{k=0}^{n-1} \left(-\frac{2h^7}{945 \cdot 4^6} f^{(6)}(\eta_k) \right) = -\frac{2(b-a)}{945} \left(\frac{h}{4} \right)^6 f^{(6)}(\eta) \\ &\approx -\frac{2h^6}{945 \cdot 4^6} [f^{(5)}(b) - f^{(5)}(a)] = -\frac{2}{945} \left(\frac{h}{4} \right)^6 [f^{(5)}(b) - f^{(5)}(a)] \end{aligned}$$

比较三种复合公式的余项

$$I - T_n \approx -\frac{1}{12}h^2[f'(b) - f'(a)] = o(h^2)$$

$$I - S_n \approx -\frac{1}{180}\left(\frac{h}{2}\right)^4[f'''(b) - f'''(a)] = o(h^4)$$

$$I - C_n \approx -\frac{2}{945}\left(\frac{h}{4}\right)^6[f^{(5)}(b) - f^{(5)}(a)] = o(h^6)$$

分别是 h 的2, 4, 6阶无穷小量

即 T_n, S_n, C_n 趋于定积分 I 的速度依次更快

不难知道,复合梯形、Simpson、Cotes公式的收敛阶分别为

2阶、4阶和6阶

而积分区间 $[a,b]$ 分割的小区间数 n 越大, I_n 精度越高

但 n 太大,运算量也很大

n 太小,运算量虽较小,但精度可能又达不到

三、复合自适应求积法的算法设计

规定步长的求积法的算法比较简单,这里只以自动选取步长的复合Simpson求积公式为例介绍自适应法的算法设计

求定积分
$$I = \int_a^b f(x)dx$$

(一) 算法名称

autosimpson(fun, a, b, EPS)

(二) 存储方式

a 存积分下限 a $y1$ 存函数值 $f(a)$

b 存积分上限 b $y2$ 存函数值 $f(b)$

EPS 存预先给定的误差限 ε

$I1$ 存前一次积分值 $I2$ 存后一次积分值

x 一维向量，存分段节点

f 一维向量，存节点处函数值

$S1$ 存旧节点处函数值之和

$S2$ 存新增节点处函数值之和

(三) 自然语言

1. 输入积分上下限 a, b , 误差限 ε
2. $n = 1, k = 1, s1 = 0, I1 = 0$
3. $y1 = f(a), y2 = f(b)$
4. $s0 = y1 + y2$
5. $h = (b - a) / n$
6. $s2(k) = 0$
7. $j = 1, 2, \dots, n$

$$7(1). \quad x = a + (2j - 1)\frac{h}{2}$$

$$7(2). \quad s2(k) = s2(k) + f(x)$$

$$8. \quad \text{if } k > 1, s1 = s1 + s2(k - 1)$$

$$9. \quad I2 = [s0 + 2s1 + 4s2(k)]\frac{h}{6}$$

$$10. \quad d = abs(I2 - I1) / 15$$

$$11. \quad \text{if } d > EPS$$

$k = k + 1, I1 = I2, n = n + n, \text{ goto } 5, \text{ 否则}$

12. 输出 $I2, h$

13. 停机

(四) 程序实例

程序名: Autosimpson.m

命令格式: autosimpson('fun',a,b,eps)

例2. 用自适应Simpson公式计算下列定积分,并比较

$$I = \int_0^1 e^{-x^2} dx$$

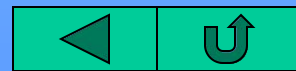
解:

Li42.m

区间数及方法		积分结果	运行时间	误差
m =		z =	t =	d =
n=8	trapz	0.745865614845695	0	-9.6e-04
n=20	trapz	0.746670836939873	0	-1.5e-04
n=50	trapz	0.746799607189351	0	-2.5e-05
n=100	trapz	0.74681800146797	0.06	-6.1e-06
n=4	simpson	0.746826120527467	0	2.0e-06
n=20	simpson	0.746824136005348	0	3.2e-09
n=4	cotes	0.746824133229615	0	4.2e-10
autosimp	1e-4	0.746826120527467	0.11	2.0e-06
autosimp	1e-6	0.746824140606985	0.16	7.8e-09
autosimp	1e-10	0.74682413281433	0.22	1.9e-12

z1= 0.746824132812427

h =0.25	4
h =0.0625	16
h =0.0078125	128



§ 5.3 Romberg算法

由复合梯形公式的余项公式

$$I - T_{2n} \approx \frac{1}{3}(T_{2n} - T_n) = \frac{1}{4-1}(T_{2n} - T_n)$$

可知利用两种步长计算的结果能估计截断误差.

$$\Rightarrow I \approx T_{2n} + \frac{1}{3}(T_{2n} - T_n)$$

可得出 “改进梯形求积公式”

$$\begin{aligned}\bar{T} &= T_{2n} + \frac{1}{3}(T_{2n} - T_n) = \frac{4}{3}T_{2n} - \frac{1}{3}T_n \\ &= \frac{4}{3}\left[\frac{1}{2}T_n + \frac{h}{2}\sum_{i=0}^{n-1}f(x_{i+\frac{1}{2}})\right] - \frac{1}{3}T_n = \frac{2}{3}T_n + \frac{4h}{6}\sum_{i=0}^{n-1}f(x_{i+\frac{1}{2}}) - \frac{1}{3}T_n \\ &= \frac{1}{3}T_n + \frac{4h}{6}\sum_{i=0}^{n-1}f(x_{i+\frac{1}{2}})\end{aligned}$$

$$\bar{T} = \frac{1}{3}T_n + \frac{4h}{6} \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}})$$

再将 $T_n = \frac{h}{2} \sum_{i=0}^{n-1} [f(x_i) + f(x_{i+1})]$ 代入得

$$\bar{T} = \frac{4}{3}T_{2n} - \frac{1}{3}T_n = \frac{h}{6} \sum_{i=0}^{n-1} [f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1})] = S_n$$

用梯形法二分前后的两个积分值 T_n 与 T_{2n} 的线性组合的结果

可得到复化辛普森求积公式

$$S_n = \frac{4}{3}T_{2n} - \frac{1}{3}T_n = \frac{4}{4-1}T_{2n} - \frac{1}{4-1}T_n$$

用辛浦生法二分前后的两个积分值 S_n 与 S_{2n} 的线性组合的结果可得到复化科特斯求积公式

$$C_n = S_{2n} + \frac{1}{15}(S_{2n} - S_n) = \frac{4^2}{4^2 - 1}S_{2n} - \frac{1}{4^2 - 1}S_n$$

用科特斯法二分前后的两个积分值 C_n 与 C_{2n} 的线性组合可得更精确的结果,称为龙贝格 (Romberg) 求积公式

$$R_n = C_{2n} + \frac{1}{63}(C_{2n} - C_n) = \frac{4^3}{4^3 - 1}C_{2n} - \frac{1}{4^3 - 1}C_n$$

外推方法

将序列 $\{T_n\}$, $\{S_n\}$, $\{C_n\}$ 和 $\{R_n\}$ 分别称为
梯形序列、辛普森序列、科特斯序列和龙贝格序列。

利用龙贝格序列求积的算法称为龙贝格算法。

这种算法具有占用内存少、精确度高的优点。

龙贝格求积算法的**计算步骤**可按下面的公式依次进行：

$$\left\{ \begin{array}{l} T_1 = \frac{b-a}{2} [f(a) + f(b)] \\ T_{2n} = \frac{1}{2} T_n + \frac{b-a}{2} \sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}}) \\ S_n = \frac{4}{3} T_{2n} - \frac{1}{3} T_n \\ C_n = \frac{16}{15} S_{2n} - \frac{1}{15} S_n \\ R_n = \frac{64}{63} C_{2n} - \frac{1}{63} C_n \end{array} \right.$$

重复上述过程可计算得到 R_1, R_2, R_4, \dots .

停止计算:龙贝格序列中前后两项之差的绝对值
不超过给定的误差限为止.

将上述结果综合后

$$\left\{ \begin{aligned} T_0(0) &= \frac{b-a}{2} [f(a) + f(b)] \\ T_0(k) &= \frac{1}{2} T_0(k-1) + \frac{b-a}{2^k} \sum_{j=0}^{2^{k-1}-1} f\left(a + (2j+1) \frac{b-a}{2^k}\right) \end{aligned} \right.$$

$$T_1(k-1) = \frac{4}{3} T_0(k) - \frac{1}{3} T_0(k-1)$$

$$T_2(k-1) = \frac{16}{15} T_1(k) - \frac{1}{15} T_1(k-1)$$

$$T_3(k-1) = \frac{64}{63} T_2(k) - \frac{1}{63} T_2(k-1)$$

$$k = 1, 2, \dots$$

外推
加速
公式

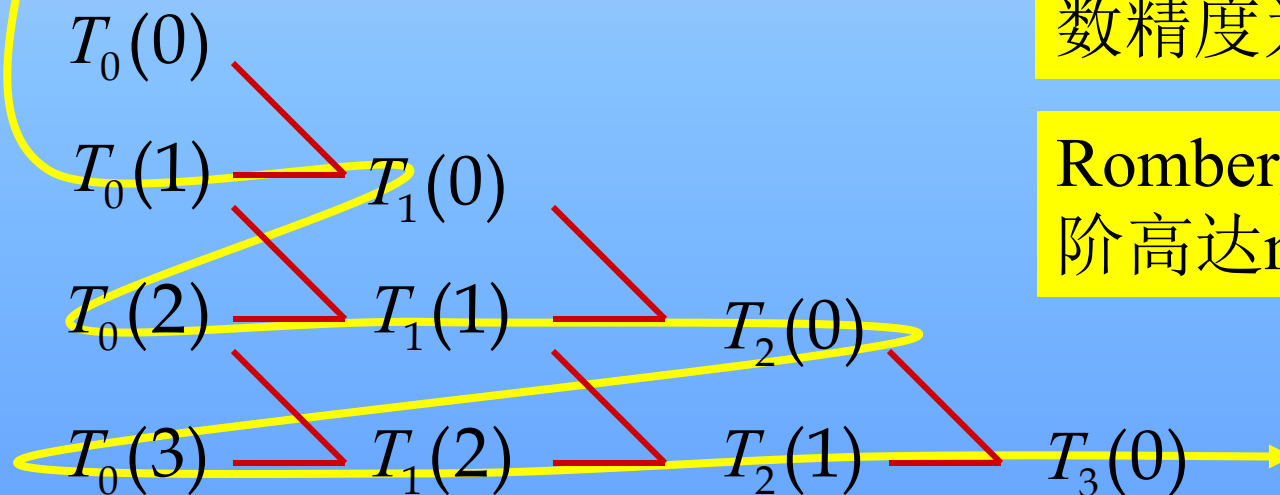
以上整个过程称为Romberg算法

其中外推加速公式可简化为

$$T_m(k-1) = \frac{1}{4^m - 1} [4^m T_{m-1}(k) - T_{m-1}(k-1)] \quad \text{-----}(9)$$

并且 m 可以推广到 $m = 1, 2, \dots$ $k = 1, 2, \dots$

Romberg算法求解步骤



Romberg算法的代数精度为 m 的两倍

Romberg算法的收敛阶高达 $m+1$ 的两倍

表 龙贝格算法计算步骤

k	梯形序列 T_{2^k}	辛浦生序列 $S_{2^{k-1}}$	柯特斯序列 $C_{2^{k-2}}$	龙贝格序列 $R_{2^{k-3}}$
0	T_1			
1	T_2	S_1		
2	T_4	S_2	C_1	
3	T_8	S_4	C_2	R_1
4	T_{16}	S_8	C_4	R_2
5	T_{32}	S_{16}	C_8	R_4
\vdots	\vdots	\vdots	\vdots	\vdots

例： 用龙贝格算法求下列积分，要求误差不超过 $\frac{1}{2} \times 10^{-5}$.

$$(1). \int_0^{1.5} \frac{1}{x+1} dx;$$

解：龙贝格求积算法的计算步骤
可按下面的公式依次计算

0	1.0500	0	0	0	0
1	0.9536	0.9214	0	0	0
2	0.9260	0.9168	0.9165	0	0
3	0.9187	0.9163	0.9163	0.9163	0
4	0.9169	0.9163	0.9163	0.9163	0.9163

$$\begin{cases} T_1 = \frac{b-a}{2} [f(a) + f(b)] \\ T_{2n} = \frac{1}{2} T_n + \frac{b-a}{2} \sum_{k=0}^{n-1} f(x_{k+1/2}) \\ S_n = \frac{4}{3} T_{2n} - \frac{1}{3} T_n \\ C_n = \frac{16}{15} S_{2n} - \frac{1}{15} S_n \\ R_n = \frac{64}{63} C_{2n} - \frac{1}{63} C_n. \end{cases}$$

$$\text{因此 } \int_0^{1.5} \frac{1}{x+1} dx \approx 0.9163.$$

例： 计算积分 $\int_0^{\frac{\pi}{2}} \sin x dx$

romberg.m

Jifenbijiao.m

积分法

积分值

绝对误差

前侧矩形公式	z1 = 0.99212545660563	-0.00787454339437
	z11 = 0.99212545660563	-0.00787454339437
后侧矩形公式	z2 = 1.00783341987358	0.00783341987358
	z22 = 1.00783341987358	0.00783341987358
梯形公式	z3 = 0.99997943823961	-0.00002056176039
Simpson公式	z4 = 1.00000000201613	0.00000000201613
8阶simpson公式	z5 = 1.00000000000000	-0.00000000000000
自选步长梯形公式	z6 = 0.99999921563419	-0.00000078436581
自选步长Simpson公式	z7 = 1.00000051668471	0.00000051668471
Romberg公式	z8 = 0.999999999999802	-0.000000000000198
Mote-Carlo算法	z9 = 0.99821071589516	-0.00178928410484

§ 5.4 数值微分

先看一个实例:

已知20世纪美国人口的统计数据为(单位:百万)

年份	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990
人口	76.0	92.0	106.5	123.2	131.7	150.7	179.3	204.0	226.5	251.4

试计算美国20世纪的(相对)年增长率

若记 t 时刻的人口为 $x(t)$,则人口的增长率为

$$r(t) = \frac{dx/dt}{x(t)}$$

dx/dt 如何求

一、差商代替微商

若精度要求不高，可以简单地取差商作为导数的近似值。

向前差商公式 $f'(a) \approx \frac{f(a+h) - f(a)}{h}$

向后差商公式 $f'(a) \approx \frac{f(a) - f(a-h)}{h}$

中心差商公式 $f'(a) \approx \frac{f(a+h) - f(a-h)}{2h}$

其中 h 为一增量,称为步长.

余项分别为 $O(h), O(h), O(h^2)$.

二、插值型数值微分公式

用函数 $f(x)$ 的插值函数的微商近似代替 $f(x)$ 的微商.

$$f'(x) \approx P'(x) \quad \cdots \cdots (5.44)$$

若限定求某个节点 x_i 上的导数值,则求导公式的余项为

$$R'_n(x_i) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega'_{n+1}(x_i) \quad \cdots \cdots (5.45)$$

高阶数值微分公式 $f^{(i)}(x) \approx P^{(i)}(x)$, $2 \leq i \leq n$

二、插值型数值微分公式

仅考虑节点处的导数值,推导数值微分公式.

① 两点公式

$$\begin{cases} f'(x_0) \approx L'_1(x_0) = \frac{y_1 - y_0}{h} \\ f'(x_1) \approx L'_1(x_1) = \frac{y_1 - y_0}{h} \end{cases},$$

截断误差为

$$\begin{cases} R'_1(x_0) = -\frac{h}{2} f''(\xi_0) \\ R'_1(x_1) = \frac{h}{2} f''(\xi_1) \end{cases} \quad \xi_0, \xi_1 \in (a, b).$$

② 三点公式

$$\begin{cases} f'(x_0) \approx \frac{1}{2h}(-3y_0 + 4y_1 - y_2) \\ f'(x_1) \approx \frac{1}{2h}(-y_0 + y_2) \\ f'(x_2) \approx \frac{1}{2h}(y_0 - 4y_1 + 3y_2) \end{cases}$$

截断误差为

$$\begin{cases} R'_2(x_0) = \frac{h^2}{3} f^{(3)}(\xi_0) \\ R'_2(x_1) = -\frac{h^2}{6} f^{(3)}(\xi_1) \\ R'_2(x_2) = \frac{h^2}{3} f^{(3)}(\xi_2) \end{cases} \quad \xi_i \in (a, b), i = 0, 1, 2$$

注 若取

$$f''(x) \approx L_2''(x) = \frac{y_0 - 2y_1 + y_2}{h^2}$$

则可得如下误差估计

$$f''(x_0) - L_2''(x_0) = O(h)$$

$$f''(x_1) - L_2''(x_1) = O(h^2)$$

$$f''(x_2) - L_2''(x_2) = O(h)$$

例 已知函数 $y = f(x)$ 的函数值如下:

x	0.2	0.6	0.8	1.2	1.4	1.8
y	1.221403	1.822119	2.225541	3.320117	4.055200	6.049647

试分别取 $h = 0.4$ 和 0.2 , 用中点公式计算 $y'(1)$ 的近似值.

解 中点公式为 $f'(x_1) \approx \frac{1}{2h}(-y_0 + y_2)$.

(1) $h = 0.4$ 时 $x_0 = 0.6, x_2 = 1.4$, 则

$$y'(1) \approx \frac{4.055200 - 1.822119}{2 \times 0.4} = 2.791351$$

(2) $h = 0.2$ 时 $x_0 = 0.8, x_2 = 1.2$, 则

$$y'(1) \approx \frac{3.320117 - 2.225541}{2 \times 0.2} = 2.736440.$$

5.5 数值实验

命令	含义
polyder	多项式求导
diff	数值差分
trapz	梯形积分法
quad	辛普森积分法

用MATLAB数值求导：将其处理成差分的商。

`diff(x)` 向量 x 的差分；若 x 为矩阵，则按各列作差分。

`diff(x,k)` k 阶差分

`q=polyder(p)` 求得由向量表示的多项式导函数的向量表示 q

```
Command Window
>> x=[1 3 8 7]
x =
     1     3     8     7
>> diff(x), diff(x,2)
ans =
     2     5    -1
ans =
     3    -6
>> |
```

```
Command Window
>> A=[1 3;5 2;6 5;7 7];
>> diff(A)
ans =
     4    -1
     1     3
     1     2
>> |
```

例： 回到实例(美国人口)

1900	1910	1920	1930	1940	1950	1960	1970	1980	1990
76.0	92.0	106.5	123.2	131.7	150.7	179.3	204.0	226.5	251.4

解： 设 t 时刻的人口为 $x(t)$, 人口的增长率为 $r(t)$

则
$$r(t) = \frac{dx/dt}{x(t)}$$

求增长率必须先求导数

先用精度较高的分段五点公式求出节点处的导数值

$x'=21.508$	13.458	16.158	11.908	12.267	25.000
27.633	23.075	22.692	28.358		

$r=dx/dt/x =0.0283$	0.0146	0.0152	0.0097	0.0093	0.0166
0.0154	0.0113	0.0100	0.0113		

将节点处的增长率作 三次样条插值

年份	增长率
1900	0.0283
1901	0.0255
1902	0.0230
1935	0.0082
1936	0.0081
1937	0.0083
1953	0.0172
1954	0.0172
1979	0.0100
1980	0.0100
1981	0.0109
1989	0.0111
1990	0.0113

Renkou.m

Renkou1.m

daoshu.m

