一，绪论

1.1 数学模型

用数学语言来表达物理系统或过程本质特征的公式或方程。

因变量=f(自变量，参数，强制因素)

1. 因变量：用来刻画系统行为与状态的特征量。
2. 自变量：通常为维度，如时间空间，系统的行为由自变量确定
3. 参数：反映系统的性质或组成
4. 强制函数：外部对系统施加的影响

数值计算方法是通过计算机求工程问题满足精度要求的近似解的学科。

1.2 误差

1.2.1 定义

舍入误差：由于计算机只能表示有限位数的量引起的

截断误差：由于数值方法可能运用近似方法表示准确数值运算或数量引起的

有效数字

准确度：计算机或测量值与真值的接近程度

精度/精确度：各计算量或测量值相互之间的集中程度

误差：源于用近似方法表示准确的数学运算和准确的数量

- 用近似方法表示准确数学过程会出现截断误差
- 当用有限个有效数字表示准确的数时会出现舍入误差

绝对误差：真值-近似值 $E = x - x^*$ 可正可负，一般准确值很难给出，用真值的最优估计值代替，或者给出误差的上界$\delta(x^*)$，定义为误差限

相对误差：$\varepsilon_r = \frac{E}{x} \times 100\%$ 当x为0时相对误差无意义，准确值往往未知，常用x^*代替x

相对误差限：$\frac{\delta(x^*)}{|x^*|}$

- 定义：设x^*是x的一个近似数，表示为 $x^* = \pm 10^k \times 0.a_1a_2\cdots a_n$ ，
  其中每个a_i（i=1，2，…，n）均为0，1，2，…，9中的一个数字，且$a_1 \neq 0$，
  如果

$$\left|x - x^*\right| \leq \frac{1}{2} \times 10^{k-n}$$

则称x^*近似x有n位有效数字。

近似数的有效数字越多，相对误差限就越小，反之亦然。

- 如果x^*具有n位有效数字，则其相对误差限为

$$\delta_r(x^*) \leq \frac{1}{2a_1} \times 10^{-(n-1)}$$

1.2.2 有效数字与误差

取有效数字时采取四舍五入，这样其误差不超过近似数末位数的半个单位，即
$|\pi - 3.1416| \leq \frac{1}{2} \times 10^{-4}$

同理由此定义 x^*是x近似n位有效数字

迭代方法的误差估计：

当前迭代结果的误差：$\varepsilon_a = \frac{\text{当前近似值}-\text{上一个近似值}}{\text{当前近似值}} \times 100\%$

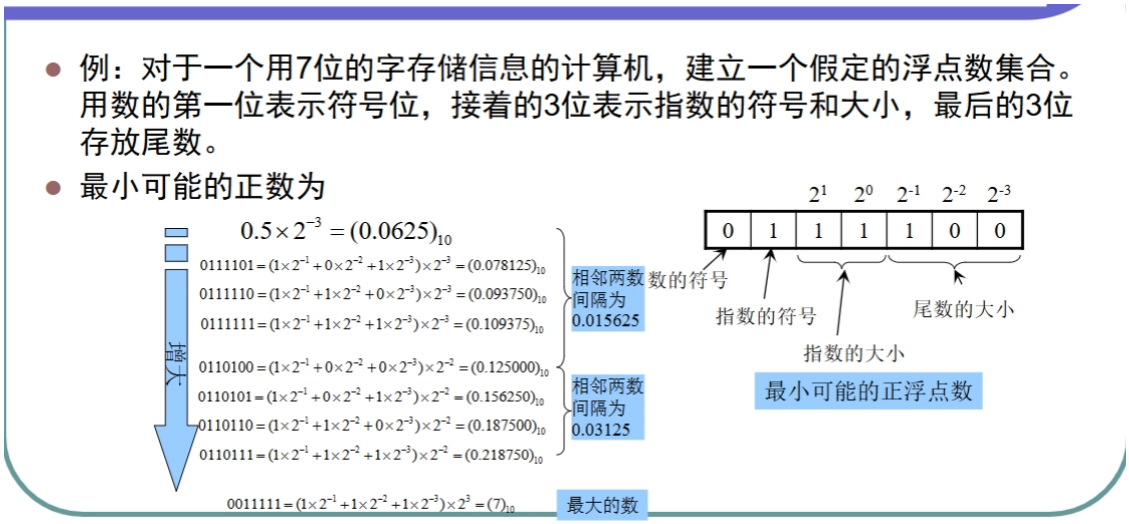
通常不关注符号，采用绝对值与预设好的容限ε_s对比，$|\varepsilon_a| < \varepsilon_s$

如果$\varepsilon_s = (0.5 \times 10^{2-n})\%$，一般至少可以保证至少有n位有效数字是正确的

1.2.3 数的计算机表示

浮点表示：$r = m \times b^e$

- m为尾数，存储了有限的有效数字，会引入舍入误差
- b为基数：是几进制的意思
- e为指数



仅能接受有限范围的数，超出会溢出，不能接受非常小的数，在0与第一个正数之间有一个洞

在可接受范围内只能表示有限个数的数，无理数以及不与该集合中的数对应的有理数均不能准确表示，即量化误差；有效数字足够多时，总的舍入误差可以忽略不计。

数之间的间隔随着数大小增大而增大，量化误差与数的大小成比例。相对量化误差：舍去的情况 $\frac{|\Delta x|}{x} \leq \varepsilon$ 舍入的情况$\frac{|\Delta x|}{x} \leq \frac{\varepsilon}{2}$

其中$\varepsilon = b^{1-t}$ 为机器精度，t为尾数的有效数字个数

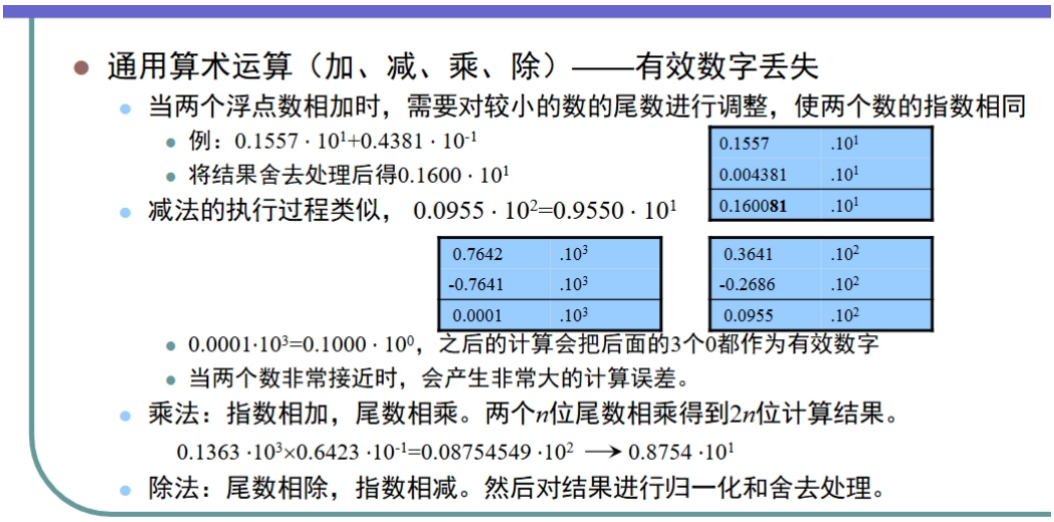
在迭代过程中，当对一个量的收敛性或迭代过程是否中止，测试它们的差是否小于某个可接受的容限。可用机器精度作为停止或收敛准则。

1.2.4 计算机中的数学运算

加减乘除：有效数字丢失

- 加减法：较小数的尾数调整，使两个数指数相等
- 乘法：指数相加，尾数相乘，两个n位尾数得到2n位尾数

- 除法：指数相除，尾数相减，然后再进行归一化和舍入处理。



大规模计算：舍入误差累积

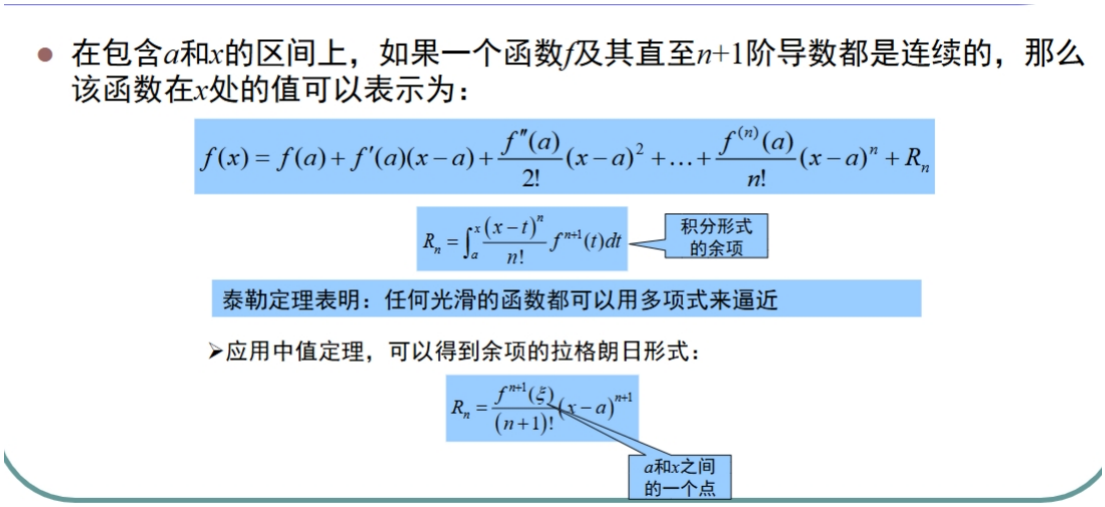
大数加小数：调整较小数，使其指数与大数匹配，再舍去处理

减性抵消：两个几乎相等的浮点数相减时引起的舍入误差

拖尾效应：求和时如果某一项的值大于和值本身，就会发生拖尾效应。

1.2.5 截断误差与泰勒级数

截断误差：运用近似方法表示准确数值运算或数量而引起的误差



- 零阶近似：$f(x_{i+1}) = f(x_i)$
- 一阶近似：$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i)$
- 完整的泰勒展开：

完整的泰勒展开

$$f(x_{i+1})=f(x_i)+f'(x_i)(x_{i+1}-x_i)+\frac{f''}{2!}(x_{i+1}-x_i)^2+\ldots+\frac{f^{(n)}}{n!}(x_{i+1}-x_i)^n+R_n$$

存在一个值可以给出准确的误差表示

$$R_n=\frac{f^{n+1}(\xi)}{(n+1)!}(x_{i+1}-x_i)^{n+1}$$

定义步长 $h=x_{i+1}-x_i$

$$f(x_{i+1})=f(x_i)+f'(x_i)h+\frac{f''}{2!}h^2+\ldots+\frac{f^{(n)}}{n!}h^n+R_n$$

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

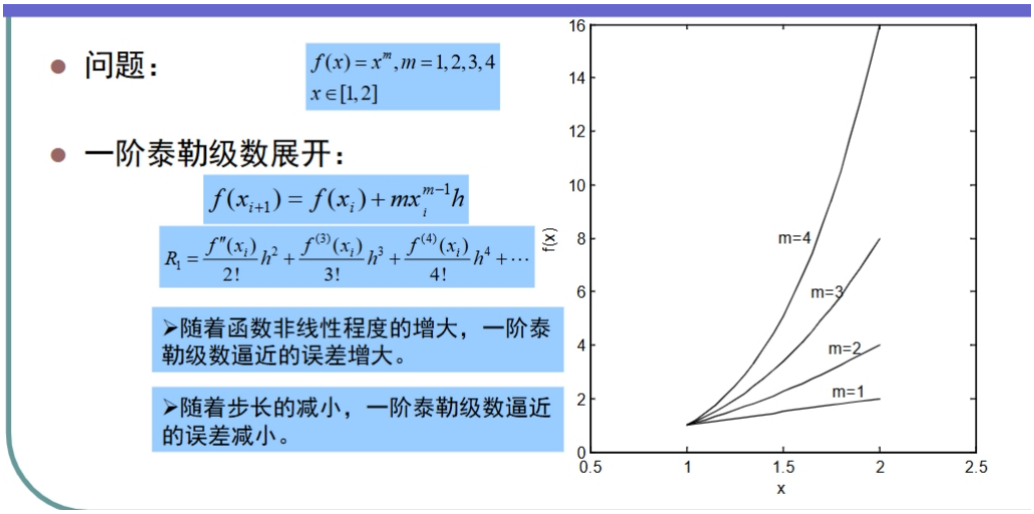
通常，n阶多项式的n阶泰勒级数展开得到的结果是准确的。对于其他连续可微函数，如指数函数和正弦函数，有限级数项是不可能得到准确结果的。每增加一项将使近似结果得到一定的改进，但改进程度不显著。

只需少数项便可获得一个足够精确的估计值。

余项$R_n = \frac{f^{n+1}(\xi)}{(n+1)!}h^{n+1}$ $R_n = O(h^{n+1})$，即误差与步长的n+1次方成正比。

如果h足够小，前几阶占估计值的绝大部分，只需前几阶误差便可极小。

非线性与步长对泰勒级数逼近的影响：



1.2.6 误差的传播与估计

单变量函数

设函数$f(x)$，x* 是x的近似值，估计x* 与x的差异对函数值的影响：$\Delta f(x^*) = |f(x^*) - f(x)|$，用泰勒级数逼近，可得

$\Delta f(x^*) = |f^{'}(x^*)||x - x^*|$

多变量函数

$\Delta f(u^*, v^*) = |\frac{\partial f}{\partial u}|\Delta u^* + |\frac{\partial f}{\partial v}|\Delta v^*$

$\Delta f(x_1^*, x_2^*, \ldots, x_n^*) = |\frac{\partial f}{\partial x_1}|\Delta x_1^* + |\frac{\partial f}{\partial x_1}|\Delta x_1^* + \ldots + |\frac{\partial f}{\partial x_n}|\Delta x_n^*$

一般运算

运算		估计误差
加法	$\Delta(\tilde{u}+\tilde{v})$	$\Delta\tilde{u}+\Delta\tilde{v}$
减法	$\Delta(\tilde{u}-\tilde{v})$	$\Delta\tilde{u}+\Delta\tilde{v}$
乘法	$\Delta(\tilde{u}\times\tilde{v})$	$\lvert\tilde{u}\rvert\Delta\tilde{v}+\lvert\tilde{v}\rvert\Delta\tilde{u}$
除法	$\Delta\left(\frac{\tilde{u}}{\tilde{v}}\right)$	$\frac{\lvert\tilde{u}\rvert\Delta\tilde{v}+\lvert\tilde{v}\rvert\Delta\tilde{u}}{\lvert\tilde{v}\rvert^2}$

函数导数的绝对值很大时，数据误差在运算中传播后，可能会导致结果的很大误差。

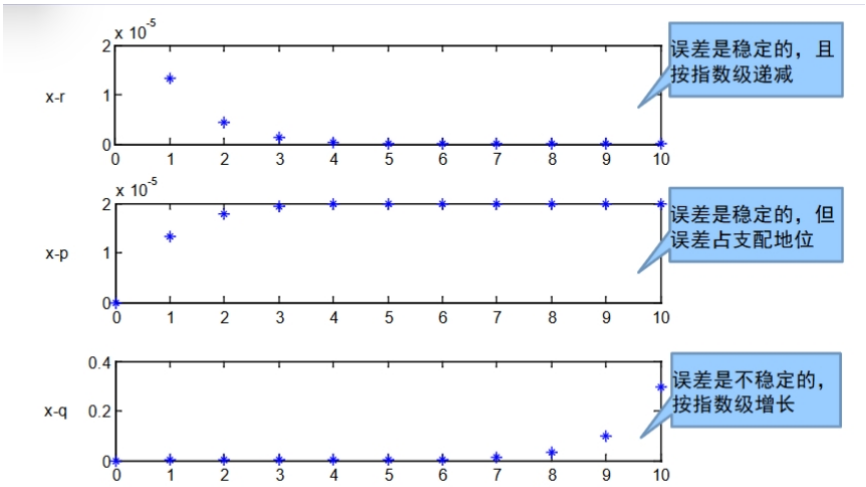
在无限迭代的序列中误差选用不同的算法误差的累积效果不一样。

- 例：用无限精度算法结合下列3个方案可递归生成序列 $\{1/3^n\}_{n=0}^{\infty}$ 中的各项值。

(1) $r_0 = 1, \quad r_n = \frac{1}{3} r_{n-1}, n = 1, 2, \cdots$

(2) $p_0 = 1, p_1 = \frac{1}{3}, p_n = \frac{4}{3} p_{n-1} - \frac{1}{3} p_{n-2}, n = 2, 3, \cdots$

(3) $q_0 = 1, q_1 = \frac{1}{3}, q_n = \frac{10}{3} q_{n-1} - q_{n-2}, n = 2, 3, \cdots$



1.2.7 稳定性与条件性

一个算法如果原始数据有误差，而计算过程舍入误差不增长，则称此算法是数值稳定的，否则，若误差增长，则称算法是不稳定的。（输入值的不确定性通过数值方法在总体上是放大的）

条件数定义为相对误差之比：

f(x)相对误差为 $\frac{f(x)-f(x*)}{f(x*)} = \frac{f'(x*)(x-x*)}{f(x*)}$，x相对误差为 $\frac{x-x*}{x*}$

条件数为: $\left| \frac{x* f'(x*)}{f(x*)} \right|$

条件数体现了x的不确定性被x)放大程度。如果条件数等于1，表示函数的相对误差等于x的相对误差，如果条件数大于1，表示相对误差被放大了，而条件数小于1，表示相对误差减小了。

条件数非常大(大于等于10)的函数称为病态函数。

- 例：问题条件数

$f(x) = \sqrt{x^2+1} - x \qquad x = 100 \Rightarrow f(x) = 0.5 \times 10^{-2}$

$f'(x) = \frac{x}{\sqrt{x^2+1}} - 1 \Rightarrow K_P \approx 1$ (当$x>>1$时)

- 算法A：4位有效数字

$\tilde{x} = 0.1 \times 10^3 \Rightarrow f(\tilde{x}) = \sqrt{(0.1000 + 0.00001) \times 10^5} - 0.1 \times 10^3 = 0$

$|\beta| = \left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| = 1 \qquad |\alpha| = \left| \frac{\tilde{x} - x}{x} \right| \le \frac{1}{2} 10^{1-t} \approx \frac{1}{2} 10^{-3}$

$K_A = \frac{|\beta|}{|\alpha|} \approx 2000$

算法条件数，改善方法：提高精度，重写算法

- 算法B：

$f(x) = \frac{1}{\sqrt{x^2+1} + x}$

$f(\tilde{x}) = \frac{1}{0.1 \times 10^3 + 0.1 \times 10^3} = 0.5 \times 10^{-2} \qquad |\beta| \approx 0 \Rightarrow K_A \approx 0 << 1$

为了最小化舍入误差，计算机会增加有效数字个数，缩短步长使得截断误差减小，但是会使计算量增大，舍入误差变大。大多数计算机可以表示足够多的有效数字，舍入误差不会占据主导地位。

避免误差危害：

- 选择数值稳定的计算方法，避开不稳定的算式。
- 注意简化计算步骤及公式，设法减少运算次数;选用运算次数少的算式尤其是乘方幂次要低，以减少舍入误差，同时可节约计算机的机时。
- 合理安排计算顺序，防止大数"淹没"小数。多个数相加时，最好从绝对值最小的数到绝对值最大的数依次相加;多个数相乘时，最好从有效位数最多的数到有效位数最少的数依次相乘
- 避免两相近数相减。
- 避免用绝对值很小的数作为除数。

二，非线性方程求根

设有非线性方程f(x)=0
当f(x)为多项式时，且最高次为n，称f(x)=0为n次代数方程，如果f(x)包含指数函数或三角函数等特殊函数时，称f(x)=0为超越方程
如果f(x)可拆为$f(x)=(x-x^*)^m g(x)$，称x*为m重根，一个就是单根

两个问题：

- 预先给出根的一个初略位置（根的分离），然后根据这个位置向真实根逼近（近似根的精确化）
- 求解多项式的所有实数根和复数根。专为多项式设计，系统化地求解多项式的所有根，而不仅仅是逼近的实数根

有根区间的情况：

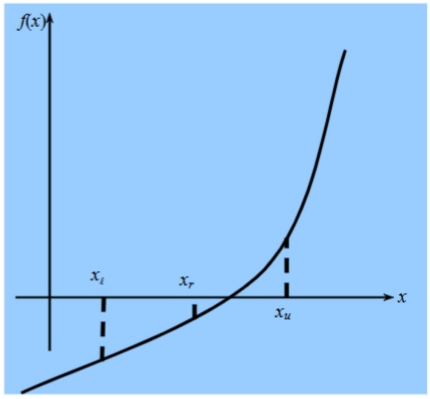
- 两端点同号，中间无根
- 两端点同号，中间偶数根
- 两端点异号，中间奇数根
- 特殊情况：与x轴相切，不连续函数。

2.1 二分法

2.1.1 步骤

首先确定有根区间，将区间二等分，通过判断f(x)的符号，逐步将有根区间缩小，直至有根区间足够地小，便可求出满足精度要求的近似根。

- 步骤
  1. 猜测初始下界x_l和上界x_u，使$f(x_l)\,f(x_u)<0$；
  2. 计算中间点
     $$x_r=\frac{x_l+x_u}{2}$$
  3. 根据情况确定根所在的区间
     a) 如果$f(x_l)\,f(x_r)<0$，则根落在左边子区间，取$x_u=x_r$，返回步骤2；
     b) 如果$f(x_l)\,f(x_r)>0$，则根落在右边子区间，取$x_l=x_r$，返回步骤2；
     c) 如果$f(x_l)\,f(x_r)=0$，则x_r根为所要求的根，算法终止。
  4. 终止条件
     - 新的估计值 $x_r^{new}=\frac{x_l+x_u}{2}$
     - 近似相对误差 $\varepsilon_a=\left|\frac{x_r^{new}-x_r^{old}}{x_r^{new}}\right|\times 100\%$
     - 当ε_a小于既定的终止条件ε_s或迭代次数达到一个上界时，计算终止。

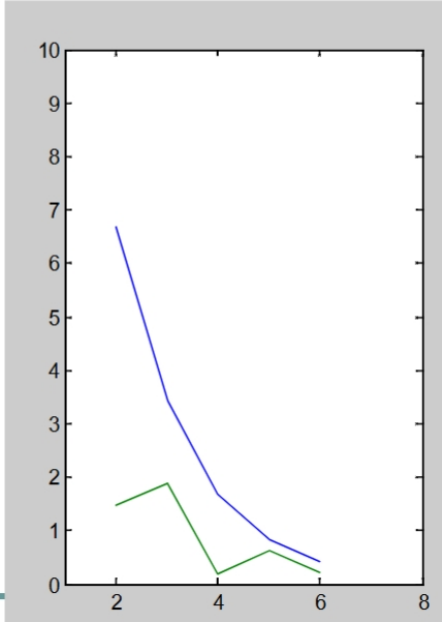


025/2/23

2.1.2 误差估计

迭代次数	x_l	x_u	x_r	ε_a	ε_t
1	12	16	14		
2	14	16	15	6.667	1.487
3	14	15	14.5	3.448	1.896
4	14.5	15	14. 75	1.695	0.204
5	14.75	15	14.875	0.840	0.641
6	14.75	14.875	14.8125	0.422	0.219

- 不是每次迭代后真实误差都会减小。
- 近似误差总是大于真实误差，当ε_a小于ε_s时，真实误差一定也小于ε_s，所以根可以达到要求的精度，算法可以终止。



近似误差总会大于真实误差，可认为近似误差是真实误差的一个精确上界，近似误差满足要求真实的一定也满足。

2.1.3 迭代次数

设初始区间长度L_0，最终期望绝对误差$E_{a,d}$

迭代次数为：$n=\frac{log(L_0/E_{a,d})}{log2}$

2.1.4 代码实现

```
function [c,err,yc]=bisect(f,a,b,delta)
%input-f是函数，a、b是左右边界值，delta是容限
%output-c是零点，yc是f(c)，err是误差
ya=feval(f,a);
yb=feval(f,b);
if(ya*yb>0)
    break
```

```
end
max1=1+round((log(b-a)-log(delta))/log(2));
for k=1:max1
    c=(a+b)/2;
    yc=feval(f,c);
    if yc==0
        a=c;
        b=c;
    elseif yb*yc>0
        b=c;
        yb=yc;
    else
        a=c;
        ya=yc;
    end
    if b-a<delta
        break;
    end
end
c=(a+b)/2;
err=abs(b-a)/2;
yc=feval(f,c);
```

2.1.5 评价

优点

- 简单，能找出根
- 迭代次数能算

缺点

- 慢，要求上下界已知，对与x轴相切的处理不了
- 多根无法处理
- 对异号无根的处理不了。

2.2 试位法

2.2. 1 步骤

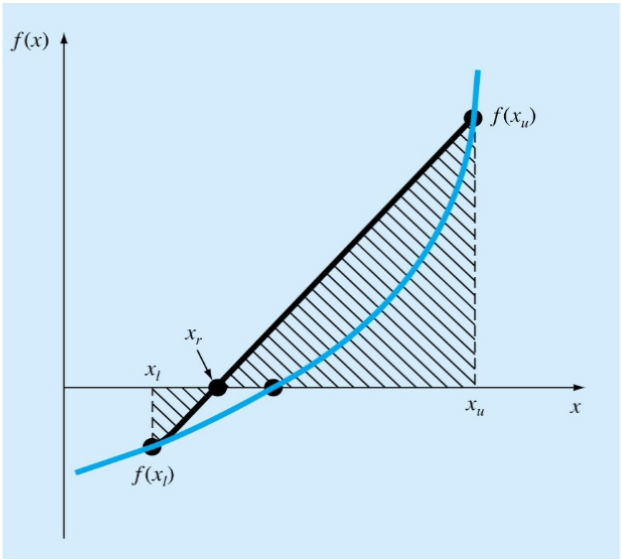
- 通过一条直线连接$[x_l, f(x_l)]$和$[x_u, f(x_u)]$，直线与x轴的交点作为新的根的估计值。

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u}$$

$$x_r = \frac{x_l f_u - x_u f_l}{f_u - f_l}$$

$$x_r = x_u - \frac{f_u(x_l - x_u)}{f_l - f_u}$$

少一次乘法计算



```
function [c,err,yc]=regular(f,a,b,delta,epsilon,max1)
%input-f是函数，a、b是左右边界点，delta是相邻两次迭代值之间误差的容限，epsilon是当前点的函数值与0之间差的容限，max1是最 %  大迭代次数
%output-c是零点，yc是f(c)，err是误差
ya=feval(f,a);
yb=feval(f,b);
if ya*yb>0
    break;
end
for k=1:max1
    dx=yb*(b-a)/(yb-ya);
    c=b-dx;
    ac=c-a;
    yc=feval(f,c);
    if yc==0
        break;
    elseif yb*yc>0
        b=c;
        yb=yc;
    else
        a=c;
        ya=yc;
    end
    dx=min(abs(dx),ac);
    if abs(dx)<delta
        break;
    end
    if abs(yc)<epsilon
        break;
    end
end
c;
err=abs(b-a);
yc=feval(f,c);
```

试位法比二分法误差减小的快。

2.2. 2 缺陷

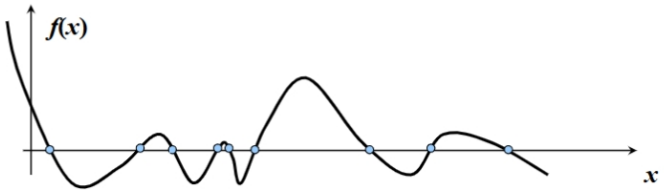
试位法收敛慢
近似误差小于真实误差：除了检查迭代近似误差外，把根的估计值代入原方程中，检验是否接近0
一个划界点保持不动可能导致很差的收敛性。

2.2. 3 修正

检测一个边界值是否固定不变，用一个计数器，如果某个边界值两次迭代不变，将此边界点的函数值变为原来的一半。

2.2. 4 增量搜索和确定初始猜值

- 函数作图
- 增量搜索
  - 从感兴趣区域的一端开始，小增量地穿过区域并计算函数值，如果有函数值改变符号时（$f(x_k)$ $f(x_k+\triangle x)<0$ ），可以确定由一个根落在这个小增量区间内。
  - 小增量区间的两个端点可以作为初始猜值
  - ***增量长度的确定：太小费时，太大丢失彼此相差近的根。***
  - ***计算区间端点处的一阶导数值，如果有符号改变，则区间中存在最大或最小值，减小区间继续搜索***



2.3 不动点迭代法

2.3.1 定义

重组方程：把$f(x)=0$转化为$x=g(x)$,然后$x_k=g(x_{k-1})$进行迭代，其中x0已知。

相对误差：$\varepsilon_a=|\frac{x_{i+1}-x_i}{x_{i+1}}|\times 100\%$

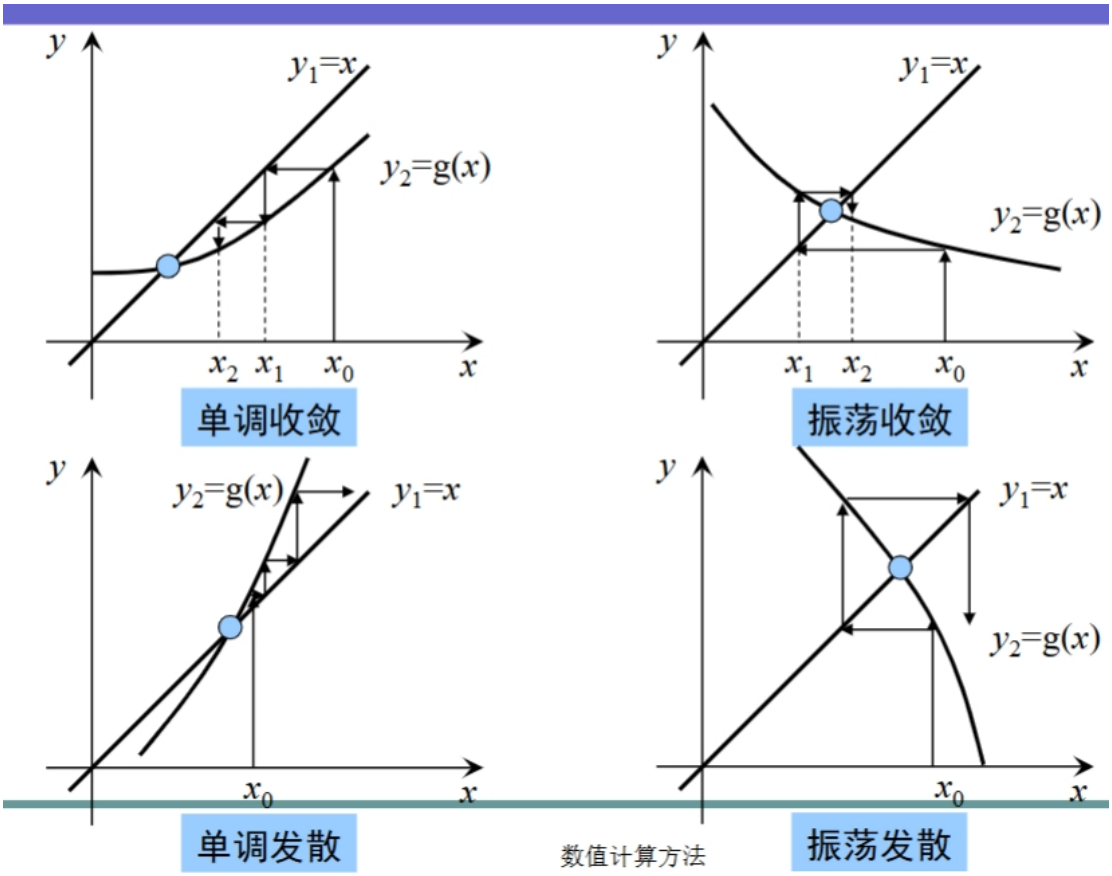
不动点：函数g(x)的一个不动点指一个实数P满足$P=g(P)$，从图形上看是$y=g(x),y=x$的交点

定理：设g(x)是一个连续函数，且$\{P\}_{n=0}^{\infty}$是不动点迭代生成的序列，如果当n->无穷时序列收敛于P，那么P就是g(x)的一个不动点。

设函数$g\in C[a，b]$，如果对于所有$x\in[a，b]$，映射$y=g(x)$的范围满足$y\in[a，b]$，则函数g在$[a，b]$内有一个不动点。此外，设$g'(x)$定义在$(a，b)$内，且对于所有$x\in(a，b)$，存在正常数$K<1$，使得$|g'(x)|\leq K<1$，则函数g在$[a，b]$内有唯一的不动点P。

转化为迭代函数时，可能会不收敛，就算收敛，收敛情况也不一样，注意选择。

四种迭代情况：均可在图中看出迭代的过程。



迭代函数的选择：不动点定理

定理（不动点定理）

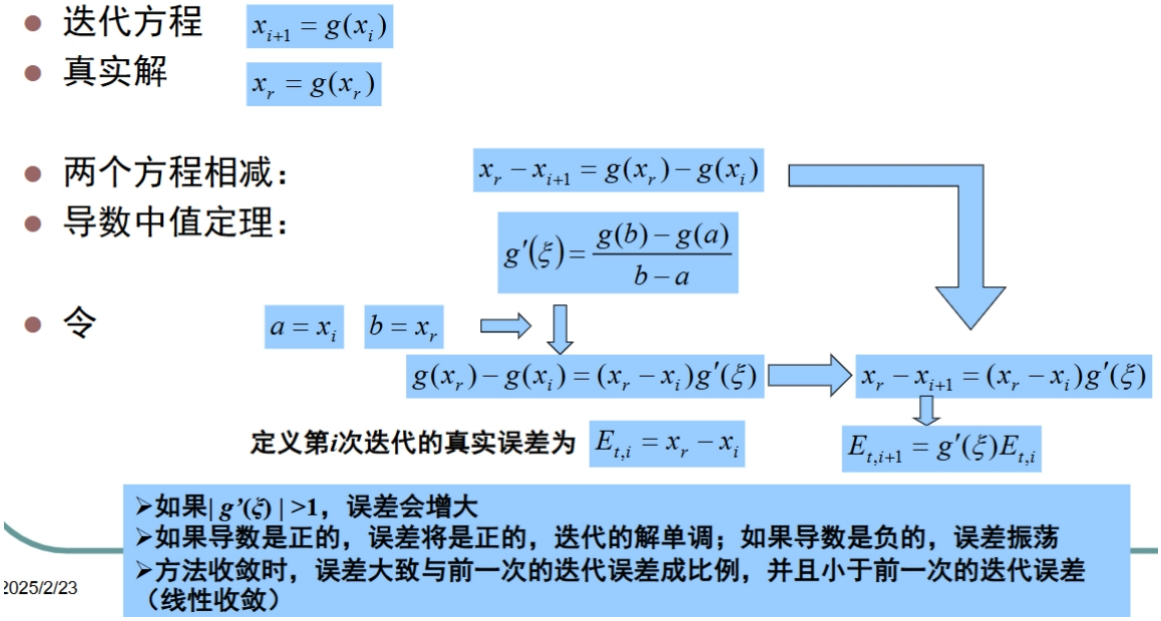
- 设有(i) g，$g' \in C[a, b]$， (ii) K是一个正常数， (iii) $p_0 \in (a, b)$， (iv)对于所有$x \in [a, b]$，有$g(x) \in [a, b]$。
  - 如果对于所有$x \in [a, b]$，有$|g'(x)| \leq K<1$，则迭代$p_n=g(p_{n-1})$将收敛到唯一的不动点$P \in [a, b]$。在这种情况下，P称为吸引(attractive)不动点。
  - 如果对于所有$x \in [a, b]$，有$|g'(x)| >1$，则迭代$p_n=g(p_{n-1})$将不会收敛到P。在这种情况下，P称为排斥(repelling)不动点，而且迭代显示出局部发散性。

2.3.2 程序实现

```
function [k,p,err,P]=fixpt(g,p0,tol,max1)
%input-g是迭代函数，p0是不动点初始值，tol是容限，max1是最大迭代次数
%output-k是当前迭代的次数，err是误差，p是不动点最终结果，P是p迭代的序列
P(1)=p0;
for k=2:max1
    P(k)=feval(g,P(k-1));
    err=abs(P(k)-P(k-1));
    relerr=err/abs(P(k)+eps);
    p=P(k);
    if (err<tol)||(relerr<tol),break;end
end
if k==max1
    disp("最大迭代次数已经达到")
end
```

```
end
```

由导数中值定理可以论证不动点迭代法的合理性。



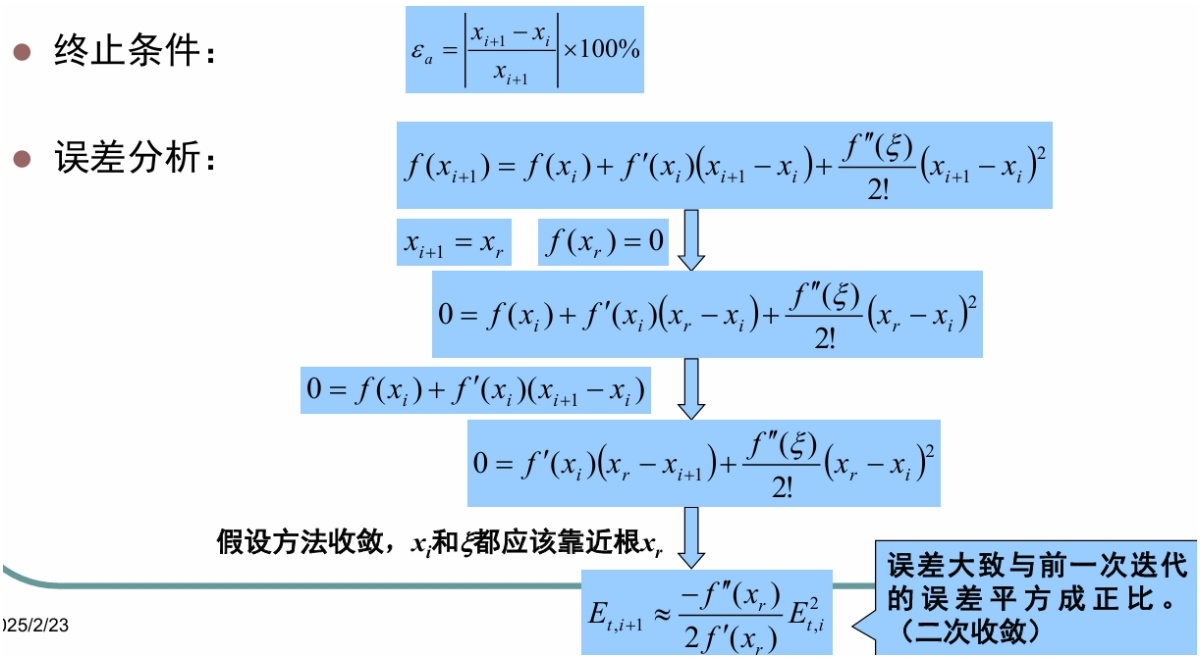
2.4 Newton-Raphson法

2.4.1 定义

已知$f(x_{i+1})\approx f(x_i)+f'(x_i)(x_{i+1}+x_i)$
假设x_{i+1}就是f(x)的零点，函数值为0，那么则有$0\approx f(x_i)+f'(x_i)(x_{i+1}+x_i)$
即$x_{i+1}=x_i-\frac{f(x_i)}{f'(x_i)}$，就是寻找$g(x)=x-\frac{f(x)}{f'(x)}$的不动点

N-R方法比不动点迭代法收敛的快得多，每次迭代误差减小比不动点方法快得多。

2.4.2 误差分析



2.4.3 收敛阶

定义

- 设序列$\{x_n\}_{n=0}^{\infty}$收敛到x_r，并令$E_n=x_r-x_n$，$n\geq 0$。如果存在两个常量$A\neq 0$和$R>0$，并且

$$\lim_{n\to\infty}\frac{|x_r-x_{n+1}|}{|x_r-x_n|^R}=\lim_{n\to\infty}\frac{|E_{n+1}|}{|E_n|^R}=A$$

- 则称该序列以收敛阶R收敛到x_r，A称为渐近误差常数。
- $R=1$，称为线性收敛
- $R=2$，称为二次收敛
- 一些序列的收敛阶不是整数。

2.4.4 收敛速度

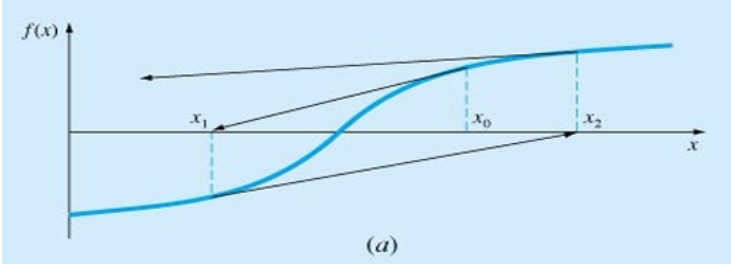
设N-R迭代产生的序列{xn}收敛到函数的根xr，如果xr是单根，则{xn}是二次收敛，而且对于足够大的n来说有

$|E_{n+1}|\approx\frac{|f''(x_r)|}{2|f'(x_r)|}|E_n|^2$，即$A=\frac{|f''(x_r)|}{2|f'(x_r)|}$

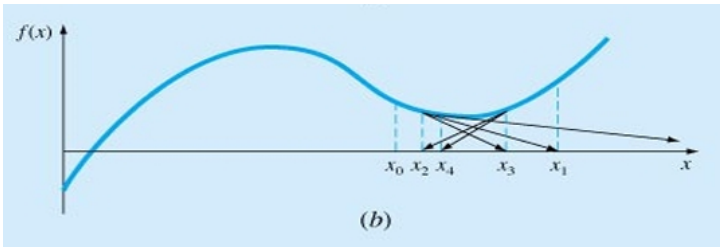
如果xr是M阶多重根，则{xn}是线性(一阶)收敛，而且对于足够大的n有$|E_{n+1}|\approx\frac{M-1}{M}|E_n|$

2.4.5 缺点

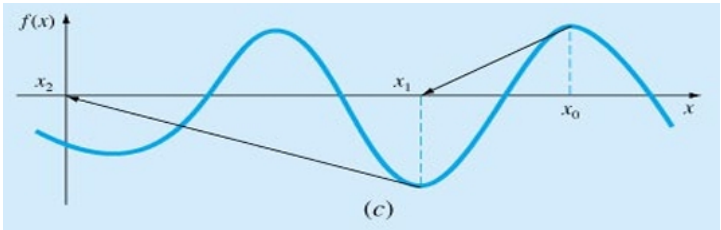
- 在根附近出现拐点，$f''(x)=0$， 迭代逐渐远离根；



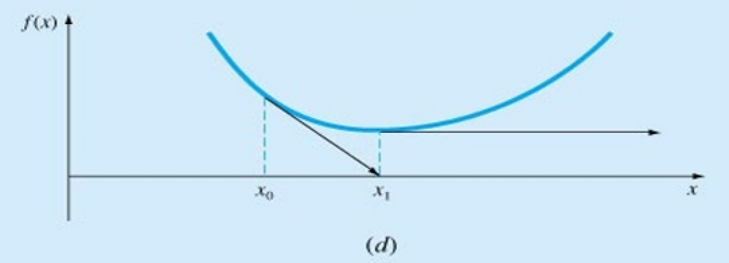
- 在局部极值点附近振荡



- 跳过了几个根



- $f'(x)=0$，被零除，解沿水平方 向永远不会和x轴相交；



收敛性依赖于函数的性质和初始猜测的准确度

2.4.6 代码实现

- 将最终的估计根代入原始函数中检查f(x)是否为0，防止由于收敛很慢或振荡 收敛导致很小的ε_a但根却离真实根仍然很远。
- 确定迭代上限，避免因振荡、慢收敛或发散的情况导致无限循环。
- 判断$f^{'}(x)=0$的概率并通知用户

```
function [p0,err,k,y]=newton(f,df,p0,delta,epsilon,max1)
%input-f是函数，df是函数一阶导，p0是初始值，delta迭代两次差的容限，epsilon是迭代当前结果函数值与0之间差的容限，max1最大
%迭代次数
%output-p0是最后结果结果，k是当前迭代次数，y是f(p0)函数值
for k=1:max1
    p1=p0-feval(f,p0)/feval(df,p0);
    err=abs(p1-p0);
    relerr=2*err/(abs(p1)+delta);
    p0=p1;
    y=feval(f,p0);
    if(err<delta)||(relerr<delta)||(abs(y)<epsilon),break;end
end
end
```

2.5 割线法（正割法，Secant Method）

2.5.1 定义

$$\frac{1}{f'(x_i)} \cong \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \qquad i = 1,2,3,\ldots$$

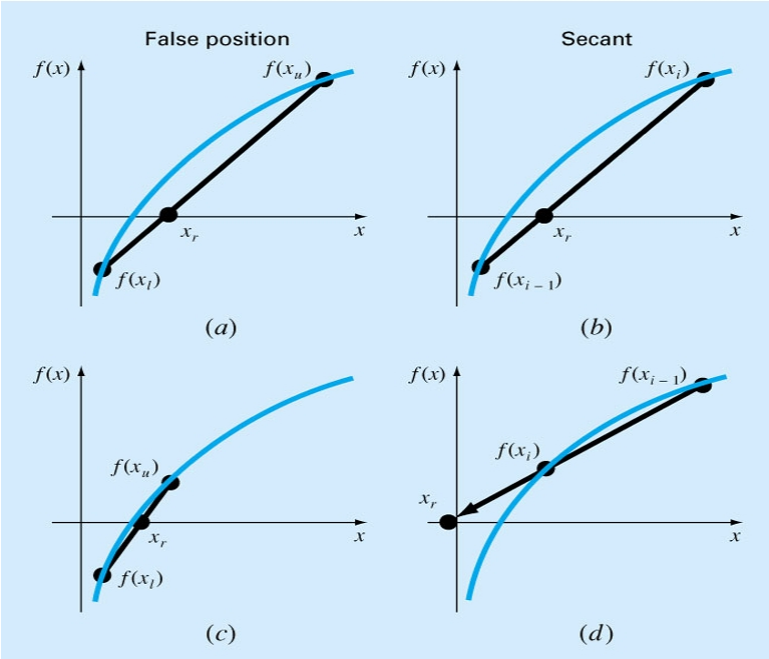
类似于N-R方法，用差商表示斜率。
类似于试位法：用两个估计值来计算函数斜率的近似，并投射到x轴上获得一个新的估计值。
收敛阶：1.618

2.5.2 程序实现

```
function [p1,err,k,y]=secant(f,p0,p1,delta,epsilon,max1)
%input-f是函数，p0是初始值，p1也是估计值，delta是相邻两项的容限，epsilon是y的容限，max1是最大迭代次数
%output-p1是估计值，err是误差，y是函数值
for k=1:max1
    p2=p1-feval(f,p1)*(p1-p0)/(feval(f,p1)-feval(f,p0));
    err=abs(p2-p1);
    relerr=2*err/(abs(p2)+delta);
    p0=p1;
    p1=p2;
    y=feval(p1);
    if (err<delta)||(relerr<delta)||(abs(y)<epsilon),break;end
end
end
```

2.5.3 与试位法区别

试位法一般收敛，割线法可能发散，如果割线法收敛，比试位法快。



2.5.4 改进

估计导数：通过独立变量的小量扰动来估计

$$f'(x_i) \cong \frac{f(x_i+\delta x_i)-f(x_i)}{\delta x_i} \qquad x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i+\delta x_i)-f(x_i)}$$

δ是一个待定变量，需要人为确定，算法不会自己选择合适的δ，如果δ太小可能会因为分母的减性抵消引起的舍入误差淹没。
如果选择正确，当导数难以计算或者两个初始值不易获得时，是一种好的算法。

求解函数$f(x)=e^{-x}-x$的根：取$\delta=0.01$

不易获取时，是一的替代算法。

i	x_{i-1}	$x_{i-1}+\delta x_{i-1}$	$f(x_{i-1})$	$f(x_{i-1}+\delta x_{i-1})$	x_i	ε_i
1	1	1.01	-0.63212	-0.64578	0.537263	5.3
2	0.537263	0.542635	0.047083	0.038579	0.56701	0.0236
3	0.56701	0.572680	0.000209	-0.00867	0.567143	2.365×10^{-5}

2.5.5 重根问题

定义一个新函数$u(x)=\frac{f(x)}{f'(x)}$，这个函数与原函数有相同的根

改进的Newton-Raphson方法

$$x_{i+1}=x_i-\frac{u(x_i)}{u'(x_i)} \qquad u'(x)=\frac{f'(x)f'(x)-f(x)f''(x)}{[f'(x)]^2}$$

$$x_{i+1}=x_i-\frac{f(x_i)f'(x_i)}{[f'(x_i)]^2-f(x_i)f''(x_i)}$$

改进的割线法$x_{i+1}=x_i-u(x_i)\frac{x_i-x_{i-1}}{u(x_i)-u(x_{i-1})}$，即把f(x)替换为了u(x)。

2.6 总结

方法	初始估计	收敛速度	稳定性	精度	应用范围	编程难度	备注
直接法	—	—	—	—	受限		
图解法	—	—	—	差	求实数根	—	比数值法时间开销大
二分法	2	慢	通常收敛	好	求实数根	容易	
试位法	2	慢/中等	通常收敛	好	求实数根	容易	
不动点迭代	1	慢	可能发散	好	求一般根	容易	
Newton-Raphson	1	快	可能发散	好	求一般根	容易	需要求导
割线法	2	中等到快	可能发散	好	求一般根	容易	初始估计不用界定根

重点在于迭代公式，误差与终止准则的选择。

2.7 matlab自带的相关函数

涉及多项式求根和方程组求根

函数	描述
fzero	单函数求根

函数	描述
roots	求多项式的根
poly	用已知根构建多项式
polyval	求多项式的值
polyvalm	求带有矩阵变量的多项式的值

三，线性代数方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

矩阵AX=B

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^T$$

$$B = \begin{bmatrix} b_1 & b_2 & \ldots & b_n \end{bmatrix}^T$$

$$\text{增广矩阵} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} & b_n \end{bmatrix}$$

系数矩阵的类型：

- 低阶稠密矩阵
- 大型稀疏矩阵（零元素较多）
- 三对角矩阵（非0元素集中于主对角线以及相邻两对角线上）

3.1 直接法

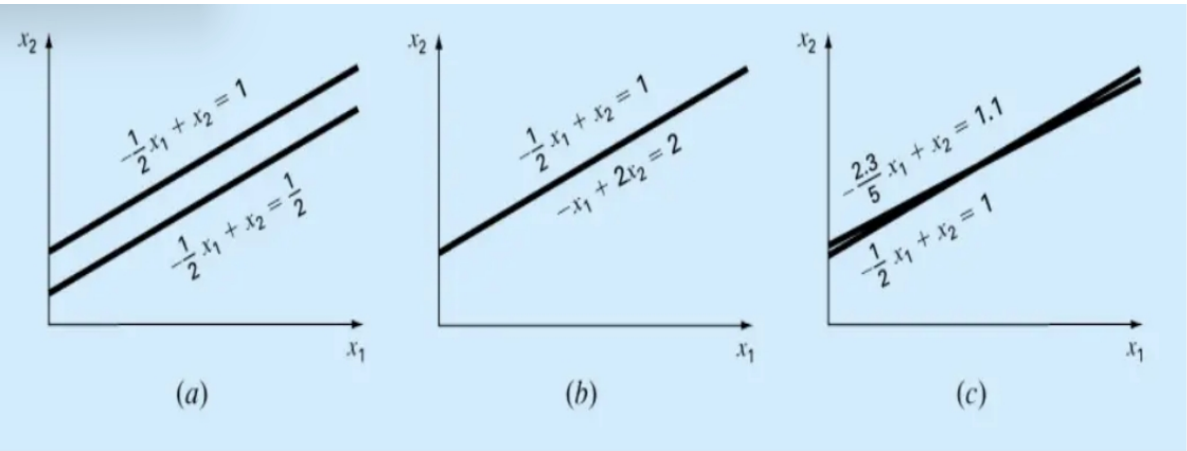
经过有限步算术运算，可求得方程组的精确解的方法（若在计算过程中没有舍入误差）
可预先估算使用机器时间，计算量小，但要占用较多内存，程序复杂。一般说来，适用于方程组的系数矩阵阶数不太高的问题。

3.1.1 非计算机方法

3.1.1.1 图解法

在二维坐标系中画出两条直线，交点即为联立方程组的解

三个方程组成的联立方程组：每个方程是三维空间中的一个平面，三个平面的交点表示方程组的根



无解　　无穷多解　　病态情况（斜率接近）

3.1.1.2 克莱姆法则

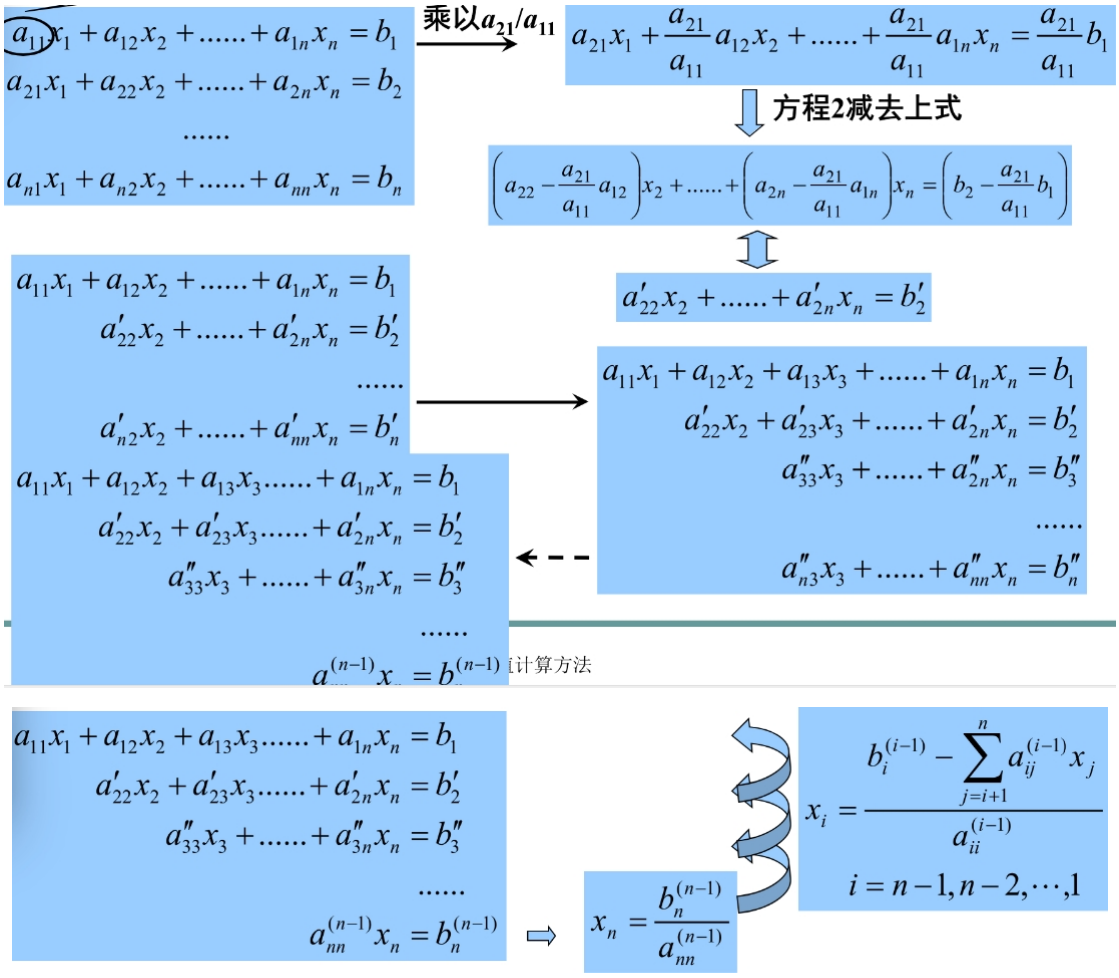
每个未知数通过一个分式来计算
分式的分母是线性代数方程组的行列式
分子是另外一个行列式，该行列式与方程组系数行列式D只有未知数对应的一列不同，不同的列为常数b1，b2，...，bn

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{D}$$

未知数消去法，用已知的根求额外根。

3.1.2 计算机方法

3.1.2.1 原始高斯消去法



原理非常简单，代码实现如下（只适用于n×n方阵）

```
function x=Gauss(a,b)
    [m,n]=size(a);
    for k=1:n-1
        for i=k+1:n
            factor=a(i,k)/a(k,k);
            for j=k+1:n
                a(i,j)=a(i,j)-factor*a(k,j);
            end
            b(i)=b(i)-factor*b(k);
        end
    end
    x(n)=b(n)/a(n,n);
    for i=n-1:-1:1
        sum=b(i);
       for j=i+1:n
           sum=sum-a(i,j)*x(j);
       end
       x(i)=sum/a(i,i);
    end
```

消去过程的时间复杂度：$\frac{2n^3}{3}+O(n^2)$；回代过程的时间复杂度：$n^2+O(n)$；
总的时间复杂度：$\frac{2n^3}{3}+O(n^2)$；

缺陷：消去与回代会出现被0除的情况
舍入误差过大：每一个计算结果都依赖于前一个结果，误差会累积
奇异方程组：两个方程组完全相等时，n个未知数而只有n-1个方程；对大规模方程组，不容易发现这种奇异性；利用奇异方程组的行列式为0进行判断
病态方程组：系数的微小改变会导致解的剧烈变化。

- 标量因子会对行列式产生影响，但对解没有影响
  - 良态方程组

$$\begin{aligned}3x_1+2x_2&=18\\-x_1+2x_2&=2\end{aligned}\qquad D=\begin{vmatrix}3&2\\-1&2\end{vmatrix}=8$$

  - 病态方程组

$$\begin{aligned}x_1+2x_2&=10\\1.1x_1+2x_2&=10.4\end{aligned}\qquad D=-0.2$$

  - 病态方程组×10

$$\begin{aligned}10x_1+20x_2&=100\\11x_1+20x_2&=104\end{aligned}\qquad D=-20$$

- 对方程组进行缩放使得任何一行中的最大系数等于1，然后计算行列式的值

$$\begin{aligned}x_1+0.667x_2&=6\\-0.5x_1+x_2&=1\end{aligned}\quad D=1.333\qquad\qquad \begin{aligned}0.5x_1+x_2&=5\\0.55x_1+x_2&=5.2\end{aligned}\quad D=-0.05$$

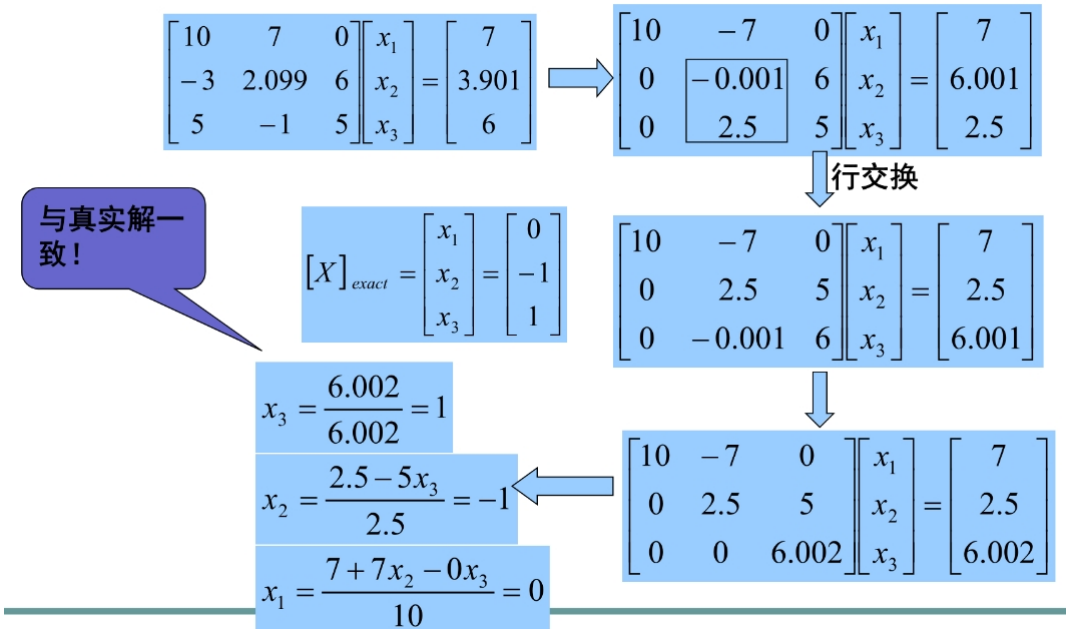
矩阵求逆
稍微改变系数然后求解，如果得到彻底不同的解，则方程组很可能是病态的。

3.1.2.2 解求精技术

3.1.2.2.1 使用拓展精度

3.1.2.2.2 列主元消去法

从第一列中选出绝对值最大的元素，将最大元素对应的行换到第一行，然后进行一次消元。在从第2行到第n行中的第二列找到绝对值最大元素，把对应此行换到第二行；此后都依次这样操作，直到消元完毕；



3.1.2.2.3 全主元消去法

- 从 $a_{ij}(i,j=1,2,...,n)$ 中选择绝对值最大者作为主元，进行消元

$$\tilde{A}_{n-k}=\begin{bmatrix} a_{kk}^{(k-1)} & a_{k,k+1}^{(k-1)} & \cdots & a_{k,n}^{(k-1)} \\ a_{k+1,k}^{(k-1)} & a_{k+1,k+1}^{(k-1)} & \cdots & a_{k+1,n}^{(k-1)} \\ \cdots & \cdots & \cdots & \cdots \\ a_{nk}^{(k-1)} & a_{n,k+1}^{(k-1)} & \cdots & a_{n,n}^{(k-1)} \end{bmatrix}$$

- 第k步，选主元

$$|a_{i_k,j_k}^{(k-1)}|=\max_{k\le i,j\le n}|a_{i,j}^{(k-1)}|$$

- 执行消元过程

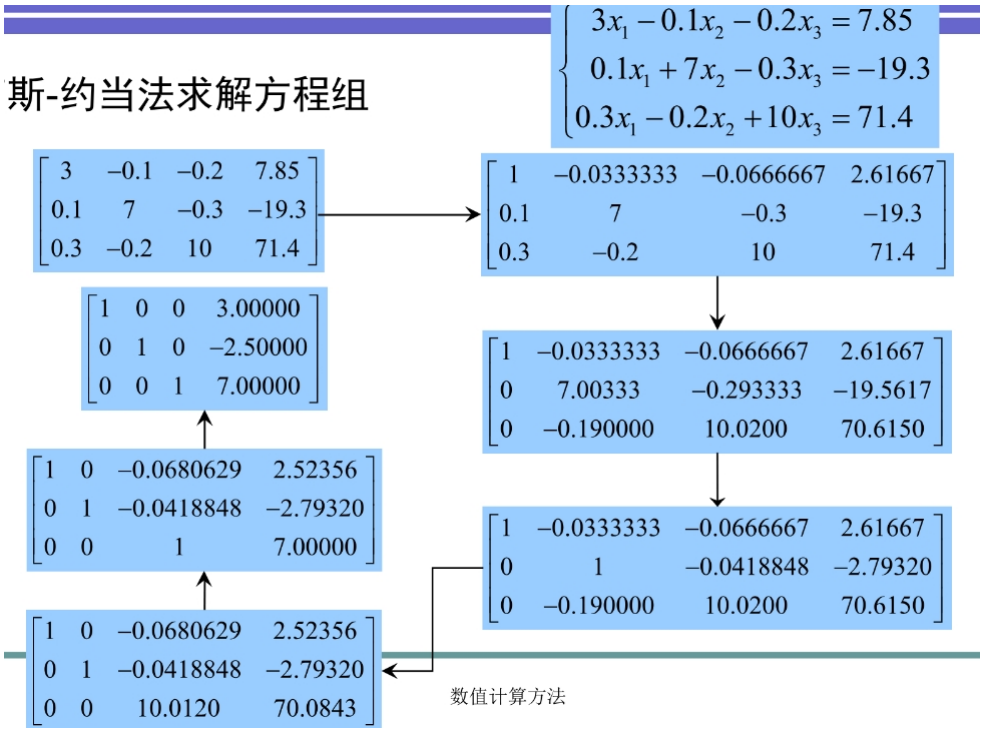
3.1.2.2.4 缩放

- 问题：采用3位有效位精度求解方程组 $\begin{aligned} 2x_1+100{,}000x_2&=100{,}000\\ x_1+x_2&=2 \end{aligned}$ 真实解 (1.00002, 0.99998)
- 解1：$\begin{aligned} 2x_1+100{,}000x_2&=100{,}000\\ -50{,}000x_2&=-50{,}000 \end{aligned}\Rightarrow\begin{aligned} x_2&=1\\ x_1&=0 \end{aligned}$ **x_2是正确的，但是因为舍入误差，x_1误差100%**
- 解2：$\begin{aligned} 0.00002x_1+x_2&=1\\ x_1+x_2&=2 \end{aligned}\Rightarrow\begin{aligned} x_1+x_2&=2\\ 0.00002x_1+x_2&=1 \end{aligned}\Rightarrow\begin{aligned} x_1+x_2&=2\\ x_2&=1.00 \end{aligned}\Rightarrow x_1=x_2=1$
- 解3：$\begin{aligned} x_1+x_2&=2\\ 2x_1+100{,}000x_2&=100{,}000 \end{aligned}\Rightarrow\begin{aligned} x_1+x_2&=2\\ 100{,}000x_2&=100{,}000 \end{aligned}\Rightarrow x_1=x_2=1$
- 通过缩放确定是否交换主元，但方程不需要缩放就可以求得正确解，消去和回代仍使用原系数。

3.1.2.3 高斯-约当法

- 高斯消去法的变形
- 把所有方程的未知数都消去
- 每个方程除以其主元进行标准化
- 最后的结果是一个单位阵
- 只需要消去，不需要回代

步骤：每行依次消，矩阵的对角线标准化为0；先消成上三角矩阵，然后每行消，消成单位矩阵



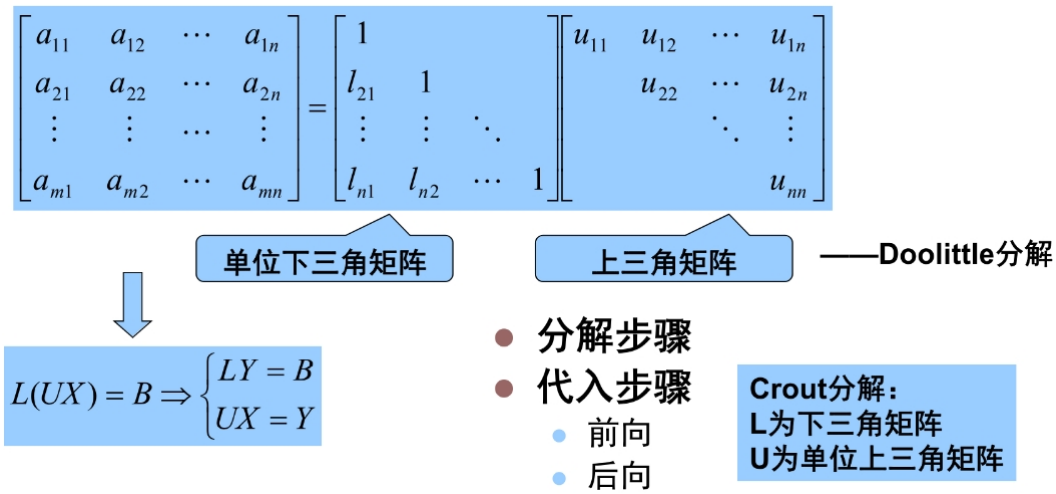
3.1.2.3 高斯消去法评价

- 条件苛刻，数值不稳定
- 全主元消去法工作量偏大，需要比较的元素及行列交换工作较多，算法复杂
- Gauss-Jordan消去法形式上比其他消元法简单，且无回代求解，但计算量大
- 从算法优化的角度考虑， Gauss列主元消去法比较好

3.1.3 LU分解、特殊矩阵

3.1.3.1 三角分解（LU分解）

A=LU



Doolittle分解：

定理：当A的各阶顺序主子式均不为零时，Doolittle分解可以实现并且唯一。

$$A_k = \begin{vmatrix} a_{11} & \dots & a_{1k} \\ \dots & \dots & \dots \\ a_{k1} & \dots & a_{kk} \end{vmatrix} \neq 0 \quad (k = 1,2,\dots n)$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}$$

$$a_{ki} = \sum_{j=1}^{\min(k,i)} l_{kj} u_{ji} \qquad l_{ii} = 1$$

计算顺序：

1，先求U第一行与L第一列：U的第一行$u_{1i} = a_{1i}$；L的第一列$l_{11} = 1, l_{k1} = a_{k1}/u_{11}$
2，求出U前k-1行与L前k-1列后，第k步中计算U的第k行与L的第k列

$$u_{ki} = a_{ki} - \sum_{j=1}^{k-1} l_{kj} u_{ji} \qquad i = k, \dots, n$$

$$l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk})/u_{kk} \qquad i = k+1, \dots, n; k \neq n$$

向前代入求解LY=B

$$y_1 = b_1$$

$$y_k = b_k - \sum_{j=1}^{k-1} l_{kj} y_j \qquad k = 2, \dots, n$$

向后代入求解UX=Y

$$x_n = y_n/u_{nn}$$

$$x_k = (y_k - \sum_{j=k+1}^{n} u_{kj} x_j)/u_{kk}$$

$$k = n-1, \dots, 1$$

要先前向求解Y再反向求解X

求解方程组

$$\begin{cases} 3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 = -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 = 71.4 \end{cases}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix}$$

$$LY = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{bmatrix} \Rightarrow Y = \begin{bmatrix} 7.85 \\ -19.5617 \\ 70.0843 \end{bmatrix}$$

$$UX = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = Y = \begin{bmatrix} 7.85 \\ -19.5617 \\ 70.0843 \end{bmatrix} \Rightarrow X = \begin{bmatrix} 3 \\ -2.5 \\ 7.00003 \end{bmatrix}$$

3.1.3.2 与高斯消元法比较

三角分解的时间复杂度：
分解：$\frac{n^3-n}{3}$；代入n^2

随着n增加，LU分解与高斯消元法计算量相当
直接三角分解法是从矩阵A的元素直接由关系式A=LU确定L和U的元素，不必像Gauss消去法那样计算那些中间结果
高斯消去法求解方程组时，右端项必须提前知道，三角分解则不需要
在实现A=LU分解后，解具有相同系数矩阵的方程组AX=Bj相当方便，每解一个方程组只需求解两个三角形方程组，用n2次乘除法运算即可完成求解。

3.1.4 求逆

- 逆矩阵A^{-1}

$$AA^{-1} = A^{-1}A = I$$

- 以n=3为例
  - A^{-1}各列分别通过 $AX = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $AX = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $AX = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ 计算
  - LU分解方法的优势在于多个右边常数向量的求解
- LU分解计算逆矩阵的计算量为 $\frac{n^3-n}{3} + n \times n^2 = \frac{4n^3-n}{3}$
- 高斯消去计算逆矩阵的计算量（只考虑乘除操作）为 $n\left(\frac{n^3}{3} + \frac{n^2}{2}\right) = \frac{n^4}{3} + \frac{n^3}{2}$

3.1.5 对称正定矩阵的平方根法（Cholesky分解）

- 对称阵：$A^T = A$
- 正定阵：$x^T Ax > 0, \forall x \in R^n, x \neq 0$
- 各阶顺序主子式均大于零

$$A_k = \begin{vmatrix} a_{11} & \dots & a_{1k} \\ \dots & \dots & \dots \\ a_{k1} & \dots & a_{kk} \end{vmatrix} > 0 \quad (k = 1,2,\dots n)$$

由Doolittle分解，A由唯一分解A=LU

LDR分解：A分解为A=LDR，L、R分别为单位下、上三角阵，D为一个对角阵。

对称正定矩阵A有三角分解$A = LDL^T$

假设A是n阶实对称正定矩阵，则必存在非奇异下三角矩阵L，使$A = LL^T$，并且当L的主对角元均为正时，这种分解是唯一的。称$A = LL^T$为矩阵A的Cholesky分解。

3.1.5.1 Cholesky分解过程：

$$A = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix} = LL^T$$

$$\begin{cases} a_{ii} = \sum_{k=1}^{i} l_{ik} l_{ik} \\ a_{ij} = \sum_{k=1}^{j} l_{ik} l_{jk} \quad i > j \end{cases} \quad \begin{cases} l_{jj} = (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{\frac{1}{2}} \\ l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj} \end{cases} \quad i = j+1,\dots,n, j = 1,2,\dots,n$$

$$y_k = (b_k - \sum_{j=1}^{k-1} l_{kj} y_j) / l_{ii} \quad k = 1,\cdots,n \qquad x_k = (y_k - \sum_{j=k+1}^{n} l_{kj} x_j) / l_{kk} \quad k = n,\cdots,1$$

3.1.5.2 优缺点

优点：数值稳定（不需要交换主元），存储量小，计算量小，约需$\frac{x^3}{6}$次乘除法，大约是高斯消去法或Doolittle
分解法的一半
缺点：存在开方运算，可能会出现根号下负数

3.1.6 改进的Cholesky分解

改进的cholesky分解$A=LDL^{\mathrm{T}}$

$$L=\begin{bmatrix}1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \cdots & \cdots & \cdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn-1} & 1\end{bmatrix} \quad D=\begin{bmatrix}d_1 & & & & \\ & d_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & d_{nn}\end{bmatrix}$$

由$A=L(DL^T)$

$$=\begin{bmatrix}1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \cdots & \cdots & \cdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn-1} & 1\end{bmatrix}\begin{bmatrix}d_1 & d_1l_{21} & d_1l_{31} & \cdots & d_1l_{n1} \\ & d_2 & d_2l_{32} & \cdots & d_2l_{n2} \\ & & d_3 & \cdots & d_3l_{n3} \\ \cdots & \cdots & \cdots & \ddots & \\ & & & & d_n\end{bmatrix}$$

可以避免开方运算

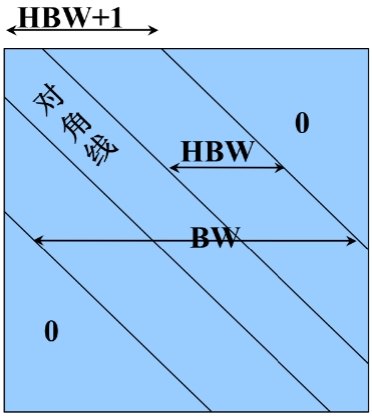
计算中，为了减少计算量，可令

$$c_{ij}=l_{ij}d_j$$

3.1.7 三对角方程组的追赶法 Thomas算法

带状方程组：除了主对角线为中心的一个带状范围内的元素不为零，其他元素都为零；带宽BW，半带宽HBW；如果|i-j|>HBW，aij=0

高斯消去或LU分解求解带状方程组效率低



系数矩阵A的元素满足对角占优条件

$$\begin{bmatrix}b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n\end{bmatrix}\begin{bmatrix}x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n\end{bmatrix}=\begin{bmatrix}f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n\end{bmatrix}$$

$$\begin{cases}|b_1|>|c_1| \\ |b_i|\geq|a_i|+|c_i|,(a_ic_i\neq 0,i=2,\cdots,n-1) \\ |b_n|>|a_n|\end{cases}$$

可以证明，A非奇异，且各阶顺序主子式都不为0

每行对角线上的值最大，而且比左右两者的和都大。

$$A=\begin{bmatrix}\alpha_1 & & & & \\ \gamma_2 & \alpha_2 & & & \\ & \ddots & \ddots & & \\ & & \gamma_{n-1} & \alpha_{n-1} & \\ & & & \gamma_n & \alpha_n\end{bmatrix}\begin{bmatrix}1 & \beta_1 & & & \\ & 1 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & \beta_{n-1} \\ & & & & 1\end{bmatrix}=LU$$

计算公式

（1）分解计算公式A=LU

$$\beta_1 = c_1/b_1$$
$$\beta_i = c_i/(b_i - a_i\beta_{i-1}) \quad i = 2,\ldots,n-1$$

（2）求解Ly=f的递推算式

$$y_1 = f_1/b_1$$
$$y_i = (f_i - a_i y_{i-1})/(b_i - a_i\beta_{i-1}) \quad i = 2,\ldots,n$$

（3）求解Ux=y的递推算式

$$x_n = y_n$$
$$x_i = y_i - \beta_i x_{i+1} \quad i = n-1,\ldots,1$$

对于舍入误差是稳定的，仅需5n−4次乘除法运算。

3.2 迭代法

3.2.1 误差分析和方程组条件数

利用逆矩阵确定方程组是否病态的方法：

- 缩放系数矩阵A，使其每一行的最大元素为1。计算缩放后的逆矩阵，如果A^{-1}中有元素的值大于1几倍，则方程组是病态的。
- 将逆矩阵与原矩阵相乘，检查结果是否接近单位阵。如果不接近单位阵，则方程组是病态的。
- 求逆矩阵的逆矩阵，与原系数矩阵对比。如果不相等，则方程组是病态的。

3.2.2 向量与矩阵的范数

向量的范数

定义：

设对任意向量$x \in R^n$,按一定的规则有一实数与之对应,记为$\|x\|$,若$\|x\|$满足

1, $\|x\| \geq 0$，且$\|x\| = 0$当且仅当$x = 0$; (正定)

2, $\|\alpha x\| = |\alpha| \cdot \|x\|$，$\alpha$为任意实数 (齐次)

3, $\|x + y\| \leq \|x\| + \|y\|$，对任意$x, y \in R^n$ (三角不等式)

则称 $\|x\|$ 为 向量x的范数

几种向量范数：

- 向量的“2”范数（欧几里德范数）

$$\|x\|_2=\sqrt{x_1^2+\cdots+x_n^2}=\left(\sum_{i=1}^{n}x_i^2\right)^{1/2}$$

- 向量的“1”范数

$$\|x\|_1=|x_1|+\cdots+|x_n|=\sum_{i=1}^{n}|x_i|$$

- 向量的“∞”范数（极大值范数或一致向量范数）

$$\|x\|_\infty=\max\{|x_1|,\cdots|x_n|\}=\max_{1\le i\le n}\{|x_i|\}$$

- 向量的“p”范数

$$\|x\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p}$$

矩阵的范数：

定义：对任意n阶方阵A，按一定的规则有一实数与之对应，记为$\|A\|$。若$\|A\|$满足：

1 $\|A\|\ge 0$，且$\|A\|=0$当且仅当$A=0$；（正定）

2 $\|\alpha A\|=|\alpha|\cdot\|A\|$，$\alpha$为任意实数；（齐次）

3 $\|A+B\|\le\|A\|+\|B\|$，对任意A,B两个n阶方阵；（三角不等式）

4 $\|AB\|\le\|A\|\|B\|$；（矩阵乘法不等式，相容性条件）

则称$\|A\|$为矩阵A的范数

几种矩阵范数

- 矩阵的“2”范数（谱范数）

$$\|A\|_2=\left[\lambda_{\max}\left(A^TA\right)\right]^{1/2}$$

- 矩阵的“1”范数（列和范数）

$$\|A\|_1=\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|$$

- 矩阵的“∞”范数（行和范数）

$$\|A\|_\infty=\max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|$$

谱和谱半径：

- A∈Rn×n的特征值为λ1，λ2，…，λn
- 称A的所有特征值的集合为A的谱
- 称ρ(A)=max|λi|为A的谱半径

||A|| 为A的任意一种范数，有ρ(A)≤||A||

严格对角占优阵：

如果A满足条件

$$|a_{ii}|>\sum_{\substack{j=1\\ j\ne i}}^{n}|a_{ij}|\qquad(i=1,2,\cdots,n)$$

即A的每一行对角元素的绝对值都严格大于同行其他元素绝对值之和

病态方程与矩阵条件数：

例：$\begin{bmatrix}1 & 0.99\\ 0.99 & 0.98\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1.99\\ 1.97\end{bmatrix}$　$x=(1,1)^T$

若系数矩阵有微小扰动　$\Delta A=\begin{bmatrix}0.0001 & 0\\ 0 & 0\end{bmatrix}$　$\frac{\|\Delta A\|_\infty}{\|A\|_\infty}=5\times10^{-5}$

则　$\begin{bmatrix}1.0001 & 0.99\\ 0.99 & 0.98\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1.99\\ 1.97\end{bmatrix}\Rightarrow\tilde{x}^{(1)}=(50,-48.5)^T$　$\frac{\|\Delta x^{(1)}\|_\infty}{\|x\|_\infty}=49.5$

若右端有微小扰动　$\Delta b=\begin{bmatrix}-0.0001\\ 0.0001\end{bmatrix}$　$\frac{\|\Delta b\|_\infty}{\|b\|_\infty}=5\times10^{-5}$

则　$\begin{bmatrix}1 & 0.99\\ 0.99 & 0.98\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1.9899\\ 1.9701\end{bmatrix}\Rightarrow\tilde{x}^{(2)}=(2.97,-0.99)^T$　$\frac{\|\Delta x^{(2)}\|_\infty}{\|x\|_\infty}=1.99$

若同时对系数矩阵和右端有扰动，则

$\begin{bmatrix}1.0001 & 0.99\\ 0.99 & 0.98\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1.9899\\ 1.9701\end{bmatrix}\Rightarrow\tilde{x}^{(3)}=(148.5,-148.005)^T$　$\frac{\|\Delta x^{(3)}\|_\infty}{\|x\|_\infty}=149.005$

扰动方程(A+ΔA)x=b+Δb

- 如果方程组的系数或常数项有微小改变时，解会发生很大的改变，则称这种方程组为“病态”的
- 扰动方程的解与原方程的解相对误差不大，称为良态方程
- 方程组的“条件”问题
- 一般说来，在用计算机解方程组时，实际上解的都是扰动方程，这是由于计算过程中不可避免地会产生舍入误差
- 对于良态问题，只要数值方法是稳定的，就可以得到较好的结果
- 而对于病态问题，即使算法是稳定的，其计算结果有时也会很坏

误差界：

b的扰动　$A(x+\Delta x)=b+\Delta b\Rightarrow\Delta x=A^{-1}\Delta b$

$\|\Delta x\|\le\|A^{-1}\|\|\Delta b\|$

$\|b\|=\|Ax\|\le\|A\|\|x\|\Rightarrow\frac{1}{\|x\|}\le\frac{\|A\|}{\|b\|}$

$\Rightarrow\frac{\|\Delta x\|}{\|x\|}\le\|A\|\|A^{-1}\|\frac{\|\Delta b\|}{\|b\|}$

A的扰动　$(A+\Delta A)(x+\Delta x)=b\Rightarrow\Delta x=-A^{-1}\Delta A(x+\Delta x)\Rightarrow\frac{\|\Delta x\|}{\|x+\Delta x\|}\le\|A\|\|A^{-1}\|\frac{\|\Delta A\|}{\|A\|}$

同时扰动　$(A+\Delta A)(x+\Delta x)=b+\Delta b$　$\|A^{-1}\|\|\Delta A\|\le1$

$$\frac{\|\Delta x\|}{\|x\|}\le\frac{\|A\|\|A^{-1}\|}{1-\|A\|\|A^{-1}\|\frac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\Delta b\|}{\|b\|}+\frac{\|\Delta A\|}{\|A\|}\right)$$

条件数：设$A\in R^{n\times n}$，非奇异，$Cond(A)=||A||\,||A^{-1}||$；$K(A)=||A||_2\,||A^{-1}||_2$称为谱条件数

性质：
对任何非奇异矩阵A，有Cond(A)≥1。
对任何非奇异矩阵A，非零常数c，有 Cond(cA) = Cond(A)
若P为正交矩阵，则K(P) =1，且K(PA)=K(AP)=K(A)
若A=AT，$K(A)=\frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|}$

线性方程组Ax=b解的相对误差直接与A的条件数相关:

- A的条件数Cond(A)相对大(>>1)，称Ax=b是病态方程组/坏条件，或A是病态的；
- 当A的条件数Cond(A)相对小，称Ax=b是良态方程组/好条件，或A是良态的。

Hilbert矩阵是一个著名的病态矩阵，它是一个对称正定矩阵，当$n\geq3$时，Hilbert矩阵是病态矩阵

n越大，条件数越大

Matlab 中 hilb() 构 造 hilbert 矩 阵 ，invhilb()可求精确逆

$$A=\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix}$$

3.2.3 相对误差的事后估计（近似解可靠性判别）

设$Ax=b$，A为非奇异矩阵，x为精确解，$\tilde{x}$ 为计算解，残差向量$r=b-A\tilde{x}$ 则近似解的相对误差估计

$$\frac{1}{cond(A)}\frac{\|r(\tilde{x})\|}{\|b\|}\leq\frac{\|\Delta x\|}{\|x\|}\leq cond(A)\frac{\|r(\tilde{x})\|}{\|b\|}$$

- Cond(A)越小，相对误差越小
- 近似解的精度不仅依赖于残差向量r，也与矩阵A的条件数有关

判别病态方程组：

- 当A的行列式值相对小，或A某些行/列近似线性相关，方程组可能病态
- 若用选主元消去法求解Ax=b，在A消去中出现小主元，方程组可能病态
- 当A元素数量级相差很大且无一定规则，方程组可能病态
- 估计条件数，若条件数较大，则方程组病态

迭代求精：

设x为精确解，$\tilde{x}_1$为得到的近似解，则

残差向量 $r_1 = b - A\tilde{x}_1$

$$x = \tilde{x}_1 + \Delta x_1$$

$$Ax = A\tilde{x}_1 + A\Delta x_1$$

$$A\Delta x_1 = r_1$$

通过求解上述方程，可以得到修正因子 Δx_1

由于舍入误差的影响， 同样 $\tilde{x}_2 = \tilde{x}_1 + \Delta x_1$ 不会是精确解，可从 $\tilde{x}_2$出发重复以上步骤

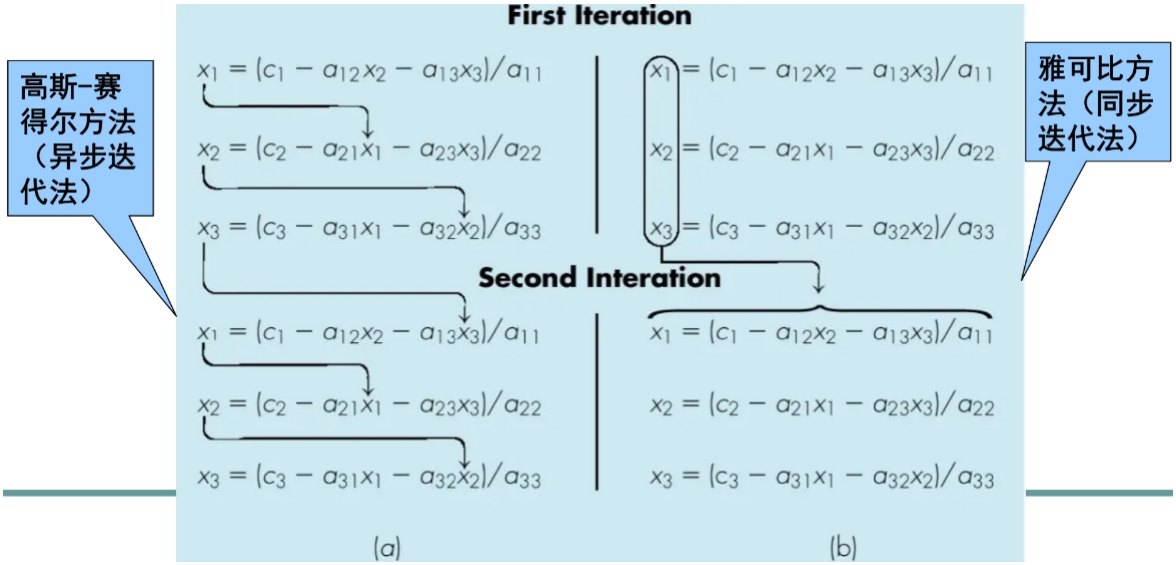
对于LU分解，只需计算残差向量再进行回代

当$Ax=b$不过分病态时，迭代求精是较成功的提高近似解精度的方法

3.2.4 迭代法

基本思想

同步迭代和异步迭代法：



由Ax=b解出x=Gx+f

任取初始向量$X^{(0)}$，$X^{(1)} = GX^{(0)} + f$，$X^{(k+1)} = GX^{(k)} + f$

构造向量序列$\{x^{(k)}\}$求方程的近似解的方法，称为一阶定常迭代法，G为该迭代法的迭代矩阵。

如果对任意取初始近似$\boldsymbol{x}^{(0)}$，都有 $\lim_{k\to\infty} x^{(k)} = x^*$ ，称迭代法为**收敛**，否则称迭代法为**发散**。若迭代法收敛，则称$\boldsymbol{x}^{(k)}$为第k步迭代得到方程组的近似解。

基本迭代方法

A分裂为A=M-N

分裂阵M：可选择的非奇异阵，Mx=d易于求解，M选为A的某种近似

$Ax = b \qquad Mx = Nx + b \qquad x = M^{-1}Nx + M^{-1}b$

迭代方程：$x^{(0)}$为初始向量

$x^{(k+1)}=M^{-1}Nx+M^{-1}b$

选取不同的M阵就是不同迭代法

$$A=\begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix}-\begin{bmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix}-\begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \cdots & -a_{2n} \\ & & \ddots & \vdots \\ & & & 0 \end{bmatrix}=D-L-U$$

雅各比迭代法（同步迭代法）

设A是非奇异矩阵且$a_{ii}\neq 0$，令M=D，N=L+U

$x^{(k+1)}=Jx^{(k)}+f$，$J=D^{-1}(L+U)$，　　$f=D^{-1}b$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}=\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}\Rightarrow\begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \dots & \\ & & & a_{nn} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}=\begin{bmatrix} 0 & -a_{12} & \dots & -a_{1n} \\ -a_{21} & 0 & \dots & -a_{2n} \\ \dots & \dots & \dots & \dots \\ -a_{n1} & -a_{n2} & \dots & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}+\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

(1)雅可比迭代法，每迭代一次主要是计算一次矩阵乘向量，即$Jx^{(k)}$。

(2)计算过程中，原始数据A始终不变。

(3)计算中需要**两组工作单元**来保存$x^{(k)}$及$x^{(k+1)}$。

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{bmatrix}=\begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \dots & -\frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & & \vdots \\ -\frac{a_{n1}}{a_{nn}} & \dots & \dots & 0 \end{bmatrix}\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix}+\begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{bmatrix}$$

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{\substack{j=1\\ j\neq i}}^{n}a_{ij}x_j^{(k)}\right)$$

高斯-赛德尔(Gauss-Seidel)迭代法（异步迭代法）

令M=D-L和N=U

$x^{(k+1)}=Gx^{(k)}+f$，$G=(D-L)^{-1}U$，　　$f=(D-L)^{-1}b$

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right)$$

(1)G-S迭代法每迭代一次主要是计算一次矩阵乘向量。

(2)计算$x^{(k+1)}$的第i个分量$x_i^{(k+1)}$时，利用已计算出的最新分量$x_j^{(k+1)}(j=1,\dots,i-1)$，因此，计算中只需要**一组工作单元**来保存$x^{(k)}$或$x^{(k+1)}$。

迭代终止准则

近似相对百分比误差

$$\left|\varepsilon_a\right|_i=\left|\frac{x_i^{(k+1)}-x_i^{(k)}}{x_i^{(k+1)}}\right|\times 100$$

对所有的i，近似相对百分误差小于预设的εs时，迭代终止。或者误差为$\dfrac{\left\|x^{(k+1)}-x^{(k)}\right\|}{\left\|x^{(k+1)}\right\|}$

迭代法收敛性

$x=Gx+f$　　$x^*=Gx^*+f$

迭代$x^{(k+1)}=Gx^{(k)}+f$

引入误差向量$e^{(k)}=x^{(k)}-x^*$

误差向量的递推公式：　$e^{(k+1)}=Ge^{(k)}$

于是$e^{(k)}=Ge^{(k-1)}=G^2e^{(k-2)}=\ldots=G^ke^{(0)}$　（$e^{(0)}=x^{(0)}-x^{(*)}$）

$$\left\|e^{(k)}\right\|=\left\|G^ke^{(0)}\right\|\le\|G\|^k\left\|e^{(0)}\right\|=q^k\left\|e^{(0)}\right\|$$

定理：设$G\in R^{n\times n}$，则$G^k\to0$（零矩阵）（当$k\to\infty$时）的充要条件为G所有特征值满足$|\lambda_i|<1$（$i=1,2,\ldots,n$）或G的谱半径$\rho(G)<1$

一阶定常迭代法收敛性的基本定理

- 设有方程组$x=Gx+f$，有迭代法$x^{(k+1)}=Gx^{(k)}+f$，则对任选初始向量$x^{(0)}$，迭代法收敛的充要条件是$\rho(G)<1$。

迭代收敛的**充分**条件

- 设有方程组$x=Gx+f$及一阶定常迭代法$x^{(k+1)}=Gx^{(k)}+f$，如果有G的某种范数$\|G\|_r=q<1$，则
  - 迭代法收敛；
  - 误差估计

$$\left\|x^*-x^{(k)}\right\|\le\frac{q}{1-q}\left\|x^{(k)}-x^{(k-1)}\right\|<\frac{1}{1-q}\left\|x^{(k)}-x^{(k-1)}\right\|$$　事后估计

$$\left\|x^*-x^{(k)}\right\|\le\frac{q^k}{1-q}\left\|x^{(1)}-x^{(0)}\right\|$$　事前估计

  - 当$q\approx1$时，迭代法收敛缓慢。

q越小收敛速度越快；可以事先估计保证误差$\|x^*-x^{(k)}\|_\infty<\varepsilon$所需要的迭代次数。

Jacobi迭代法收敛的充要条件是$\rho(J)<1\quad(J=D^{-1}(L+U))$
G-S迭代法收敛的充要条件是$\rho(G)<1\quad(G=(D-L)^{-1}U)$

如果A为（按行或列）严格对角占优阵，则Ax=b的Jacobi方法和G-S方法都收敛。G-S迭代法比Jacobi迭代法收敛得快。实际上，如果A为严格对角占优阵，可以证明，$||J||_\infty<1$并且$||G||_\infty<1$，因此，Jacobi方法和G-S方法都收敛。

如果A为对称正定矩阵，G-S迭代收敛。

逐次超松驰迭代法(SOR，Successive Over-Relaxation)

在G-S方法上修改

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\omega\tilde{x}_i^{(k+1)}=x_i^{(k)}+\frac{\omega}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i}^{n}a_{ij}x_j^{(k)}\right)$$

$0<\omega<2$，称为松驰因子

其中 $\tilde{x}_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right)$ G-S方法得到的解

- ω=1，SOR方法即为G-S方法
- 0<ω<1，结果为当前迭代结果和上一次迭代结果的加权平均，称为低松弛方法。用于使得非收敛方程组收敛或者克服振荡加速收敛。
- 1<ω<2，超松弛方法：隐含假设：新值沿正确方向向真实解移动，但是移动的速度慢；用于加速已知是收敛的方程组的收敛速度；根据经验确定ω值

矩阵形式

SOR的矩阵形式

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right)$$

$M=(D-\omega L)/\omega$

$$(D-\omega L)x^{(k+1)} = \left[(1-\omega)D + \omega U\right]x^{(k)} + \omega b$$

$$x^{(k+1)} = G\omega x^{(k)} + f$$

$$G\omega = (D-\omega L)^{-1}\left[(1-\omega)D + \omega U\right]$$

$$f = \omega(D-\omega L)^{-1}b$$

SOR法收敛的充要条件是$\rho(G\omega)<1$

如果A为对称正定阵，且$0<\omega<2$，则解$Ax=b$的SOR法收敛。

3.3 MATLAB自带函数

矩阵分析		线性方程组	
函数	描述	函数	描述
cond	计算矩阵条件数	右除/和左除\	help slash
norm	计算矩阵或向量的范数	lu	lu分解
inv	矩阵求逆	chol	cholesky分解
pinv	求矩阵伪逆	rref	高斯-约当法
det	计算行列式的值		
rank	求秩		
eig，eigs	矩阵特征值		

方法	稳定性	精度	应用范围	编程难度	备注
图解法	-	差	受限	-	比数值方法耗时
克莱默法则	-	受舍入误差影响	受限	-	方程数多于3个时计算复杂
列主元高斯消元法	-	受舍入误差影响	一般，适用于方程组系数矩阵为低阶稠密矩阵、带状矩阵	中等，计算量较小，存储量较大	-
LU分解	-	受舍入误差影响	一般，适用于方程组系数矩阵为低阶稠密矩阵、带状矩阵	中等，计算量较小，存储量较大	常用；进行矩阵求逆计算

方法	稳定性	精度	应用范围	编程难度	备注
迭代法	可能不收敛	优秀	收敛时，适用于大型稀疏线性方程组	较简单，计算量有时较大，存储量较小	-

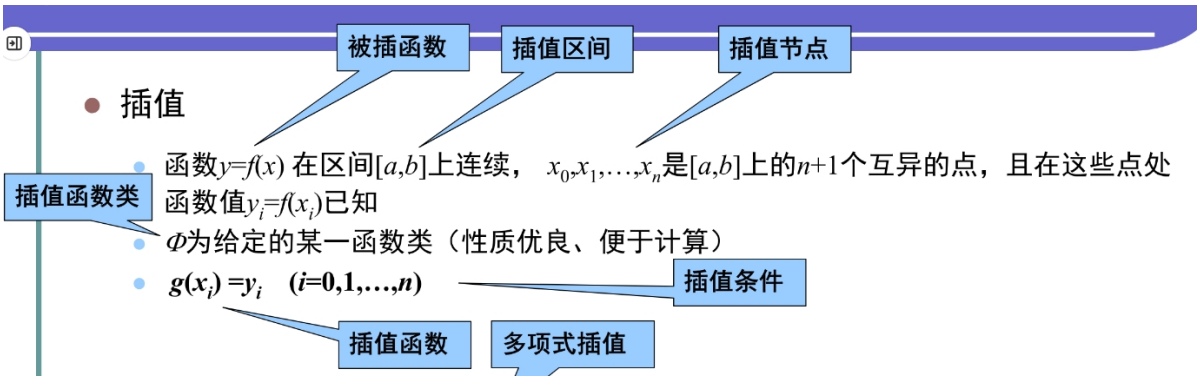
四，插值和拟合

问题：有的函数虽有表达式，但较复杂，也可用简单的函数g(x)来逼近它
　　某些变量之间的函数关系y=f(x)存在，但没有f(x)的解析式，y=f(x)以函数表格或曲线形式给出

要求：根据函数表推算该函数在某些点上的函数值。
解决与该函数有关的一些问题，如分析函数的性态，研究y=f(x)的变化规律，求导数、积分、零点与极值点等。

4.1 插值问题

4.1.1 定义



4.1.2 插值函数类

代数多项式：选取多项式Pn作为插值函数，Pn(x)称为插值多项式。

$$p_n(x)=\sum_{i=0}^{n}a_ix^i$$

有理函数：用有理函数（多项式的商）作为插值多项式。

$$\Phi^{n,m}(x)=\frac{P_n(x)}{Q_m(x)}=\frac{p_0+p_1x_1+\ldots+p^nx^n}{q_0+q_1x_1+\ldots+q^mx^m}$$

有理插值可使区间内插值误差分布较为均匀，特别适用于某些被插函数具有无穷间断点的附近，这种情况下若用多项式逼近效果很差。

三角函数：选取正弦和余弦等三角函数作为插值函数。

4.1.3 插值函数唯一性定理

设 $g(x)=a_0\varphi_0(x)+\cdots+a_n\varphi_n(x)$

则 $g(x_i)=f(x_i)=a_0\varphi_0(x_i)+\cdots+a_n\varphi_n(x_i)$

$$\begin{bmatrix}\varphi_0(x_0) & \cdots & \varphi_n(x_0)\\ \vdots & \ddots & \vdots\\ \varphi_0(x_n) & \cdots & \varphi_n(x_n)\end{bmatrix}\begin{bmatrix}a_0\\ \vdots\\ a_n\end{bmatrix}=\begin{bmatrix}f(x_0)\\ \vdots\\ f(x_n)\end{bmatrix}$$

则 $\begin{bmatrix}a_0\\ \vdots\\ a_n\end{bmatrix}$ 有唯一解 $\begin{vmatrix}\varphi_0(x_0) & \cdots & \varphi_n(x_0)\\ \vdots & \ddots & \vdots\\ \varphi_0(x_n) & \cdots & \varphi_n(x_n)\end{vmatrix}\neq 0$

- 与基函数无关
- 与原函数$f(x)$无关
- 基函数个数与点个数相同

定理：$\{x_i\}_{i=0}^n$ 为$n+1$个节点，$\Phi=span\{\varphi_0,\varphi_1,\cdots\varphi_n\}$

$n+1$维空间，则插值函数存在唯一，当且仅当

$$\begin{vmatrix}\varphi_0(x_0) & \cdots & \varphi_n(x_0)\\ \vdots & \ddots & \vdots\\ \varphi_0(x_n) & \cdots & \varphi_n(x_n)\end{vmatrix}\neq 0$$

$$\Phi=P_n(x)=span\{1,x,x^2,\cdots x^n\}$$

$$V=\begin{vmatrix}1 & x_0 & x_0^2 & \cdots & x_0^n\\ 1 & x_1 & x_1^2 & \cdots & x_1^n\\ \vdots & \vdots & \vdots & \cdots & \vdots\\ 1 & x_n & x_n^2 & \cdots & x_n^n\end{vmatrix}=\prod_{i=1}^{n}\prod_{j=0}^{i-1}(x_i-x_j)$$

$n+1$阶范德蒙(Vandermonde)行列式

- 若插值节点互异，则存在唯一的多项式$P_n(x)$使$f(x_i)=P_n(x_i)$，$i=0,1,\ldots,n$

4.1.4 插值余项——截断误差

Rn=f(x)-Pn(x)

- 定理：
  - 若$f(x)$在区间$[a,b]$上有直到$n+1$阶导数$f^{(n+1)}(x)$存在，$P_n(x)$为$f(x)$在$n+1$个节点x_i上的n次插值多项式，则$\forall x\in[a,b]$有

$$R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

$\xi\in(a,b)$，且依赖于x

➢$f^{(n+1)}(\xi)$对余项的影响

➢在实际问题中，往往不知道$f(x)$的具体解析表达式，因此要估计$f^{(n+1)}(x)$的上界是不现实的。在这种情形下，可以采用**误差的事后估计法**。

$\omega_{n+1}(x)=\prod_{i=0}^{n}(x-x_i)$

➢当插值点x位于插值区间的节点附近时，插值误差较小，而距离节点较远处插值误差较大。

插值多项式 仅与已知数据有关，与f(x)原本形式无关，但是余项与f(x)密切相关。

若f(x)本身是一个不超过n次的多项式，那么Pn(x)=f(x)

对多项式插值，增加阶数不一定能提高精度，一般用3-4个。

插值点x不能位于插值区间之外的远处。如果数据表中包含的点多于插值所需的数目，则在每次插值时，就必须选取表中合适位置的点。

4.1.5 事后估计

- $\{x_i\}_{i=0}^{n+1}$ 任取$n+1$个构造$P_n(x)$

$$\begin{aligned} i=0,\cdots,n &\Rightarrow P_n(x) \\ i=1,\cdots,n+1 &\Rightarrow \tilde{P}_n(x) \end{aligned}$$

$$f(x)-P_n(x)=\frac{f^{(n+1)}(\xi_1)}{(n+1)!}(x-x_0)\cdots(x-x_n)$$

$$f(x)-\tilde{P}_n(x)=\frac{f^{(n+1)}(\xi_2)}{(n+1)!}(x-x_1)\cdots(x-x_{n+1})$$

- 假设 $f^{(n+1)}(\xi_1)\approx f^{(n+1)}(\xi_2)$ ⇨

$$\frac{f(x)-P_n(x)}{f(x)-\tilde{P}_n(x)}\approx\frac{x-x_0}{x-x_{n+1}}$$

$$\Rightarrow f(x)\approx\frac{x-x_{n+1}}{x_0-x_{n+1}}P_n(x)+\frac{x-x_0}{x_{n+1}-x_0}\tilde{P}_n(x)$$

$$\Rightarrow f(x)-P_n(x)\approx\frac{x-x_0}{x_0-x_{n+1}}\left(P_n(x)-\tilde{P}_n(x)\right)$$

通过两个结果的偏差来估计插值误差

4.2 插值方法

4.2.1 拉格朗日插值多项式

对于n个点，先求出每个点的基函数对于的lk(x)，然后再求Ln(x)

$$l_k(x_i)=\begin{cases}0, i\neq k\\ 1, i=k\end{cases}$$

x	x_0	x_1	…	x_k	…	x_n
$l_k(x)$	0	0	…	1	…	0

- 可设

$$l_k(x)=A_k(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)$$

- 则

$$A_k=\frac{1}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

$$l_k(x)=\frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}=\prod_{\substack{i=0\\k\neq i}}^{n}\frac{(x-x_i)}{(x_k-x_i)}$$

n次基本插值多项式或n次插值基函数

Lagrange插值多项式：取$l_i(x)$为基函数，令$a_i=y_i$

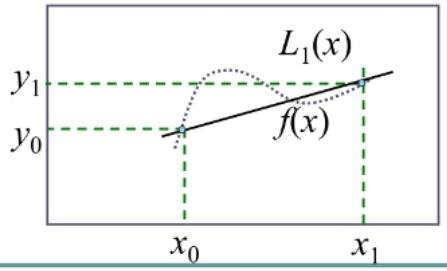
$$L_n(x)=y_0l_0(x)+y_1l_1(x)+\cdots+y_nl_n(x)=\sum_{i=0}^{n}l_i(x)y_i$$

线性插值(n=1)两个点

- 线性插值（n=1）

$$l_0(x)=\frac{x-x_1}{x_0-x_1}\qquad l_1(x)=\frac{x-x_0}{x_1-x_0}$$

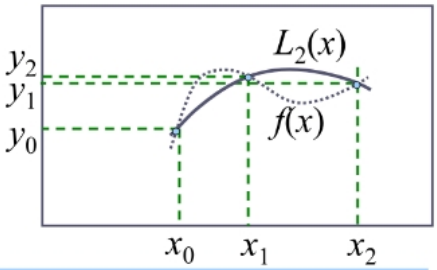
$$L_1(x)=y_0\frac{x-x_1}{x_0-x_1}+y_1\frac{x-x_0}{x_1-x_0}=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0)$$



二次(抛物线)插值(n=2)三个点

- 二次（抛物线）插值（n=2）

$$l_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$$
$$l_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$
$$l_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$



$$L_2(x)=y_0\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}+y_1\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}+y_2\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

选用更小的区间插值得出的结果更优
高次插值往往优于低次插值

内插与外推

内插：插值点位于插值区间内部
外推：插值点位于插值区间外部但是与端点较近。

采用外推法时插值点x不宜位于插值区间之外的远处。

缺点

缺点：没有承袭性，增加一个节点，所有基函数都要重新算

当插值节点在区间[a,b]上，适当加密，其误差在多数情况下会小些，但节点过多，不仅高次Ln(x)计算量大，精度也不一定好。$lim_{n->\infty}L_n(x)=f(x)$一般不成立。

代码实现

```
syms x;
function f=lagrange(x,X,Y)
    n=length(X);
    f=0;
    for i=1:n
        up=1;
        down=1;
        for j=1:n
            if i~=j
                up=up*(x-X(j));%求基函数分子
            end
        end
        for j=1:n
            if i~=j
                down=down*(X(i)-X(j));%求基函数分母
            end
        end
        lk=up/down;%基函数
        f=f+Y(i)*lk;%累加
    end
end
```

4.2.2 牛顿插值法

原理

承袭性：$N_{n+1}(x) = N_n(x) + q_{n+1}(x)$

$$N_n(x) = a_0 + a_1(x-x_0) + \cdots + a_n(x-x_0)\cdots(x-x_{n-1})$$

$$\begin{cases} N_n(x_0) = a_0 = f(x_0) \\ N_n(x_1) = a_0 + a_1(x_1-x_0) = f(x_1) \\ N_n(x_2) = a_0 + a_1(x_2-x_0) + a_2(x_2-x_0)(x_2-x_1) = f(x_2) \\ \vdots \\ N_n(x_n) = a_0 + a_1(x_n-x_0) + \cdots + a_n(x_n-x_0)\cdots(x_n-x_{n-1}) = f(x_n) \end{cases}$$

连接x_0,x_1两点直线的斜率

$$a_0 = f(x_0)$$

$$a_1 = \frac{f(x_1)-f(x_0)}{x_1-x_0}$$

$$a_2 = \frac{1}{x_2-x_1}\left(\frac{f(x_2)-f(x_0)}{x_2-x_0} - a_1\right) \Rightarrow a_2 = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0}$$

$$a_3 = \frac{1}{x_3-x_2}\left(\left(\frac{f(x_3)-f(x_0)}{x_3-x_0} - a_1\right)\frac{1}{x_1-x_0} - a_2\right)$$

$$\vdots$$

有限差商

设实数$x_0,x_1,\ldots,x_n$互异，$f(x)$关于点的

n阶差商可表示为函数值的线性组合

$$f[x_0,\cdots,x_n] = \sum_{i=0}^{n}\frac{f(x_i)}{\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)}$$

- 零阶差商 $f[x_i] = f(x_i)$
- 一阶差商 $f[x_i,x_j] = \frac{f(x_i)-f(x_j)}{x_i-x_j}$
- 二阶差商 $f[x_i,x_j,x_k] = \frac{f[x_i,x_j]-f[x_j,x_k]}{x_i-x_k}$
- k阶差商 $f[x_i,x_{i+1},\cdots,x_{i+k}] = \frac{f[x_{i+1},\cdots,x_{i+k}]-f[x_i,\cdots,x_{i+k-1}]}{x_{i+k}-x_i}$

对称性 $f[x_0,\cdots,x_k] = f[x_{i_0},\cdots,x_{i_k}]$ $i_0,\cdots,i_k$是$0,\cdots,k$的任意排列

重点在于右上角的公式

这样可得
a0=f[x0]
a1=f[x0,x1]
a2=f[x0,x1,x2]
an=f[x0,x1...xn]
牛顿插值基本多项式：

$$N_n(x) = f(x_0) + f[x_0,x_1](x-x_0) + \cdots + f[x_0,\cdots,x_n](x-x_0)\cdots(x-x_{n-1})$$

拉格朗日插值多项式与牛顿插值多项式是同一个n次多项式。

误差估计

$$R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$$

- 增加一个节点a

$$N_{n+1}(x)=N_n(x)+f[x_0,\cdots,x_n,a](x-x_0)\cdots(x-x_n)$$

$$N_{n+1}(a)=f(a)\qquad f(a)-N_n(a)=f[x_0,\cdots,x_n,a](a-x_0)\cdots(a-x_n)$$

$$f[x_0,\cdots,x_n,a]=\frac{f^{n+1}(\xi)}{(n+1)!}$$

➢n阶多项式的误差估计等于$n+1$阶和n阶估计值之差

$$R_n(x)=N_{n+1}(x)-N_n(x)$$

➢如果有额外的数据点$f(x_{n+1})$，可以近似计算

选择节点的重要性：应尽可能集中或靠近未知点

n+1阶和n阶估计值之差小于真实误差，因此采用两次迭代结果之差无法作为迭代的停止准则。实际上，高次多项式插值往往会发散。

等距插值

数据在空间上间距相等

- $x_0<\ldots<x_n$，其中$x_i-x_{i-1}=h$，$x_i=x_0+ih$，h称作步长
- 有限差分

$\Delta f(x)=f(x+h)-f(x)$ 一阶向前差分

$\nabla f(x)=f(x)-f(x-h)$ 一阶向后差分

$\delta f(x)=f(x+h/2)-f(x-h/2)$ 一阶中心差分

Δ称为前差算子，∇称为后差算子，δ称为中差算子

n阶前差 $\Delta^n f(x)=\Delta[\Delta^{n-1}f(x)]=\Delta^{n-1}f(x+h)-\Delta^{n-1}f(x)$

n阶后差 $\nabla^n f(x)=\nabla[\nabla^{n-1}f(x)]=\nabla^{n-1}f(x)-\nabla^{n-1}f(x-h)$

n阶中差 $\delta^n f(x)=\delta[\delta^{n-1}f(x)]=\delta^{n-1}f(x+h/2)-\delta^{n-1}f(x-h/2)$

规定 $\Delta^0 f(x)=f(x)$，$\nabla^0 f(x)=f(x)$，$\delta^0 f(x)=f(x)$

差商与有限差分的关系

前差

$$f[x_0,x_1,\cdots,x_n]=\frac{\Delta^n f(x_0)}{n!h^n}$$

后差

$$f[x_0,x_1,\cdots,x_n]=\frac{\nabla^n f(x_n)}{n!h^n}$$

牛顿向前插值公式

x在节点x_0的附近($x>x_0$)，将节点按$x_0,\ldots,x_n$的顺序排列

$$N_n(x)=f(x_0)+f[x_0,x_1](x-x_0)+\cdots+f[x_0,\cdots,x_n](x-x_0)\cdots(x-x_{n-1})$$

$$f[x_0,x_1,\cdots,x_n]=\frac{\Delta^n f(x_0)}{n!h^n}$$

$$N_n(x)=f(x_0)+\frac{(x-x_0)}{h}\Delta f(x_0)+\frac{(x-x_0)(x-x_1)}{2!h^2}\Delta^2 f(x_0)+\cdots+\frac{(x-x_0)\cdots(x-x_{n-1})}{n!h^n}\Delta^n f(x_0)$$

令$x=x_0+th$

$$N_n(x)=f(x_0+th)=f(x_0)+t\Delta f(x_0)+\frac{t(t-1)}{2}\Delta^2 f(x_0)+\cdots+\frac{t(t-1)\cdots(t-n+1)}{n!}\Delta^n f(x_0)$$

- 常用来计算表头x_0附近的函数值，$x\in[x_0,x_1]$
- 余项

$$R_n(x_0+th)=\frac{t(t-1)\cdots(t-n)}{(n+1)!}h^{n+1}f^{(n+1)}(\xi)\qquad \xi\in(x_0,x_n)$$

牛顿向后插值公式

x在节点x_n的附近$(x<x_n)$，将节点按$x_n,\dots,x_0$的顺序排列

$$N_n(x)=f(x_n)+f[x_n,x_{n-1}](x-x_n)+\cdots+f[x_n,\cdots,x_0](x-x_n)\cdots(x-x_1)$$

$$f[x_n,x_{n-1},\cdots,x_0]=f[x_0,x_1,\cdots,x_n]=\frac{\nabla^n f(x_n)}{n!h^n}$$

$$N_n(x)=f(x_n)+\frac{x-x_n}{h}\nabla f(x_n)+\frac{(x-x_n)(x-x_{n-1})}{2!h^2}\nabla^2 f(x_n)+\cdots+\frac{(x-x_n)\cdots(x-x_1)}{n!h^n}\nabla^n f(x_n)$$

令$x=x_n+th$ （一般$t<0$）

$$N_n(x)=f(x_n+th)=f(x_n)+t\nabla f(x_n)+\frac{t(t+1)}{2}\nabla^2 f(x_n)+\cdots+\frac{t(t+1)\cdots(t+n-1)}{n!}\nabla^n f(x_n)$$

常用来计算表尾x_n附近的函数值，即$x\in[x_n,x_{n-1}]$

$$R_n(x_n+th)=\frac{t(t+1)\cdots(t+n)}{(n+1)!}h^{n+1}f^{(n+1)}(\xi)$$

hermite插值多项式

构造插值函数除了函数值的条件以外，还需要一定的连续性条件，如一阶导数值等

> 构造插值函数除了函数值的条件以外，还需要一定的连续性条件，如一阶导数值等
>
> 定义：满足$\{x_i,f(x_i)\}_{i=0}^n$,和$\{(x_i,f^k(x_i),k=1,\cdots,k_i),\}_{i=0}^n$条件的插值称为Hermite插值
>
> 以所有k_i=1为例，即每个点上还要满足一阶导数条件
>
> $$\{x_i,f(x_i)\}_{i=0}^n,\text{和}\{x_i,f'(x_i)\}_{i=0}^n$$
>
> 在设计交通工具的外形时，参照海豚的标本上已知点及已知点的导数，做插值在计算机上模拟海豚的外形制成飞机、汽车等外形。

Runge现象

n越大，端点附近抖动越大，成为Runge现象

适当提高插值多项式的次数，可能提高计算结果的准确程度。然而次数越高意味着参加插值节点数越多，计算量大，插值函数曲线在部分区间上（两端）发生激烈振荡，插值多项式截断误差/计算余项偏大。

龙格/Runge 现象说明加密节点并不能保证所得到的插值多项式能更好地逼近f(x)，高次插值效果不一定比低次插值好。

- 截断误差是插值的收敛性问题，Runge现象
- 舍入误差（在插值计算过程中可能被扩散或放大）是插值的稳定性问题。
  - 随插值多项式次数的提高，计算误差的影响也会增大。假设给出$f(x)$的一组等距节点上x_i的函数值y_i 有初始误差变为$\tilde{y}_i$，误差限为ε，则拉格朗日插值多项式$L_n(x)$的计算值 $\tilde{L}_n(x)$也会有误差：

$$\left|L_n(x)-\tilde{L}_n(x)\right|=\left|\sum_{i=0}^{n}l_i(x)y_i-\sum_{i=0}^{n}l_i(x)\tilde{y}_i\right|$$

$$\leq\sum_{i=0}^{n}\left|l_i(x)\right|\left|y_i-\tilde{y}_i\right|\leq\sum_{i=0}^{n}\left|l_i(x)\right|\varepsilon=\varepsilon\cdot\sum_{i=0}^{n}\left|l_i(x)\right|$$

n很大时，此项>>1，插值多项式的计算误差大，即当n增大时，拉格朗日插值法不稳定。

- 为避免龙格现象和不稳定，常限定n<7。当n较大时不采用高次多项式插值，而采用分段低次插值、样条插值或用次数较低的最小二乘逼近。

代码实现

```
syms x;
function result=different(X,Y)
    n=length(X);
    result=0;
    for i=1:n
        down=1;
        for j=1:n
            if i~=j
                down=down*(X(i)-X(j));
            end
        end
        lk=Y(i)/down;
        result=result+lk;
    end
end

function f=newton(x,X,Y)
    f=Y(1);
    n=length(X);
    for i=2:n
        k=1;
        result=different(X(1:i),Y(1:i));
        k=result;
        for j=1:i-1
            k=k*(x-X(j));
        end
        f=f+k;
    end
end
```

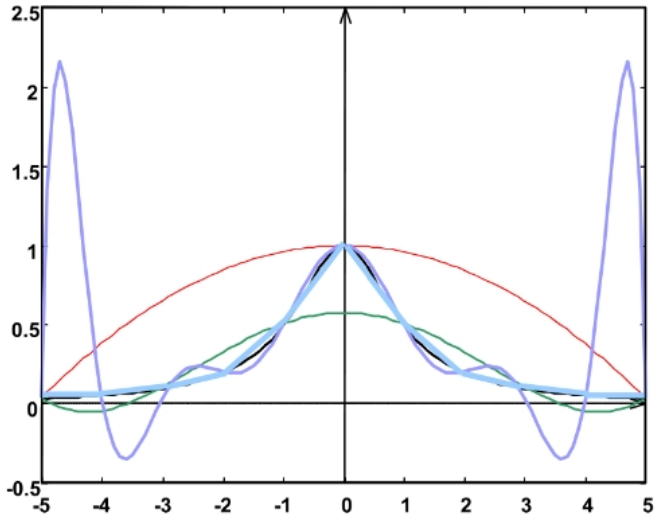
4.2.3 分段低次插值

把整个插值区间分成若干个小区间，在每个小区间上，进行低次插值

设a = x0< x1< x2< ...< xn-1< xn = b，则节点把[a, b]分成n个小区间

当插值点x在第i个小区间[xi-1,xi]上时，采用一次线性插值公式，称为分段线性插值。也称为折线插值，即通过曲线n+1个点(xi,yi)的折线去近似替代曲线

在每个小区间[xi-1,xi,xi+1]上使用二次插值公式，称为分段抛物插值或分段二次插值



优点：分段低次插值函数公式简单，只要区间充分小，就能保证误差要求；局部性质好，如果修改了某个节点xi的值，仅在相邻的两个区间[xi-1, xi]，[xi, xi+1]受到影响

缺点：不能保证节点处插值函数的导数连续，因而不能满足某些工程上技术上曲线光滑性的要求

代码实现

```
syms x;
function f=cut(x,X,Y)%分段线性插值，可以改区间
    f_set=[];
    n=length(X);
    for i=1:n-1
        X_part=X(i:i+1);
        Y_part=Y(i:i+1);
        f_element=lagrange(x,X_part,Y_part);
        f_set=[f_set,f_element];
    end
    f=f_set(1).*(x>=X(1)&x<=X(2));
    for i=2:length(f_set)
        f=f+f_set(i).*(x>X(i)&x<=X(i+1));
    end
end
```

4.2.4 样条插值

线性样条：

- 分段线性插值
- 节点处一阶导数不连续（不光滑）

二次样条：

- 在节点处一阶导数连续
- 每两个相邻节点组成的区间中推导一个不 高于二阶的多项式

三次样条：

- 节点处具有连续的一阶、二阶导数节点处具有连续的一阶、二阶导数
- 每两个相邻节点组成的区间中推导一个不 高于三阶的多项式
- 三阶导数和高阶导数可能不连续

定义：设f(x) 是区间[a,b]上的一个二次(m-1次)连续可微函数。在区间上给定一组节点： a=x0 <...<xn=b, 设函数

$$S(x)=\begin{cases} S_1(x), & x\in[x_0,x_1] \\ \cdots & \\ S_i(x), & x\in[x_{i-1},x_i] \\ \cdots & \\ S_n(x), & x\in[x_{n-1},x_n] \end{cases}$$

满足条件：

- S(x)在区间[a,b]上存在二阶(m-1阶)的连续导数。
- 每个子区间[x(i−1) ,xi]上Si (x)都是一个不高于3次(m次)的多项式；
- 满足插值条件S(xi)=f(xi)，i=0,...,n。

那么S(x)为函数f(x)关于节点x0，x1，...，xn的三次(m次)样条插值函数，简称三次(m次)样条。

二次样条

$$f_i(x)=a_ix^2+b_ix+c_i \quad x\in[x_{i-1},x_i] \quad i=1,2,\dots,n$$

相邻多项式在内部节点处函数值相等

$$a_{i-1}x_{i-1}^2+b_{i-1}x_{i-1}+c_{i-1}=f(x_{i-1})$$
$$a_ix_{i-1}^2+b_ix_{i-1}+c_i=f(x_{i-1})$$

第一个和最后一个函数通过端点

$$a_1x_0^2+b_1x_0+c_1=f(x_0)$$
$$a_nx_n^2+b_nx_n+c_n=f(x_n)$$

内部节点的一阶导数相等

$$2a_{i-1}x_{i-1}+b_{i-1}=2a_ix_{i-1}+b_i \quad i=2,3,\dots,n$$

在第一个节点处二阶导数为0，a1=0，连接前两点的为一条直线。

三次样条插值函数

$$S_i(x)=a_ix^3+b_ix^2+c_ix+d_i,x\in[x_{i-1},x_i] \qquad i=1,\cdots,n$$

有4n个未知数

满足条件

$$\left.\begin{aligned} S(x_i-0)&=S(x_i+0) && i=1,\cdots,n-1\\ S'(x_i-0)&=S'(x_i+0) && i=1,\cdots,n-1\\ S''(x_i-0)&=S''(x_i+0) && i=1,\cdots,n-1\\ S(x_i)&=f(x_i) && i=0,\cdots,n \end{aligned}\right\}\ 4n\text{-}2\text{个}$$

边界条件

1. 假设在两端点的导数$f'(a)=f_0'$和$f'(b)=f_n'$为已知， $S'(x_0)=f_0'$, $S'(x_n)=f_n'$
2. 假设在两端点的二阶导数$f''(a)=f_0''$和$f''(b)=f_n''$为已知， $S''(x_0)=f_0''$, $S''(x_n)=f_n''$。若$S''(x_0)=S''(x_n)=0$，称为自然边界条件
3. 给定函数具有周期特性，$S^{(t)}(x_0+0)=S^{(t)}(x_n-0)$, $t=0,1,2$

三弯矩法

- 利用$S(x)$在节点x_i处的二阶导数值$M_i=S''(x_i)$ 表示$S_i(x)$。用$S'(x)$在内节点x_i上的连续性和边界条件来确定M_i。
- $S''(x_i)$在每一个子区间上是线性函数

$$S_i''(x)=M_{i-1}\frac{x-x_i}{x_{i-1}-x_i}+M_i\frac{x-x_{i-1}}{x_i-x_{i-1}} \qquad x\in[x_{i-1},x_i] \qquad h_i=x_i-x_{i-1}$$

- 积分两次，并利用插值条件$S_i(x_{i-1})=f(x_{i-1}),S_i(x_i)=f(x_i)$可得

$$S_i(x)=\frac{1}{6h_i}[M_{i-1}(x_i-x)^3+M_i(x-x_{i-1})^3]+(f_{i-1}-\frac{M_{i-1}h_i^2}{6})\frac{x_i-x}{h_i}+(f_i-\frac{M_ih_i^2}{6})\frac{x-x_{i-1}}{h_i}$$

$$S_i'(x)=-\frac{M_{i-1}}{2h_i}(x_i-x)^2+\frac{M_i}{2h_i}(x-x_{i-1})^2+f[x_{i-1},x_i]-\frac{h_i}{6}(M_i-M_{i-1})$$

n+1个未知数，n-1个方程

- 由$S'(x_i+0)=S'(x_i-0)$

$$\mu_iM_{i-1}+2M_i+\lambda_iM_{i+1}=g_i \qquad i=1,2,\cdots,n-1$$

$$\mu_i=\frac{h_i}{h_i+h_{i+1}} \qquad \lambda_i=\frac{h_{i+1}}{h_i+h_{i+1}}=1-\mu_i \qquad g_i=\frac{6}{h_i+h_{i+1}}\left(f[x_i,x_{i+1}]-f[x_{i-1},x_i]\right)=6f[x_{i-1},x_i,x_{i+1}]$$

边界条件1

$S'(x_0)=f_0'$ => $2M_0+M_1=6([f[x_0,x_1]-f_0'])/h_1$

$S'(x_n)=f_n'$ => $M_{n-1}+2M_n=6(f_n'-f[x_{n-1},x_n])/h_n$

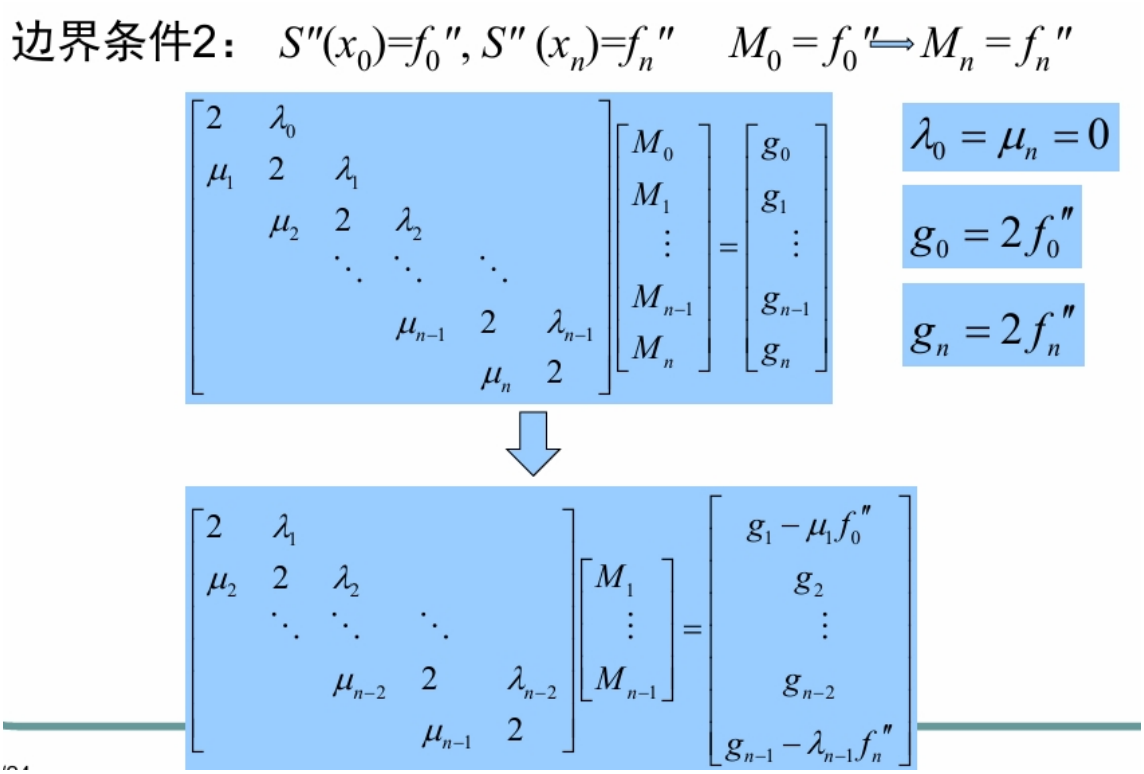
边界条件

$$\begin{bmatrix} 2 & \lambda_0 & & & & \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \lambda_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & \mu_n & 2 \end{bmatrix}\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix}=\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix}$$

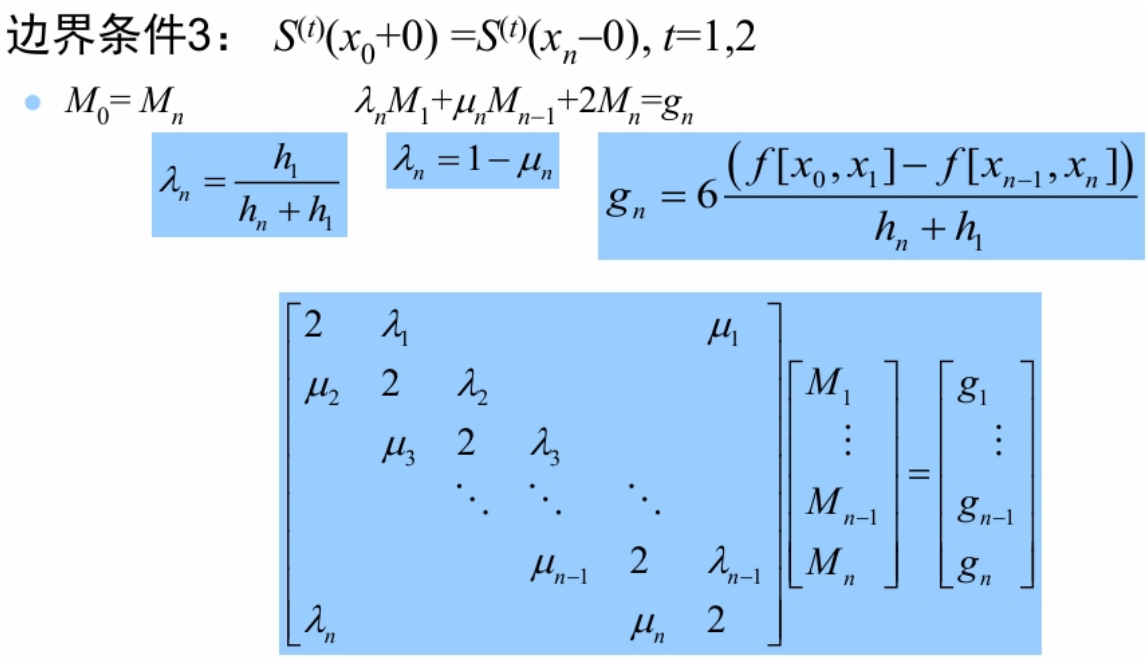
边界条件

$$\lambda_0=\mu_n=1 \qquad g_0=6\frac{(f[x_0,x_1]-f_0')}{h_1} \qquad g_n=6\frac{(f_n'-f[x_{n-1},x_n])}{h_n}$$

边界条件2



边界条件3



- 简单插值的次数与节点个数有关，n+1个节点上要用n次多项式来 插值，而样条插值多项式的次数与节点个数无关，便于在多个节 点上用低次插值。
- 一般的分段低次插值，每段上插值多项式不同，各段表达式之间 没有内在联系，而样条插值多项式各段之间有联系。
- 样条多项式往往只需已知m-1个边界节点上的导数值就够了，其 余节点上的导数值是自然形成的

插值法的缺陷：

- 由实验提供的数据带有测试误差，插值函数会保留数据的全部测 量误差
- 当插值函数的阶数较高时，曲线摆动很大，而求得的插值函数与 实验规律可能偏离甚远
- 实验数据往往很多，用插值法得到的近似表达式缺乏实用价值

4.2 拟合问题

4.2.1 曲线拟合

从一组实验测量数据中(xi ,yi) (i=1,...,m)找出实验规律的数学表达式，即求函数y=f(x)的 一个近似表达式y=ϕ(x)（经验公式）

构造一个能逼近列表数据的近似的数学表达式，使各数据点从总体上最贴近，而不一 定要求构造的函数曲线通过所给数据点

从几何上来看，就是通过给定的m个数据点(xi ,yi)，求曲线y=f(x) 的一条近似曲线y=ϕ(x)

最小二乘法

一般情况下，不要求近似曲线y=ϕ(x)严格地经过所有数据点 (xi ,yi)(i=1,2...,m)，即不要求拟合函数ϕ(x)在xi 处的偏差（残差）δi =ϕ(xi)−yi 严格地等于0。

为使近似曲线能尽量反映所给数据点的变化趋势，要求偏差δi 适当地小

- 偏差绝对值之和最小∑|δi |
- 最大偏差绝对值最小\max|δi
- 偏差平方和最小∑|δi 2|——最小二乘原则

对于给定数据(xi ,yi)，要求在某个函数类Φ中寻求一个函数ϕ*(x)，使

$$\sum_{i=1}^{m}[\varphi^*(x_i)-y_i]^2=\min_{\varphi(x)\in\Phi}\sum_{i=1}^{m}[\varphi(x_i)-y_i]^2$$

基本环节

确定ϕ(x)的形式：通常的作法是描绘出数据点(xi, yi)，然后根据这些点的分布情 况来选择ϕ(x)的形式。
求最小二乘解：求满足条件的近似函数ϕ*(x)。

设 $\varphi(x)=F(a_0,\ldots,a_n,x)\ (n<m)$

选择适当参数$a_k=a_k^*$，使相应的函数$\varphi^*(x)=F(a_0^*,\ldots,a_n^*,x)$满足最小二乘条件，即点$(a_0^*,\ldots,a_n^*)$是多元函数

$$S(a_0,\cdots,a_n)=\sum_{i=1}^{m}[F(a_0,\cdots,a_n,x_i)-y_i]^2$$

的极小点，从而$a_0^*,\ldots,a_n^*$满足方程组

$$\frac{\partial S}{\partial a_k}=0$$ 法方程组

若 $\varphi(x)=a_0\varphi_0(x)+a_1\varphi_1(x)+\cdots+a_n\varphi_n(x)$，相应的法方程组必是线性方程组

$$\varphi(x)=a_0\varphi_0(x)+a_1\varphi_1(x)+\cdots+a_n\varphi_n(x)$$

$$S(a_0,a_1,\ldots\ldots,a_n)=\sum_{i=1}^{m}[a_0\varphi_0(x_i)+a_1\varphi_1(x_i)+\ldots\ldots+a_n\varphi_n(x_i)-y_i]^2$$

$$\frac{\partial S}{\partial a_k}=0\Rightarrow\sum_{i=1}^{m}\varphi_k(x_i)[a_0\varphi_0(x_i)+a_1\varphi_1(x_i)+\ldots\ldots+a_n\varphi_n(x_i)-y_i]=0$$

$$(h,g)=\sum_{i=1}^{m}h(x_i)g(x_i)$$

$$a_0(\varphi_k,\varphi_0)+a_1(\varphi_k,\varphi_1)+\ldots\ldots+a_n(\varphi_k,\varphi_n)=(\varphi_k,f)\quad k=0,1,\cdots n$$

函数h,g的关于离散点列$\{x_i\}_{i=1}^m$的离散内积

$$a_0(\varphi_k,\varphi_0)+a_1(\varphi_k,\varphi_1)+\cdots\cdots+a_n(\varphi_k,\varphi_n)=(\varphi_k,f)$$
$$k=0,1,\cdots n$$

$$\begin{bmatrix}(\varphi_0,\varphi_0) & (\varphi_0,\varphi_1) & \cdots & (\varphi_0,\varphi_n)\\ (\varphi_1,\varphi_0) & (\varphi_1,\varphi_1) & \cdots & (\varphi_1,\varphi_n)\\ \vdots & \vdots & \ddots & \vdots\\ (\varphi_n,\varphi_0) & (\varphi_n,\varphi_1) & \cdots & (\varphi_n,\varphi_n)\end{bmatrix}\begin{bmatrix}a_0\\ a_1\\ \vdots\\ a_n\end{bmatrix}=\begin{bmatrix}(\varphi_0,f)\\ (\varphi_1,f)\\ \vdots\\ (\varphi_n,f)\end{bmatrix}$$

当$\varphi_0(x),\varphi_1(x),\ldots\varphi_n(x)$线性无关时，存在唯一解

最小二乘解的存在唯一性

对于给定的一组实验数据(x_i,y_i)(x_i互异；$i=1,2,\ldots,m$)，在函数类 $\Phi=\{\varphi_0(x),\varphi_1(x),\cdots,\varphi_n(x)\}$ ($n<m$且$\varphi_0(x),\varphi_1(x),\ldots,\varphi_n(x)$线性无关)中，存在唯一的函数 $\varphi^*(x)=a_0^*\varphi_0(x)+a_1^*\varphi_1(x)+\cdots+a_n^*\varphi_n(x)$ 使得

$$\sum_{i=1}^{m}[\varphi^*(x_i)-y_i]^2=\min_{\varphi(x)\in\Phi}\sum_{i=1}^{m}[\varphi(x_i)-y_i]^2$$

系数 a_k^* 可以通过解法方程组得到。

4.2.2 代数多项式拟合

$$\varphi_0(x)=1,\varphi_1(x)=x,\cdots,\varphi_n(x)=x^n$$

$$(\varphi_j,\varphi_k)=\sum_{i=1}^{m}x_i^{\,j}x_i^{\,k}=\sum_{i=1}^{m}x_i^{\,j+k}\qquad(j,k=0,1,\cdots,n)$$

$$(\varphi_k,f)=\sum_{i=1}^{m}x_i^{\,k}y_i\qquad(k=0,1,\cdots,n)$$

Ax=b

$$\begin{bmatrix}m & \sum x_i & \cdots & \sum x_i^{\,n}\\ \sum x_i & \sum x_i^{\,2} & \cdots & \sum x_i^{\,n+1}\\ \vdots & \vdots & \ddots & \vdots\\ \sum x_i^{\,n} & \sum x_i^{\,n+1} & \cdots & \sum x_i^{\,2n}\end{bmatrix}\begin{bmatrix}a_0\\ a_1\\ \vdots\\ a_n\end{bmatrix}=\begin{bmatrix}\sum y_i\\ \sum x_iy_i\\ \vdots\\ \sum x_i^{\,n}y_i\end{bmatrix}$$

当$m=n+1$时，所得的拟合多项式就是Newton或Lagrange插值多项式，对数据拟合效果并不好，对次数较高的多项式进行拟合，方程组的系数矩阵是病态的。

4

插值

$$\begin{bmatrix}1&2&4&8\\1&3&9&27\\1&4&16&64\\1&5&25&125\end{bmatrix}\begin{bmatrix}c_0\\c_1\\c_2\\c_3\end{bmatrix}=\begin{bmatrix}8\\27\\64\\125\end{bmatrix}$$

- 拟合

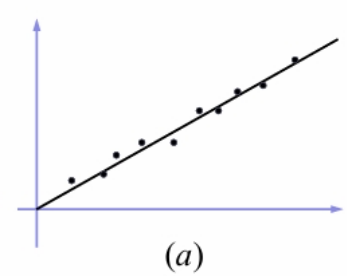
$$\begin{bmatrix}4&14&54&224\\14&54&224&978\\54&224&978&4424\\224&978&4424&20514\end{bmatrix}\begin{bmatrix}c_0\\c_1\\c_2\\c_3\end{bmatrix}=\begin{bmatrix}224\\978\\4424\\20514\end{bmatrix}$$

$$C=\begin{bmatrix}1&x_1&\cdots&x_1^{\,n}\\1&x_2&\cdots&x_2^{\,n}\\\vdots&\vdots&\ddots&\vdots\\1&x_m&\cdots&x_m^{\,n}\end{bmatrix}\qquad Y=\begin{bmatrix}y_1\\y_2\\\vdots\\y_m\end{bmatrix}$$

$$A=C^TC\qquad b=C^TY$$

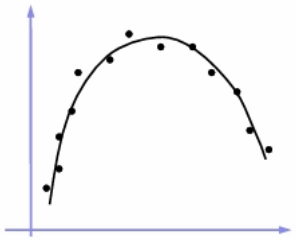
4.2.3 选择合适的拟合公式

在作数据拟合时，选择合适的拟合数学公式是很重要的。通常最好是选用所给数据预先 作出列表函数的曲线图，再根据曲线的大致形状，选择和确定合适的拟合数学公式形状， 一味采用多项式拟合并不一定可取。



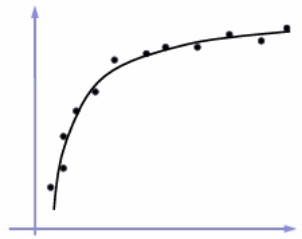
(a)

- 采用线性拟合



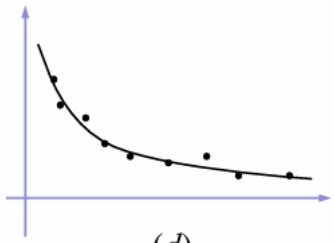
(b)

- 采用多项式拟合



(c)

- 采用双曲线型函数
  $\varphi(x)=x/(ax+b)$
- 或指数型函数
  $\varphi(x)=ae^{-b/x}$



(d)

- 采用
  - $\varphi(x)=1/(ax+b)$
  - $\varphi(x)=1/(a+bx^2)$
  - $\varphi(x)=ae^{-bx}$

  等函数拟合

已知数据，试用最小二乘法拟合x和y之间的经验公式

i	x_i	y_i	x_i^2	x_iy_i
1	36.9	181	1361.61	6678.9
2	46.7	197	2180.89	9199.9
3	63.7	235	4057.69	14969.5
4	77.8	270	6052.84	21006.0
5	84.0	283	7056.00	23772.0
6	87.5	292	7656.25	25550.0
Σ	396.6	1458	28365.28	101176.3

- 绘出散点图
- 确定曲线形式——线性$\varphi(x)=a+bx$
- 建立法方程组

$$\begin{bmatrix} 6 & \sum_{i=1}^{6} x_i \\ \sum_{i=1}^{6} x_i & \sum_{i=1}^{6} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{6} y_i \\ \sum_{i=1}^{6} x_i y_i \end{bmatrix}$$

$$a = 95.3524, b = 2.2337$$

- 结果检验
- 均方误差

$$\frac{1}{n}\Sigma\delta_i^2 = 4.445$$

- 均方根误差

$$\sqrt{\frac{1}{n}\Sigma\delta_i^2} = 2.108$$

- 平均绝对误差——误差绝对值的平均

如果认为误差不能满足要求，需要改变函数类型或增加实验数据等方法建立新的拟合曲线

i	x_i	y_i	$\hat{y}_i$	$\delta_i=\hat{y}_i-y_i$	δ_i^2	$\Sigma\delta_i^2$
1	36.9	181	177.78	-3.22	10.37	26.6704
2	46.7	197	199.67	2.67	7.13	
3	63.7	235	237.64	2.64	6.97	
4	77.8	270	269.13	-0.87	0.76	
5	84.0	283	282.98	-0.02	0.0004	
6	87.5	292	290.80	-1.20	1.44	

4.2.4 扩展内容

加权最小二乘

$$\sum_{i=1}^{m} W_i[\varphi^*(x_i)-y_i]^2 = \min_{\varphi(x)\in\Phi}\sum_{i=1}^{m} W_i[\varphi(x_i)-y_i]^2$$

利用正交函数作最小二乘拟合

$$(\varphi_k,\varphi_j)=\sum_{i=1}^{m} W_i\varphi_k(x_i)\varphi_j(x_i)=\begin{cases}0 & (k\neq j)\\ A_k>0 & (k=j)\end{cases}$$

拟合结果的置信区间

4.2.5 自带函数

函数	描述
polyfit	根据数据用多项式进行最小二乘拟合
interp1	一维插值（查表）
interp1q	快速一维线性插值
interp2	二维插值
interpn	n维插值
spline,csape	三次样条插值
ppval	分段多项式估计函数
cftool	Curve fitting Tool

4.3 总结

相同点：

- 从函数角度看，插值法与最小二乘法都是一种根据函数表求函数的近似 表达式的问题，属于函数逼近问题。
- 从几何上看，二者都是根据一列数据点求曲线的近似曲线问题，是曲线 拟合问题。

不同点：

- 插值法根据插值条件来选择近似函数；最小二乘法根据“偏差平方和最 小”原则选择近似函数。

五 积分和微分

5.1 数值微分

5.1.1 差商近似

5.1.1.1 三种差商

- 向前差商

$$f'(x_0)\approx\frac{f(x_0+h)-f(x_0)}{h}$$

$$R(x)=f'(x_0)-\frac{f(x_0+h)-f(x_0)}{h}=-\frac{h}{2!}f''(\xi)=O(h)$$

- 向后差商

$$f'(x_0)\approx\frac{f(x_0)-f(x_0-h)}{h}$$

$$R(x)=f'(x_0)-\frac{f(x_0)-f(x_0-h)}{h}=\frac{h}{2!}f''(\xi)=O(h)$$

- 中心差商

$$f'(x_0)\approx\frac{f(x_0+h)-f(x_0-h)}{2h}$$

$$R(x)=f'(x_0)-\frac{f(x_0+h)-f(x_0-h)}{2h}$$
$$=-\frac{h^2}{12}[f'''(\xi_1)+f'''(\xi_2)]=-\frac{h^2}{6}f'''(\xi)=O(h^2)$$

5.1.1.2 误差

h越小，误差越小，舍入误差越大。

事后估计：增加泰勒级数展开式的项数可以提高精度。

高精度微分公式：

$$f(x_{i+1})=f(x_i)+f'(x_i)h+\frac{f''(x_i)}{2}h^2+\cdots$$

$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{h}-\frac{f''(x_i)}{2}h+O(h^2)$$

$$f''(x_i)=\frac{f(x_{i+2})-2f(x_{i+1})+f(x_i)}{h^2}+O(h)$$

$$f'(x_i)=\frac{f(x_{i+1})-f(x_i)}{h}-\frac{f(x_{i+2})-2f(x_{i+1})+f(x_i)}{2h^2}h+O(h^2)$$

$$f'(x_i)=\frac{-f(x_{i+2})+4f(x_{i+1})-3f(x_i)}{2h}+O(h^2)$$

增加二阶导数项可以将精度提高到 $O(h^2)$

例：

- 例：利用有限差商估计函数

$$f(x)=-0.1x^4-0.15x^3-0.5x^2-0.25x+1.2$$

- 在x=0.5处的导数值（真值为$f'(0.5)$=-0.9125）
- 解：h=0.5，

i	x_i	$f(x_i)$
i-1	0	1.2
i	0.5	0.925
i+1	1.0	0.2

- 前向差商 $f'(0.5)=\frac{0.2-0.925}{0.5}=-1.45 \quad |\varepsilon_t|=58.9\%$
- 后向差商 $f'(0.5)=\frac{0.925-1.2}{0.5}=-0.55 \quad |\varepsilon_t|=39.7\%$

i	x_i	$f(x_i)$
i-1	0.25	1.10351563
i	0.5	0.925
i+1	0.75	0.63632813

- 中心差商 $f'(0.5)=\frac{0.2-1.2}{1}=-1.0 \quad |\varepsilon_t|=9.6\%$
- h=0.25 $f'(0.5)=-1.155 \quad |\varepsilon_t|=26.5\%$

$f'(0.5)=-0.714 \quad |\varepsilon_t|=21.7\%$

$f'(0.5)=-0.934 \quad |\varepsilon_t|=2.4\%$

- 取h=0.25，利用高精度微分公式

i	x_i	$f(x_i)$
i-2	0	1.2
i-1	0.25	1.10351563
i	0.5	0.925
i+1	0.75	0.63632813
i+2	1.0	0.2

- 前向差商 $f'(0.5)=-0.859375 \quad |\varepsilon_t|=5.82\%$
- 后向差商 $f'(0.5)=-0.878125 \quad |\varepsilon_t|=3.77\%$
- 中心差商 $f'(0.5)=-0.9125 \quad |\varepsilon_t|=0\%$

5.1.2 插值型数值微分

用插值函数的导数近似为原函数的导数

$$\left(x_j, f\left(x_j\right)\right)\left(j=0,1,2,\cdots,n\right)$$

$$f(x)=L_n(x)+R_n(x) \qquad f(x)=N_n(x)+R_n(x)$$

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

$$R_n^{(k)}(x)=\frac{d^k}{dx^k}\left[\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)\right]$$

$k=1$

$$R_n'(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}'(x)+\frac{\omega_{n+1}(x)}{(n+1)!}\frac{d}{dx}\left[f^{(n+1)}(\xi)\right]$$

ξ未知，当$x\neq x_i$时，无法利用上式估计误差，若求某个插值节点的导数值，有

$$f'(x_i)=P_n'(x_i)+\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}'(x_i)$$

特例：n=1

插值节点为x_0和x_1，若记$h=x_1-x_0$——一阶微分两点公式

$$f'(x_0)=\frac{f(x_1)-f(x_0)}{h}-\frac{h}{2}f''(\xi)$$ 在x_0处的向前差商

$$f'(x_1)=\frac{f(x_1)-f(x_0)}{h}+\frac{h}{2}f''(\xi)$$ 在x_1处的向后差商

n=2

$n=2$，插值节点为$x_k=x_0+kh(k=0,1,2)$——一阶微分三点公式

$$f'(x_0)=\frac{1}{2h}\left[-3f(x_0)+4f(x_1)-f(x_2)\right]+\frac{h^2}{3}f^{(3)}(\xi)$$ 三点向前差商

$$f'(x_1)=\frac{1}{2h}\left[-f(x_0)+f(x_2)\right]-\frac{h^2}{6}f^{(3)}(\xi)$$ 中心差商

$$f'(x_2)=\frac{1}{2h}\left[f(x_0)-4f(x_1)+3f(x_2)\right]+\frac{h^2}{3}f^{(3)}(\xi)$$ 三点向后差商

基于泰勒级数的公式等价于通过对数据点进行插值求微分

- 用三次样条插值的导数近似被插值函数的导数，效果相当好
- 在实际问题中使用哪种公式要视具体问题而定。有时三种公式都要用到，给定一个列表函数，对函数中间各点都可使用精度较高的中心差商公式，但起始点只能使用前差公式，终点则使用后差公式。
- 一般情况下，三点公式比两点公式准确，步长越小结果越准确。但当余项中的高阶导数无界或计算过程中的舍入误差超过截断误差时，这个结论不成立。

5.2 数值积分

定义数值积分是离散点上的函数值的线性组合

$$I=\int_a^b f(x)dx\approx\sum_{i=0}^{n}A_i f(x_i)=I_n(f)$$

$$I=I_n(f)+R_n(f)$$

求积公式的余项

称为求积系数，与$f(x)$无关，与积分区间和积分点有关

称为求积节点，与$f(x)$无关

- 如果一个求积公式对任何次数不超过m次的代数多项式都准确成立，而对于m+1次的代数多项式不能准确成立，即$R(f)\neq 0$，则称求积公式$I_n(f)$的代数精度是m
- 梯形公式的代数精度m=1：对不高于1次的代数多项式都准确成立，对2次以上的代数多项式存在误差。
- 一个求积公式的代数精度越高，就能对更多的被积函数准确或较准确地成立

用插值函数的积分，作为数值积分：

用插值函数的积分，作为数值积分

A_i

$$I_n(f)=\int_a^b L_n(x)dx=\int_a^b\sum_{i=0}^{n}l_i(x)f(x_i)dx=\sum_{i=0}^{n}\left(\int_a^b l_i(x)dx\right)f(x_i)$$

代数精度

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

$$I(f)-I_n(f)=\int_a^b R_n(x)dx=\int_a^b\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)dx$$

$f^{(n+1)}(x)=0, f(x)=x^k, k\le n$

至少n阶代数精度

5.2.1 Newton-Cotes积分

采用拉格朗日插值多项式P(x)来逼近f(x)，将积分区间n等分，取分点为求积节点，并作变量替换

$$h=\frac{b-a}{n}, x_i=a+ih, i=0,\cdots,n$$

$$x=a+th,$$

$$A_i=\int_a^b l_i(x)dx=\int_0^n\frac{t(t-1)\cdots(t-i+1)(t-i-1)\cdots(t-n)}{i!(n-i)!(-1)^{n-i}}hdt$$

$$=nh\cdot\frac{(-1)^{n-i}}{i!(n-i)!n}\int_0^n t(t-1)\cdots(t-i+1)(t-i-1)\cdots(t-n)dt$$

$b-a$

与步长h无关，可以预先求出 $C_i^{(n)}$ ——Cotes系数

$$A_i=(b-a)C_i^{(n)}$$

5.2.1.1 梯形公式(n=1)

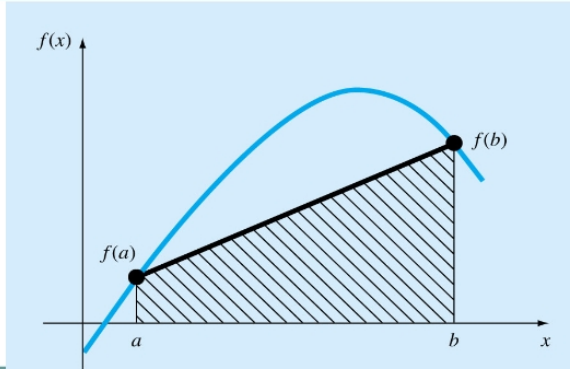
$n=1$

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

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

$$I_1(f) = (b-a)\frac{1}{2}f(a) + (b-a)\frac{1}{2}f(b)$$
$$= (b-a)\frac{f(a)+f(b)}{2}$$

$$E_1(f) = \int_a^b \frac{f''(\xi)}{2!}(x-a)(x-b)dx$$
$$= \frac{f''(\xi)}{2!}\int_a^b (x-a)(x-b)dx$$
$$= \frac{-(b-a)^3}{12}f''(\xi)$$



例：由于ξ无法计算，用平均值代替

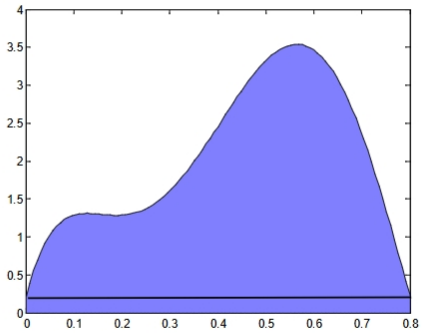
- 例：利用梯形公式计算函数在[0 0.8]上的积分（精确值为1.640533）
- 解：$f(0)=0.2$

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

$f(0.8)=0.232$

$$I_1 \cong (0.8-0)\frac{1}{2}(0.2+0.232) = 0.1728$$

$$E_t = 1.467733 \quad \varepsilon_t = 89.5\%$$



估计误差：用被积函数二阶导数的积分平均值代替$f''(\xi)$

$$f''(x) = \frac{\int_0^{0.8}(-400+4050x-10800x^2+8000x^3)dx}{0.8-0} = -60$$

$$E_a = -\frac{1}{12}(-60)(0.8)^3 = 2.56$$

5.2.1.2 Simpson公式(n=2)

$n=2$

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

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

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

$$E_2(f) = \int_a^b \frac{f^{(4)}(\xi)}{4!}(x-a)\left(x-\frac{a+b}{2}\right)^2(x-b)dx$$
$$= \frac{f^{(4)}(\xi)}{4!}\int_a^b (x-a)\left(x-\frac{a+b}{2}\right)^2(x-b)dx$$
$$= -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi) = -\frac{1}{90}h^5 f^{(4)}(\xi)$$

$$I_2(f) = (b-a)\frac{1}{6}f(a) + (b-a)\frac{4}{6}f\left(\frac{b+a}{2}\right) + (b-a)\frac{1}{6}f(b)$$
$$= \frac{h}{3}\left[f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right]$$

其中$h = \frac{b-a}{2}$，其中h的系数为1/3，称为辛普森1/3法则

例：

例：利用Simpson公式计算函数在[0 0.8]上的积分（精确值为1.640533）

解：$f(0)=0.2$ $f(x)=0.2+25x-200x^2+675x^3-900x^4+400x^5$

$f(0.4)=2.456$

$f(0.8)=0.232$

$$I\cong\frac{0.8}{6}(0.2+4(2.456)+0.232)=1.367467$$

$E_t=0.2730667$ $\varepsilon_t=16.6\%$

估计误差：$$E_a=-\frac{1}{2880}(-2400)(0.8)^5=0.2730667$$

用被积函数四阶导数的积分平均值代替$f^{(4)}(\xi)$，由于被积函数恰好是五次多项式，与真实误差一致

5.2.1.3 辛普森3/8法则(n=3)

构造区间上的四点三次Lagrange插值多项式，并逼近被积函数

$$I(f)=\frac{3h}{8}[f(x_0)+3f(x_1)+3f(x_2)+f(x_3)]$$
$$=(b-a)\frac{[f(x_0)+3f(x_1)+3f(x_2)+f(x_3)]}{8}$$

$$E(f)=-\frac{(b-a)^5}{6480}f^{(4)}(\xi)=-\frac{3}{80}h^5f^{(4)}(\xi)$$

***I*=(*b*-*a*)×平均高度**

具有与Simpson公式相同阶次的代数精度，比Simpson公式精确一点。
一般情况下，优先使用Simpson公式。

其中$h=\frac{b-a}{3}$

例：

$$f(x)=0.2+25x-200x^2+675x^3-900x^4+400x^5$$

- 子区间数目分别为3和5时，计算积分。
- 子区间数目为3时，利用Simpson 3/8 法则：

$f(0)=0.2;f(0.2667)=1.432724;f(0.5333)=3.487177;f(0.8)=0.232$

$$I\cong\frac{0.8}{8}(0.2+3(1.432724+3.487177)+0.232)=1.519170$$

$E_t=1.640533-1.519170=0.1213630$ $\varepsilon_t=7.4\%$

$$E_a=-\frac{1}{0.6480}(-2400)(0.8)^5=0.1213630$$

- 子区间数目为5时，结合Simpson 1/3 法则：

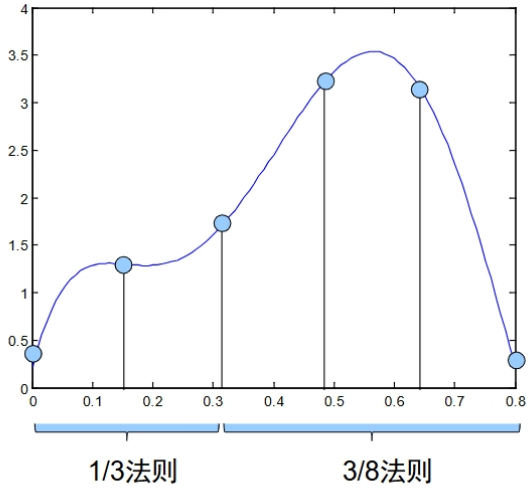
$f(0)=0.2; f(0.16)=1.296919;$

$f(0.32)=1.743393;$

$f(0.48)=3.186015;$

$f(0.64)=3.181929;$

$f(0.8)=0.232$



$I_1 \cong 0.3803237$ $I_2 \cong 1.264754$

$I = I_1 + I_2 = 1.645077$

$E_t = 1.640533 - 1.645077 = -0.00454383$ $\varepsilon_t = -0.28\%$

对于n大的，可以分成若干个两段+三段

n	$C_k^{(n)}$ k=0,…,n									
1	1	1								/2
2	1	4	1							/6
3	1	3	3	1						/8
4	7	32	12	32	7					/90
5	19	75	50	50	75	19				/288
6	41	216	27	272	27	216	41			/840
7	751	3577	1323	2989	2989	1323	3577	751		/17280
8	989	5888	-928	10496	-4540	10496	-928	5888	989	/28350

Cotes公式

5.2.1.4 数值积分的稳定性

$f(x)=1$，$\int_a^b f(x)dx = (b-a)\sum_{k=0}^{n} C_k^{(n)} y_k = b-a \Rightarrow \sum_{k=0}^{n} C_k^{(n)} = 1$

初始数据(x_k, y_k)→实际计算数据(x_k^*, y_k^*)

$\sum_{k=0}^{n} C_k^{(n)} y_k \approx \sum_{k=0}^{n} C_k^{(n)} y_k^*$

$$e = \left|\sum_{k=0}^{n} C_k^{(n)}(y_k - y_k^*)\right| = \left|\sum_{k=0}^{n} C_k^{(n)}(f(x_k) - f(x_k^*))\right| \le \sum_{k=0}^{n} |C_k^{(n)}| \cdot \max_k |f(x_k) - f(x_k^*)|$$

如果$C_k^{(n)}>0$， $\sum_{k=0}^{n} |C_k^{(n)}| = \sum_{k=0}^{n} C_k^{(n)} = 1$ $e \le \max_k |f(x_k) - f(x_k^*)|$

积分过程中计算结果的误差没有扩大，它不会超过$|f(x_k)-f(x_k^*)|$的最大误差，故求积过程是稳定的。

如果$C_k^{(n)}$有正有负，如表中$n>7$时，$\sum_{k=0}^{n} |C_k^{(n)}| > \sum_{k=0}^{n} C_k^{(n)} = 1$

若$C_k^{(n)}(f(x_k)-f(x_k^*))>0$，

$$e = \sum_{k=0}^{n} |C_k^{(n)}| \cdot |f(x_k) - f(x_k^*)| \ge \sum_{k=0}^{n} |C_k^{(n)}| \cdot \min_k |f(x_k) - f(x_k^*)|$$

初始数据的误差在求和过程中扩大，将导致计算的不稳定。此外，由插值的龙格现象可知，高阶Newton-Cotes积分不能保证等距数值积分序列的收敛性。因此，一般不采用高阶($n>7$)的Newton-Cotes求积公式。

余项：

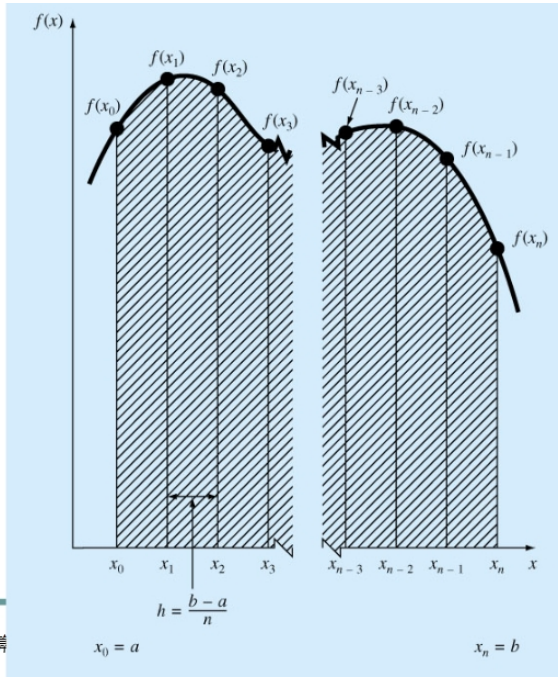
对偶数有$n+1$阶代数精度，而奇数为n阶代数精度

$$E_n(f)=\frac{K_n}{(n+2)!}f^{(n+2)}(\xi),K_n=\int_a^b\omega_{n+1}(x)dx<0,n=2k$$

$$E_n(f)=\frac{K_n}{(n+1)!}f^{(n+1)}(\xi),K_n=\int_a^b\omega_{n+1}(x)dx<0,n=2k-1$$

5.2.1.5 复合N-C公式

先将积分区间分成几个小区间，并在每个小区间上用低阶Newton-Cotes公式计算积分的近似值，然后对这些近似值求和，从而得到所求积分的近似值。



复合梯形公式

$$h=\frac{b-a}{n},x_i=a+ih,i=0,\cdots,n$$

$$f\in C^2[a,b]\Rightarrow\exists\xi\in[a,b],s.t.,\sum_{i=0}^{n-1}f''(\xi_i)=nf''(\xi)$$

$$\int_{x_i}^{x_{i+1}}f(x)dx=\frac{h}{2}(f(x_i)+f(x_{i+1}))-\frac{h^3}{12}f''(\xi_i)$$

$$T_n(f)=\sum_{i=0}^{n-1}\left\{\frac{h}{2}(f(x_i)+f(x_{i+1}))-\frac{h^3}{12}f''(\xi_i)\right\}$$

$$=h\left(\frac{1}{2}f(a)+\sum_{i=1}^{n-1}f(x_i)+\frac{1}{2}f(b)\right)-\sum_{i=0}^{n-1}\frac{h^3}{12}f''(\xi_i)$$

$$E_n(f)=-\frac{nh^3}{12}f''(\xi)=-\frac{h^2}{12}(b-a)f''(\xi)=-\frac{(b-a)^3}{12n^2}f''(\xi)$$

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

I=(b-a)×平均高度

例：

- 例：利用复合梯形公式计算函数在[0 0.8]上的积分（精确值为1.640533）
- 解：n=2

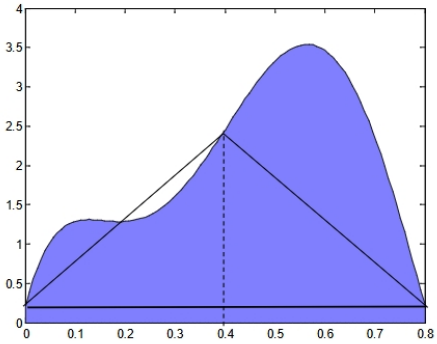
$$f(x)=0.2+25x-200x^2+675x^3-900x^4+400x^5$$

$f(0)$=0.2　$f(0.4)$=2.456　　$f(0.8)$=0.232

$$I_1 \cong 0.8\frac{(0.2+2(2.456)+0.232)}{2}=1.0688$$

$$E_t=0.57173 \qquad \varepsilon_t=34.9\%$$

$$E_a=-\frac{0.8^3}{12(2)^2}(-60)=0.64$$



例：计算 $I=\int_0^1 e^x dx$，要求至少有5位有效数字(即要求截断误差$\le 0.5\times10^{-4}$)，用复合梯形公式计算至少要在[0,1]区间划分多少个等分区间？

解：

$$f(x)=e^x \qquad f''(x)=e^x \qquad 0\le x\le 1$$

$$|E_T(f)|=\frac{b-a}{12}h^2 f''(\xi)=\frac{(b-a)^3}{12n^2}f''(\xi)\le\frac{1}{12n^2}e\le 0.5\times10^{-4}$$

$n=67.3$　至少要在[0,1]区间划分68个等分区间

复合辛普森公式

$$h=\frac{b-a}{n},x_i=a+ih,i=0,\cdots,n$$

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

$l=2m$ 偶数个子区间

$$\int_{x_{2i}}^{x_{2i+2}}f(x)dx=\frac{2h}{6}(f(x_{2i})+4f(x_{2i+1})+f(x_{2i+2}))-\frac{(2h)^5}{2880}f^{(4)}(\xi_i)$$

$$S_m(f)=\sum_{i=0}^{m-1}\left\{\frac{2h}{6}(f(x_{2i})+f(x_{2i+1})+f(x_{2i+2}))-\frac{(2h)^5}{2880}f^{(4)}(\xi_i)\right\}$$
$$=\frac{h}{3}\left(f(a)+4\sum_{i=0}^{m-1}f(x_{2i+1})+2\sum_{i=1}^{m-1}f(x_{2i})+f(b)\right)-\sum_{i=0}^{m-1}\frac{(2h)^5}{2880}f^{(4)}(\xi_i)$$

误差

`

$$E_m(f)=-\frac{(2h)^5m}{2880}f^{(4)}(\xi)$$
$$=-\frac{(b-a)^5}{2880m^4}f^{(4)}(\xi)$$
$$=-\frac{(b-a)^5}{180l^4}f^{(4)}(\xi)$$

收敛阶

定义：复合求积公式 $\int_a^b f(x)dx\approx I_n$

$$\lim_{h\to0}\frac{\int_a^b f(x)dx-I_n}{h^p}=C<\infty,C\neq0$$

则称I_n是p阶收敛的。

可以证明

$$\lim_{h\to0}\frac{\int_a^b f(x)dx-T_n}{h^2}=-\frac{1}{12}[f'(b)-f'(a)]$$

二阶收敛性

复合Simpson公式具有四阶收敛性。

收敛阶越高，近似值I_n收敛到真值的速度就越快，在相近的计算工作量下，有可能获得较精确的近似值。

复合辛普森收敛阶高，比复合梯形好用。

非等距积分

非等距梯形法则

$$f(x)=0.2+25x-200x^2+675x^3-900x^4+400x^5$$

Simpson法则

	x	h	f(x)	x	h	f(x)	
	0.0		0.200000	0.44	0.04	2.842985	Simpson 1/3法则
Simpson 1/3法则	0.12	0.12	1.309729	0.54	0.1	3.507297	
	0.22	0.1	1.305241	0.64	0.1	3.181929	
	0.32	0.1	1.743393	0.70	0.06	2.363000	
Simpson 3/8法则	0.36	0.04	2.074903	0.80	0.1	0.232000	
	0.40	0.04	2.456000				

事后估计与误差自适应

事后误差估计

- 复合梯形公式

$$I(f)-T_n(f)=-\frac{(b-a)}{12}h^2f''(\xi)$$ n等分区间

$$I(f)-T_{2n}(f)=-\frac{(b-a)}{12}\left(\frac{h}{2}\right)^2f''(\eta)$$ $2n$等分区间

- 近似有

$$f''(\eta)\approx f''(\xi)\Rightarrow I(f)-T_{2n}(f)\approx\frac{1}{3}\left(T_{2n}(f)-T_n(f)\right)$$

- 复合Simpson公式

$$I(f)-S_{2n}(f)\approx\frac{1}{15}\left(S_{2n}(f)-S_n(f)\right)$$

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

$$\frac{\left|T_{2n}(f)-T_n(f)\right|}{3}<\varepsilon$$

在区间逐步分半进行计算的过程中可以用前后两次计算结果来估计误差

递推公式

$$T_{2n}(f)=\frac{b-a}{4n}\left(f(a)+2\sum_{i=1}^{2n-1}f(a+i\frac{b-a}{2n})+f(b)\right)$$

当i取偶数时为n等分点，当i取奇数时为新增加的分点

$$\begin{aligned}T_{2n}(f)&=\frac{b-a}{4n}\left(f(a)+2\sum_{i=1}^{n-1}f(a+2i\frac{b-a}{2n})+2\sum_{i=1}^{n}f(a+(2i-1)\frac{b-a}{2n})+f(b)\right)\\&=\frac{b-a}{4n}\left(f(a)+2\sum_{i=1}^{n-1}f(a+2i\frac{b-a}{2n})+f(b)\right)+\frac{b-a}{2n}\sum_{i=1}^{n}f(a+(2i-1)\frac{b-a}{2n})\\&=\frac{1}{2}T_n+\frac{b-a}{2n}\sum_{i=1}^{n}f(a+(2i-1)\frac{b-a}{2n})\end{aligned}$$

5.2.1.6 重积分

在微积分中，二重积分的计算是用化为累次积分的方法进行的。计算二重数值积分也同样采用累次积分的计算过程。
简化起见，我们仅讨论矩形区域上的二重积分。
对非矩形区域的积分，大多可以变化为矩形区域上的累次积分。

$$\int_a^b\int_c^d f(x,y)dydx$$

$$\int_a^b\int_c^d f(x,y)dydx=\int_a^b\left(\int_c^d f(x,y)dy\right)dx=\int_c^d\left(\int_a^b f(x,y)dx\right)dy$$

5.2.2 龙贝格（Romberg）积分

外推算法：用若干个积分近似值来推算更精确的新的近似值的方法

验证可得 $$I(f)\approx T_{2n}(f)+\frac{1}{3}\left(T_{2n}(f)-T_n(f)\right)=S_n(f)$$

$$I(f)\approx S_{2n}(f)+\frac{1}{15}\left(S_{2n}(f)-S_n(f)\right)=C_n(f)$$

用低阶的公式组合后成为一个高阶的公式

$$R_n=C_{2n}+\frac{1}{63}(C_{2n}-C_n)=\frac{1}{63}(64C_{2n}-C_n)$$

龙贝格（Romberg）积分

Richardson外推法 $$I-G_{2n}=\frac{1}{4^m-1}(G_{2n}-G_n)\qquad m=1,2,3,\cdots$$

当m>4时上式趋近于0，再进行外推已无必要。故在实际中只作到R为止。

计算过程

计算过程

- 记号 $T_m^{(k)}$

积分区间的二分次数

近似值所在序列的性质，m=0在梯形序列中；m=1在Simpson序列中；m=2在Cotes序列中；m=3在Romberg序列中

$$T_0^{(0)}=\frac{b-a}{2}[f(a)+f(b)]$$

$$T_0^{(l)}=\frac{1}{2}T_0^{(l-1)}+\frac{b-a}{2^l}\sum_{i=1}^{2^{l-1}}f\left[a+(2i-1)\frac{b-a}{2^l}\right],l=1,2,3\cdots$$

$$T_m^{(k)}=\frac{4^mT_{m-1}^{(k+1)}-T_{m-1}^{(k)}}{4^m-1},k=0,1,2,\cdots;m=1,2,3,\cdots$$

① $T_1=T_0^{(0)}$ $<\varepsilon$?
② $T_2=T_0^{(1)}$ ③ $S_1=T_1^{(0)}$ $<\varepsilon$?
④ $T_4=T_0^{(2)}$ ⑤ $S_2=T_1^{(1)}$ ⑥ $C_1=T_2^{(0)}$ $<\varepsilon$?
⑦ $T_8=T_0^{(3)}$ ⑧ $S_4=T_1^{(2)}$ ⑨ $C_2=T_2^{(1)}$ ⑩ $R_1=T_3^{(0)}$
…… …… …… …… …… ……

$\left|T_m^{(0)}-T_{m-1}^{(0)}\right|<\varepsilon$

$\left|T_3^{(k-3)}-T_3^{(k-4)}\right|<\varepsilon$

或用相对误差判断

$O(h^2)$ $O(h^4)$ $O(h^6)$ $O(h^8)$

例：

计算 $\pi = \int_0^1 \frac{4}{1+x^2} dx$

解：

k	T_{2k}	S_{2k-1}	C_{2k-2}	R_{2k-3}
0	3			
1	3.1	3.133333		
2	3.131176	3.141569	3.142118	
3	3.138988	3.141593	3.141594	3.141586
4	3.140941	3.141592	3.141592	3.141593

数值微分的外推法

$$f(x \pm h) = f(x) \pm hf'(x) + \frac{h^2}{2!} f''(x) \pm \frac{h^3}{3!} f^{(3)}(x) + \cdots$$

$$f'(x) = T(h) \approx \frac{1}{2h}\left[f(x+h) - f(x-h)\right] = f'(x) + \frac{h^2}{3!} f^{(3)}(x) + \frac{h^4}{5!} f^{(5)}(x) + \cdots$$

$$f'(x) = T(h) - \frac{h^2}{3!} f^{(3)}(x) - \frac{h^4}{5!} f^{(5)}(x) - \cdots$$

作为$f'(x)$的近似值，$S(h)$比$T(h)$要更好。

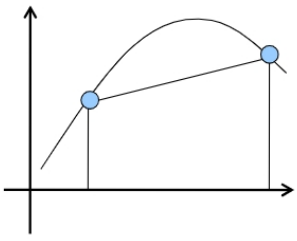
中点公式的误差与h^2成正比

$$\frac{f'(x) - T(h)}{f'(x) - T(2h)} \approx \frac{1}{4}$$

$$f'(x) \approx \frac{4}{3} T(h) - \frac{1}{3} T(2h) = S(h)$$

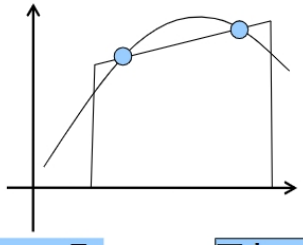
5.2.3 高斯积分公式

- 梯形公式
  - 限定节点为两个端点，代数精度为1
- 对节点不加限制，可以适当选取x_0，x_1，使得积分公式具有更高的代数精度。



$$\int_a^b f(x)dx = A_0 f(x_0) + A_1 f(x_1)$$

- 假设积分区间为[-1,1]，若要求代数精度为3



$$\begin{cases} A_0 \cdot 1 + A_1 \cdot 1 = \int_{-1}^{1} 1dx = 2 \\ A_0 \cdot x_0 + A_1 \cdot x_1 = \int_{-1}^{1} x dx = 0 \\ A_0 \cdot x_0^2 + A_1 \cdot x_1^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3} \\ A_0 \cdot x_0^3 + A_1 \cdot x_1^3 = \int_{-1}^{1} x^3 dx = 0 \end{cases} \Leftarrow \begin{cases} A_0 = A_1 = 1 \\ x_0 = -\frac{\sqrt{3}}{3} \\ x_1 = \frac{\sqrt{3}}{3} \end{cases}$$

$$\int_{-1}^{1} f dx = f(-\frac{\sqrt{3}}{3}) + f(\frac{\sqrt{3}}{3})$$

两点Gauss-Legendre公式

两点高斯公式

积分区间变换$[a,b]\rightarrow[-1,1]$

- 定义一个新变量x_d $x = a_0 + a_1 x_d$
- 下限$x=a$，对应于$x_d=-1$ $a = a_0 + a_1(-1)$
- 上限$x=b$，对应于$x_d=1$ $b = a_0 + a_1(1)$
- 可得 $a_0 = \frac{b+a}{2}$ $a_1 = \frac{b-a}{2}$

$$x = \frac{(b+a)+(b-a)x_d}{2} \Rightarrow dx = \frac{(b-a)}{2}dx_d$$

定理

- ***n*+1**个积分点的数值积分公式，最高**2*n*+1**阶

高斯求积公式 $\int_a^b f(x)dx = \sum_{k=0}^{n} A_k f(x_k)$

- 若代数精度达到**2*n*+1**，则称为**高斯求积公式**，相应的求积节点称为**高斯点**

构造

- 待定系数法，x_k，A_k，令$f(x)=1, x, x^2, \ldots, x^{2n+1}$使求积公式精确成立
- 首先利用$[a,b]$上的$n+1$次正交多项式确定高斯点，再利用高斯点确定求积系数
  - $[a,b]$上的$n+1$次正交多项式的零点是Gauss点

几个常用的高斯多项式

legendre

在区间[-1,1]上

$$\begin{cases} P_0(x) = 1 \\ P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n] \quad (n = 1,2,\ldots) \end{cases}$$

$$P_0(x) = 1$$
$$P_1(x) = x$$
$$P_2(x) = \frac{1}{2}(3x^2-1)$$
$$P_3(x) = \frac{1}{2}(5x^3-3x)$$
$$P_4(x) = \frac{1}{8}(35x^4-30x^2+3)$$

$$\begin{cases} P_0(x) = 1, \quad P_1(x) = x \\ P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x) \quad (n = 1,2,3,\ldots..) \end{cases}$$

Gauss-Legendre求积公式

$$A_k=\frac{2(1-x_k^2)}{[(1+n)P_{n+1}(x_k)]^2}$$

$$E_t=\frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3}f^{(2n+2)}(\xi)\quad \xi\in[-1,1]$$

n=2

$$\int_{-1}^{1}f(x)dx=f(-\frac{1}{\sqrt{3}})+f(\frac{1}{\sqrt{3}})+\frac{1}{135}f^{(4)}(\xi)$$

n=3

$$\int_{-1}^{1}f(x)dx=\frac{5}{9}f(-\frac{\sqrt{15}}{5})+\frac{8}{9}f(0)+\frac{5}{9}f(\frac{\sqrt{15}}{5})+\frac{1}{15750}f^{(6)}(\xi)$$

求积公式节点系数表

n	x_k	A_k
1	0	2
2	±0.5773502692	1
3	±0.7745966692 0	0.5555555556 0.8888888889
4	±0.8611363116 ±0.3399810436	0.3478548451 0.6521451549
5	±0.9061798459 ±0.5384693101 0	0.2369268851 0.4786286705 0.5688888889

n	x_k	A_k
6	±0.9324695142 ±0.6612093865 ±0.2386191861	0.1713244924 0.3607615730 0.4679139346
7	±0.9491079123 ±0.7415311856 ±0.4058451514 0	0.1294849662 0.2797053915 0.3818300505 0.4179591837
8	±0.9602898565 ±0.7966664774 ±0.5255324099 ±0.1834346425	0.1012285363 0.2223810345 0.3137066459 0.3626837834

Hermite

$$\int_{-\infty}^{+\infty}e^{-x^2}f(x)dx\approx\sum_{k=0}^{n}A_kf(x_k)$$

n	x_k	A_k
2	±0.7071067811	0.8862269254
3	±1.2247448713 0	0.2954089751 1.8163590006
4	±0.5246476232 ±1.6506801238	0.8049140900 0.0813128354
5	±0.9585724646 ±2.0201828704 0	0.3936193231 0.0199532421 0.9453087204

n	x_k	A_k
6	±0.4360774119 ±1.3358490704 ±2.3506049736	0.7246295952 0.1570673203 0.0045300099
7	±0.8162878828 ±1.6735516287 ±2.6519613563 0	0.4256072526 0.0545155828 0.0009717812 0.8102646175

优缺点

优点：计算量小，精度高

缺点：n改变大小时，节点和系数几乎都改变；利用余项控制精度困难；需要计算节点处的函数值，不适用于函数表达式未知的情况

较多采用复合求积的方法，将积分区间分成m个等长的小区间，在每个小区间上使用同一低阶Gauss求积公式算出积分的近似值，再相加

5.3 MATLAB自带函数

函数	描述
quad	Simpson公式
trapz	梯形公式
diff	数值微分（连续函数求导）
quadl	Lobatto求积（一种高斯求积公式，取代了quad8）
sum	求和
dblquad	二重积分
integral	使用全局自适应积分
integral2	二重积分
integral3	三重积分

方法	需要的节点数	复合公式需要的节点数	精度	应用范围	编程难度	备注
梯形公式	2	$n+1$	$h^3f''(\xi)$	广泛	容易	
Simpson1/3法则	3	$2n+1$	$h^5f^{(4)}(\xi)$	广泛	容易	
Simpson3/8法则	4	≥3		广泛	容易	
高阶Newton-Cotes公式	≥5			很少	容易	
Romberg积分	3			要求已知$f(x)$表达式	容易	不适于列表型数据
Gauss求积公式	≥2			要求已知$f(x)$表达式	容易	不适于列表型数据