计算物理
—方程组的数值解法

田付阳

数理学院物理系
北京科技大学

March 26, 2018

Outline of Topics

1. Last lecture
2. Linear equation sets
    1. Elimination method
    2. Iterative method
3. Nolinear equation sets
    1. Nonlinear equation sets
    2. Contradictory Equation sets
4. Numerical Algorithm
5. Homework

数值近似

EOS: BM, Morse

四个参数，E_0, V_0, B_0, and B_0'

$$P(V) = \frac{3}{2}B_0[(\frac{V_0}{V})^{\frac{7}{3}} - \frac{V_0}{V})^{\frac{5}{3}}]\{1 + \frac{3}{4}(B_0' - 4)[(\frac{V_0}{V})^{\frac{2}{3}} - 1]\}$$

$$E(V) = D_0 e^{-\gamma(\frac{V}{V_0}-1)} - 2D_0 e^{-\frac{\gamma}{2}(\frac{V}{V_0}-1)} + E_0. \quad (1)$$

γ is related to bulk modulus B.

数值近似计算

- 插值方法：多项式；三次样条；
- 拟合方法：基于最小二乘法，线性拟合及非性线拟。

线性方程组

n阶线性方程组的一般形式为：

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

或写成矩阵形式：

$$Ax = b$$

式中，A称为常数矩阵，x称为解向量，b称为右端常向量，分别为：

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}.$$

$$x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}},\ b = (b_1, b_2, \cdot, b_n)^{\mathrm{T}}.$$

在$detA \neq 0$ (A非奇异矩阵)，由克拉默法则，方程组有唯一解

$$x_i = D_i/D$$

其中$D = detA$, D_i表示为第i列换成b后所得的行列式. 一个n阶行列式，结果有$n!$项，第一项又是n个数的乘积，所以一个n阶行列式乘法运算次数为$(n-1)n!$,共有$N = (n+1)(n-1)n!$+n; n=20, $N = 9.7 * 10^{20}$.

直接法和迭代法

1. 直接法：根据矩阵方程的初等变换性质，将方程组变换成上(或下)三角方程组或对角方程组来求解，如高斯消元法、LU分析，三对角矩阵。

2. 迭代法：用某种极限过程逐步逼近线性方程组精确解的方法。优点，便于编制计算程序，但存在迭代收敛性和收敛快慢的问题，且只能满足一定精度的近似解。

高斯消元法

利用线性方程组的初等变换方法通过将一个方程乘以或除以某个常数，以及将两个方程相加减，将方程组变换成上(或下)三角方程组或对角方程组来求解，通常由“消元”和“回代”两个过程组成。

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

首先，由第一式消去第二、第三式中的x_1,得$a_{21}/a_{11} \to m_{21}$, $a_{31}/a_{11} \to m_{31}$,

continued

$$\begin{cases} a_{11}^{(1)}x_1 & + & a_{12}^{(1)}x_2 & + & a_{13}^{(1)}x_3 = b_1^{(1)} \\ & + & a_{22}^{(2)}x_2 & + & a_{23}^{(2)}x_3 = b_2^{(2)} \\ & + & a_{32}^{(2)}x_2 & + & a_{33}^{(2)}x_3 = b_3^{(2)} \end{cases}$$

其中，

$$a_{22}^{(2)} = (a_{22} - m_{21}a_{12})^{(1)}, a_{23}^{(2)} = (a_{23} - m_{21}a_{13})^{(1)},$$

$$a_{32}^{(2)} = (a_{32} - m_{31}a_{12})^{(1)}, a_{33}^{(2)} = (a_{33} - m_{31}a_{13})^{(1)},$$

$$b_2^{(2)} = (b_2 - m_{21}b_1)^{(1)}, b_3^{(2)} = (b_3 - m_{31}b_1)^{(1)},$$

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 = b_1^{(1)} \\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 = b_2^{(2)} \\ a_{33}^{(3)}x_3 = b_3^{(3)} \end{cases}$$

其中，$a_{33}^{(3)} = a_{33}^{(2)} - m_{32}a_{23}^{(2)}$, $b_3^{(3)} = a_3^{(2)} - m_{32}b_2^{(2)}$,
这样先出x_3,回代，求出x_2，再求出x_1.
总结：对于n阶代数方程组，递推出消元与回代的步骤如下：

1. 消元过程：

$$m_{ik} \leftarrow \frac{a_{ik}}{a_{kk}}, a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}^{(k)}a_{kj}^{(k)},$$

$$b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}, k = 1, 2, \cdots, n; i, j = k+1, \cdots, n$$

2. 回代过程：

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}, x_i = \frac{1}{a_{ii}^{(i)}}(b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j)$$

$$i = n-1, \cdots, 2, 1$$

在第k步消元时，x_k的系数$a_{kk}^{(k)}$称为第k步的主元素。在求解线性方程组时要求其系数矩阵是非奇异的，但若出现主元素$a_{kk}^{(k)} = 0$,消去过程无法进行或者即使$a_{kk}^{(k)} \neq 0$,但如果绝对值很小，会出现用小数作除数导致舍入误差扩大，严重影响计算结果精度的问题。解决方案：在消元前调整方程的次序, *i.e.* 将绝对值大的元素换到主对角线的位置。

高斯消元法算法

1. 输入数据a_{ij}, b_i;
2. 消元：do $k = 1, n$, 选主元

$$
\begin{aligned}
a_{kj} &\Leftarrow a_{kj}/a_{kk}, j = k+1, \cdots, n \\
b_k &\Leftarrow b_k/a_{kk} \\
a_{ij} &\Leftarrow a_{ij} - a_{ik} \cdot a_{kj}, i, j = k+1, \cdots, n \\
b_i &\Leftarrow b_i - a_{ik} \cdot b_k, i = k+1, \cdots, n
\end{aligned}
$$

3. 回代：$x_n \Leftarrow b_n/a_n$

$$
x_i \Leftarrow \frac{1}{a_{ii}}(b_i - \sum_{j=i+1}^{n} a_{ij} b_j), \quad i = n-1, \cdots, 1
$$

LU分解法

高斯消元法是将系数矩阵化成上三角矩阵后，采用回代，很容易得到方程的解。
对于一般的n阶线性方程组$Ax = b$,可以将矩阵A分解成一个下三角矩阵L和上三角矩阵U积的形式$A = LU$,于是方程组$Ax = b$化成

$$Ax = LUx = L(Ux) = Ly = b, Ly = b, Ux = y$$

由此，可以由下三角方程组$Ly = b$求得y列解，然后由上三角方程组Ux=y求得问题解x.

杜利特尔分解法

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}=\begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}$$

$$\times\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix}.$$

根据矩阵乘法规则，有关系$u_{1i}=a_{1i}, i=1,2,\cdots,n, l_{i1}=a_{i1}/u_{11}, i=2,3,\cdots,n.$

首先将U的第一行和L的第一列求出来。根据LU矩阵元特点，推导其他矩阵元间关系

$$a_{ri} = \sum_{k=1}^{n} l_{rk} u_{ki} = \sum_{k=1}^{r} l_{rk} u_{ki} = u_{ri} + \sum_{k=1}^{r-1} l_{rk} u_{ki}$$

得到

$$u_{ri} = a_{ri} - \sum_{k=1}^{r-1} l_{rk} u_{ki}, r = r, r+1, \cdots, n$$

$$a_{ir} = \sum_{k=1}^{n} l_{ik} u_{kr} = \sum_{k=1}^{r} l_{ik} u_{kr} = l_{ir} u_{rr} + \sum_{k=1}^{r-1} l_{ik} u_{kr}$$

得到

$$l_{ir} = \frac{1}{u_{rr}}(a_{ir} - \sum_{k=1}^{r-1} l_{ik} u_{kr}), r = r+1, r+2, \cdots, n.$$

LU杜利特尔分解求解$Ax = b$的算法

1. 计算$u_{1i} = a_{1i}, i = 1, 2, \cdots, n;$
   $l_{i1} = a_{i1}/u_{11}, i = 2, 3, \cdots, n;$
2. 对于$r = 2, 3, \cdots, n$, 由上式计算

$$u_{ri}, i = r, r+1, \cdots, n; l_{ri}, i = r+1, r+2, \cdots, n;$$

3. 计算中间解。求解下三角矩阵方程$Ly = b$

$$y_1 = b_1, y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k, i = 2, 3, \cdots, n$$

4. 计算最终解。求解上三角矩阵方程$Ux = y$

$$x_n = y_n/u_{nn}, x_i = (y_i - \sum_{k=i+1}^{n} u_{ik} x_k)/u_{ii}, i = n-1, \cdots, 2, 1.$$

三对角矩阵追赶法

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

其递推形式为

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = f_i, i = 2, 3, \cdots, n-1$$

$$b_1 x_1 + c_1 x_2 = f_1$$

$$a_n x_{n-1} + b_n x_n = f_n$$

令

$$x_{i-1} + e_{i-1} x_i = d_{i-1}, i = 2, 3, \cdots, n$$

或

$$x_i + e_i x_{i+1} = d_i, i = 1, 2, 3, \cdots, n-1$$

得

$$x_i + \frac{c_i}{b_i - a_i e_{i-1}} x_{i+1} = \frac{f_i - a_i d_{i-1}}{b_i - a_i e_{e-1}}$$

$$e_i = \frac{c_i}{b_i - a_i e_{i-1}}, \quad d_i = \frac{f_i - a_i d_{i-1}}{b_i - a_i e_{i-1}}, i = 2, \cdots, n-1$$

$$e_1 = \frac{c_1}{b_1}, \quad d_1 = \frac{f_1}{b_1}$$

$$x_{n-1} + e_{n-1} x_n = d_{n-1}$$

$$x_n = \frac{f_n - a_n d_{n-1}}{b_n - a_n e_{n-1}} = " d_n "$$

迭代法

n阶线性方程组的一般形式为：

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

或写成矩阵形式：

$$Ax = b$$

式中，A称为常数矩阵，x称为解向量，b称为右端常向量.

迭代法

$$\begin{cases} x_1 = (b_1 - a_{12}x_2 - \cdots - a_{1n}x_n)/a_{11} \\ x_2 = (b_2 - a_{21}x_1 - \cdots - a_{2n}x_n)/b_{22} \\ \cdots\cdots\cdots \\ x_n = (b_n - a_{n1}x_1 - \cdots - a_{nn-1}x_{n-1})/b_{nn} \end{cases}$$

若设

$$g_{ij} = \begin{cases} -a_{ij}/a_{ii} \quad , \quad i,j = 1,2,\cdots,n, & i \neq j \\ 0 & i = j \end{cases}$$

$$d_i = b_i/a_{ii}, i = 1,2,\cdots,n.$$

$$x_i = \sum_{j=1}^{n} g_{ij}x_j + d_i, i = 1,2,\cdots,n$$

$$x_i^{(k+1)} = \sum_{j=1}^{n} g_{ij}x_j^{(k)} + d_i, i = 1,2,\cdots,n.$$

当满足关系$\max|x_i^{(k+1)} - x_i^k| < \varepsilon$时，迭代终止，$\varepsilon$是要求解的精度，此迭代方法称为雅可比(Jacobi)迭代法。

1. 选主元
2. 变换为雅可比迭代格式
3. 选$x^{(0)} = (0,0,0)^T$,求出计算结果

由于计算$x_i^{(k+1)}$时，$x_{i-1}^{(k+1)}$, $x_{i-2}^{(k+1)}$, $\cdots$,已经计算出来，对于收敛的迭代过程，$x_{i-1}^{(k)}$, $x_{i-2}^{(k)}$要比$x_{i-1}^{(k+1)}$, $x_{i-2}^{(k+1)}$更接近精确值，实际迭代可以表示为

$$x_i^{(k+1)} = d_i + \sum_{j=1}^{i-1} g_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} g_{ij} x_j^{(k)}, i = 1, 2, \cdots, n$$

实例

用高斯-赛德尔迭代法求解下列四阶方程组：

$$\begin{cases} 7x_1 + 2x_2 + x_3 - 2x_4 = 4 \\ 9x_1 + 15x_2 + 3x_3 - 2x_4 = 7 \\ -2x_1 - 2x_2 + 11x_3 + 5x_4 = -1 \\ x_1 + 3x_2 + 2x_3 + 13x_4 = 0 \end{cases}$$

要求的精度：$\varepsilon = 0.000001$

- 需要选主元，以保证迭代过程是收敛的
- 收敛态慢，为了加速收敛，常采用松弛法，即将第k与k+1步加权平均，作为新的迭代结果.

二分法

对于工程中遇到的非线性方程，特别是根本不存在解析解的超越方程的数值求解问题，首先要确定方程解或根所在的区间。常采用图示法或函数分析的方法大体确定根所在的位置，再采用逐步逼近方法得到满足一定精度的近似解.

在方程的根或零点附近，通常$f(x)$要改变符号。如果$f(x)$在区间$[x_l, x_u]$是连续的实函数，并且$f(x_l)$和$f(x_u)$ 有相反的符号，即$f(x_l) \cdot f(x_u) \leq 0$, 在区间$[x_l, x_u]$内至少有一个实根。对于多根的情况，一般采用增量搜寻的方法来确定函数变号的区间，使得每一个区间只含一个根。

二分法：求方程根最简单的一种方法，其基本思想是将方程根所在区间$[x_l, x_u]$平分为两个小区间，再判断根在哪个小区间，然后将有根的小区间再一分为二，重复上面过程，直到有根区间小于解的精度要求为止。

误差估计

对于n步迭代，计算了$a_0, b_0, c_0, \cdots, a_n, b_n, c_n$,(这里的$c_i$为根的$(a_i+b_i)/2$), 得到根$r$的误差为

$$|r - c_n| \leq \frac{b_n - a_n}{2}$$

而$b_n - a_n = \frac{b_{n-1}-a_{n-1}}{2} = \cdots = \frac{b_0-a_0}{2^n}$,所以误差为

$$|r - c_n| \leq \frac{b_n - a_n}{2} = \frac{b_0 - a_0}{2^{n+1}}$$

由此如果要求误差ε,可得迭代次数为

$$n > \frac{log(b_0 - a_0) - log\varepsilon}{log2} - 1$$

bisect.m

```
function root = bisect(fun,a,b,eps)
n = 1+round((log(b-a)-log(eps))/log(2))
fa = feval(fun,a);
fb = feval(fun,b);
for i = 1:n
c = (b+a)/2; fc = feval(fun,c);
if fc * fa < 0
b=c; fb = fc;
else
a=c; fa = fc;
end  end
root = c;
```

例：用二分法计算非线性方程$e^x ln(x) - x^2 = 0$在区间(1,2)内的根。

弦截法(secant method)

也称为线性插值法,用连接点$(x_l, f(x_l))$和$(x_u, f(x_u))$的弦与x轴的交点作为试探根，弦方程为

$$\frac{0 - f(x_l)}{x_r - x_l} = \frac{f(x_u) - 0}{x_u - x_r}$$

试探根为

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

对于$(x_l, f(x_l))$和$(x_u, f(x_u))$,哪个是有根区间，即如果$f(x_l) \cdot f(x_r) > 0$,将$x_r \Rightarrow x_l$,否则$x_r = x_u$,重复计算上式，当相继两次计算的x_r之差满足一定精度时，得到所要求的解。要求是有两个点迭代初值。

secant.m

```
function [root n] = secant(f,x0,x1,eps,nmax)
dx = x1-x0; n = 0;
while((abs(dx)>=eps) and (n<=nmax))
x2 = x1-f(x1)*(x1-x0)/(f(x1)-f(x0));
if(f(x0)*f(x2)>0)
dx = x2 -x0;
x0 = x2;
else
dx = x2 - x1;
x1 = x2 ;
end
n=n+1; end root = x2;
```

例：用弦截法计算方程$e^x ln(x) - x^2 = 0$在区间[1,2]的根。

不动点迭代法

设给定一个非线性方程$f(x)=0$,当采用迭代方法求根时，可以先将其转换成等价方程

$$x=g(x),$$

接着构造如下的迭代格式

$$x_{k+1}=g(x_k), k=0,1,2,\cdots .$$

对于给定的迭代初始值x_0,若由此生成的迭代序列x_k有极限，记为

$$\lim_{k\to\infty} x_k = x^*, k=0,1,2,\cdots$$

则称x^*是方程的解。$g(x)$称为迭代函数。

迭代函数收敛的条件

设迭代方程$x = g(x)$, x^*是解$(x^* = g(x^*))$,迭代格式为$x_{n+1} = g(x_n)$,则有关系

$$x_{n+1} - x^* = g(x_n) - g(x^*) = \frac{g(x_n) - g(x^*)}{x_n - x^*}(x_n - x^*)$$

i.e.

$$e_{n+1} = \frac{g(x_n) - g(x^*)}{x_n - x^*} e_n$$

由数学中值定理，在x^*和x_n之间存在ξ_n,有关系$g'(\xi_n) = \frac{g(x_n)-g(x^*)}{x_n-x^*}$,则

$$|\frac{e_{n+1}}{e_n}| = |g'(\xi_n)|$$

当$|g'(\xi_n)| < 1$时，$|e_{n+1}| < |e_n|$,迭代误差减小，迭代收敛。所以，设$g(x)$在区间[a,b]上有连续的一阶导数，并满足

1. $a \leq g(x) \leq b$;
2. $|g'(x)| < 1$, 即$y_2 = g(x)$曲线斜率的绝对值小于$y_1 = x$的斜率。

得，

1. 函数$g(x)$在区间[a,b]上存在唯一的不动点，即方程的根；
2. 在区间[a,b]上任取迭代初值x_0,得到的序列$\{x_k\}$收敛到方程的根x^* (也可以采用图示法，即做两个曲线，迭代点向两曲线交点靠近，迭代收敛)

牛顿迭代法(切线法)

在方程近似解x_k附近，将非线性方程线性化，用近似线性方程的根(切线方程与x轴的交点)作为非线性方程的近似根。设非线性函数$f(x)$是连续可微的，x^*是方程$f(x)$的实根，x_k是迭代过程中的近似根，将$f(x)$在x_k展开成Taylor级数，

$$f(x) = f(x_k) + (x - x_k)f'(x_k) + (x - x_k)^2 \frac{f''(x_k)}{2!} + \cdots = 0$$

取其线性部分作为非线性方程$f(x) = 0$的近似方程，即

$$f(x_k) + (x - x_k)f'(x_k) = 0$$

若$f'(x_k) \neq 0$,则线性方程解的迭代格式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

牛顿迭代法(切线法)的几何意义在迭代过程中，用函数$f(x)$过点$(x_k, f(x_k))$的切线

$$y = f(x_k) + (x - x_k)f'(x_k)$$

作为$f(x)$的近似，并以此切线与x轴的交点作为新的迭代点x_{k+1}.

牛顿迭代法实际上是取迭代函数$g(x) = x - f(x)/f'(x)$, $|g'(x)| = |f(x)f''(x)/f'^2(x)|$，因为要求$f'(x) \neq 0$,只要是在根的附近迭代求解，通常有$f(x) \approx 0$,总能满足$|g'(x) \ll 1|$条件，所以牛顿迭代法通常收敛速度很快，是非线性方向求根的常用方法。

在其计算过程中，每一步都要计算导数值$f'(x_k)$,通常计算量很大或者导函数很复杂。为了克服这些困难，同时利用牛顿法收敛快的特点，可以用x_k, x_{k-1}两点处的差商代替$f'(x_k)$,

$$f'(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

代入

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

得

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$$

Examples

用牛顿迭代法求下列函数的根

$$f(x) = x^3 + 2x^2 + 3x - 1 = 0$$

```
format long;
f = inline('exp(x)*log(x)-x*x');
x0 =1.0; x1 = 2.0; eps = 0.000001;
nmax = 100; n = 0;
while((abs(x1-x0)>=eps) and (n <=nmax))
x2 = x1-f(x1)*(x1-x0)/(f(x1)-f(x0));
x0=x1;x1=x2;n=n+1;
end
x2
n
```

非线性方程组

对于多变量的非线性方程组

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0 \\ f_2(x_1, x_2, \cdots, x_n) = 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_n(x_1, x_2, \cdots, x_n) = 0 \end{cases}$$

常用的两种求解方法：一种是牛顿迭代法，另一种是求函数极小值方法。

牛顿迭代法

对于简单的方程组

$$\begin{cases} f_1(x,y)=0 \\ f_2(x,y)=0 \end{cases}$$

设(x_k,y_k)是试探解，在(x_k,y_k)的邻域作泰勒展开，保留线性项

$$\begin{cases} f_1(x_{k+1},y_{k+1}) = f_1(x_k,y_k)+(\frac{\partial f_1}{\partial x})_k(x_{k+1}-x_k)+ \\ \qquad\qquad (\frac{\partial f_1}{\partial y})_k(y_{k+1}-y_k)=0 \\ f_2(x_{k+1},y_{k+1}) = f_2(x_k,y_k)+(\frac{\partial f_2}{\partial x})_k(x_{k+1}-x_k)+ \\ \qquad\qquad (\frac{\partial f_2}{\partial y})_k(y_{k+1}-y_k)=0 \end{cases}$$

令

$$A^{(k)} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix}$$

$$B^{(k)} = \begin{pmatrix} -f_1 \\ -f_2 \end{pmatrix}$$

$$\delta^k = \begin{pmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{pmatrix} = \begin{pmatrix} \delta_x^{(k)} \\ \delta_y^{(k)} \end{pmatrix}$$

得到关系

$$A^{(k)} \bullet \delta^{(k)} = B^{(k)}$$

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} \delta_x^{(k)} \\ \delta_y^{(k)} \end{pmatrix}$$

计算步骤如下

1. 给定初始近似根和允许误差(x_0, y_0), ε_1, ε_2，并假定已得到第k次近似根(x_k, y_k);
2. 计算$A^{(k)}$, $B^{(k)}$;
3. 计算$s_1 = |f_1^{(k)}| + |f_2^{(k)}|$,若$s_1 < \varepsilon_1$,则计算结束，$(x_k, y_k)$作为满足精度要求的近似解，否则，执行(4);
4. 求解代数方程组$A^{(k)} \bullet \delta^{(k)} = B^{(k)}$,得到$(\delta_x^{(k)}, \delta_y^{(k)})$;
5. 计算上式，及$s_2 = |\delta_x^{(k)}| + |\delta_y^{(k)}|$,若$s_2 < \varepsilon_2$,则计算结束，$(x_{k+1}, y_{k+1})$作为满足精度要求的近似解，否则，将$(x_{k+1}, y_{k+1}) \to (x_k, y_k) \Rightarrow (2)$.

最速下降法

求函数极小值最常用的方法之一. 以两变量方程组为例,

$$\begin{cases} f_1(x,y) = 0 \\ f_2(x,y) = 0 \end{cases}$$

首先构造一个指标函数

$$F(x,y) = f_1^2 + f_2^2$$

通过求F的极小值点来得到方程的解。

continued

首先从某一点(x_0,y_0)出发，沿F下降的方向逐步接近F的零极小值。通常梯度的反方向是最陡下降方向。函数$F(x,y)$在(x,y)点的梯度方向就是该点等值线过切点的法线方向，梯度为

$$G = \nabla F = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y})$$

设(x_0,y_0)是方程组的一个近似解，计算在该点的梯度值

$$G_0 = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y})|_{(x_0,y_0)} = (G_{0x}, G_{0y})$$

从(x_0,y_0)出发，沿负梯度$-G_0$方向前进一适当步长得新点

$$\begin{cases} x_1 = x_0 - \lambda G_{0x} \\ y_1 = y_0 - \lambda G_{0y} \end{cases}$$

如何选择λ,才能使新点(x_1,y_1)是$F(x,y)$在$-G_0$方向上的相对极小值点。将$F(x_1,y_1)$在(x_0,y_0)附近展开，略去λ的二阶以上项

$$\begin{aligned} F(x_1,y_1) &= F(x_0,y_0)+(\frac{\partial F}{\partial x})_0(x_1-x_0)+(\frac{\partial F}{\partial y})_0(y_1-y_0) \\ &= F(x_0,y_0)-\lambda G_0^2=0 \end{aligned}$$

得到

$$\lambda=\frac{F(x_0,y_0)}{G_0^2}$$

推广到n个联立方程组

$$f_i(x_1,x_2,\cdots,x_n)=0, i=1,2,\cdots,n$$

定义指标函数

$$F(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}f_i^2$$

梯度下降法的迭代公式为

$$x_i^{(k+1)} = x_i^{(k)} - \lambda^{(k)} \frac{\partial F^{(k)}}{\partial x_i^{(k)}}, \quad \lambda^{(k)} = \frac{F^{(k)}}{\sum_{j=1}^{n} \left(\frac{\partial F^{(k)}}{\partial x_j^{(k)}} \right)^2}.$$

计算步骤如下：

1. 选取试探解$(x_1^{(0)}, x_2^{(2)}, \cdots, x_n^{(0)})$,假设已经计算到第$k$步，得$(x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)})$

2. 计算指标函数

$$F^{(k)} = F(x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)});$$

3. 若$F^{(k)} < eps$，则认为$(x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)})$为所求的解，否则计算偏导数

$$\frac{\partial F^{(k)}}{\partial x_i^{(k)}} \approx \frac{F(\cdots, x_i^{(k)} + h_i^{(k)}, \cdots) - F(\cdots, x_i^{(k)}, \cdots)}{h_i^{(k)}}.$$

$i = 1, 2, \cdots, n$

4. 再计算新的迭代值.

```
function [ x n f ] = dsense(x0)
eps=1.0e-6;
x = x0;
[ f f2 g g2] = myfun(x);
lamda = f2/g2;
x = x - lamda*g;
n =0;
while(norm(x-x0)>=eps)
x0 = x; n = n+1;
[ f f2 g g2 ] = myfun(x);
lamda = f2/g2;
x = x - lamda*g;
if(n>100000)
disp('迭代次数太多，可能不收敛！');
return;
end  end  end
```

矛盾方程组的解法：最小二乘法

也称为超定方程，方程数超过变量数，其解法采用函数求极值的优化方法，常采用最小二乘法，其基本要点是构造指标函数.

设$f_i(x_1, x_2, \cdots, x_m) = 0, i = 1, 2, \cdots, n, n > m$

$$F(x_1, x_2, \cdots, x_m) = \sum_{i=1}^{n} [f_i(x_i, x_2, \cdots, x_m)]^2$$

然后取极值，得线性方程组

$$\frac{\partial F}{\partial x_i} = 0, i = 1, 2, \cdots, m,$$

求解这m个线性方程组的解即矛盾方程组的解.

总结

1. 线性方程组的数据解法
2. 非线性方程的数据解法
3. 矛盾方程组的数值解法

算法重在设计

1. 算法设计关系科学计算设计的成败
2. 算法设计追求简单与统一
3. Zeno悖论的启示：一个人不管跑得多快，也永远追不上爬在前面的一只乌龟。每追赶一步所归结出的是同样类型的追赶问题，这种追赶过程永远不会终结。但这样的论断从算法设计的角度来看却极为精辟。
   - 3.1 追的计算：令乌龟不动，计算人追上乌龟所费的时间，$\Delta t_k = s_{k-1}/v_A$
   - 3.2 赶的计算：再令人不动，计算乌龟在这段时间内爬行的路程：$s_k = v_B \Delta t_k$

定义s_k为追赶问题的规模，则$s_k = v_B/v_A * s_{k-1}$

Continued

问题的规模被压缩了v_B/v_A倍，由于乌龟的速度v_B远远小于人的速度v_A,压缩系数v_B/v_A很小，所以这项计算的逼近效果极为显著。

$$\begin{cases} s_k = \frac{v_B}{v_A} s_{k-1} \\ s_0 = s \end{cases}$$

由于s_0是已知，经过有限步后，s_k可以忽略不计，从而得出人追上乌龟实际所花费的时间t_k.

追的计算和赶的计算都是简单的行程计算，Zeno算法的设计思想是：将人与乌龟的追赶化归为简单的行程计算的重复.

直接法的缩减技术

不考虑舍入误差情况下，通过有限步计算可以直接得出问题的精确解。

- 数列求和的累加算法: 将多项求和化归为两项求和的重复$S = a_0 + a_1 + \cdots + a_n$
- 缩减技术的设计机理：规模缩减技术,1. 结构递归: 每一步将所考察的计算模型加工成同样类型的计算模型，这类计算具有明晰的递归结构；2. 规模递减：加工成同一类型后，规模已经被压缩，压缩系数愈小，算法的效率愈高.
- 中国秦九韶算法(国外Horner算法)
$P = a_0x^n + a_1x^{n-1} + \cdots + a_n = \sum_k^n a_k x^{n-k}$

迭代法的校正技术

1. Zeno悖论中的"Zeno钟"
2. 开方算法
3. 校正技术的设计机理：将复杂计算化归为一系列简单计算的重复，以简御繁，逐步求精。其特点为逼近性和简单性。

迭代优化的超松弛技术

- Zeno算法的升华
- 设计机理
- 刘徽“割圆术”

递推加速的二分技术

- 结绳记数的快速算法
- 设计机理：每一步使问题的规模减半，即其规模等比级数(公比为1/2)递减，直到规模变为1时终止计算。

避免误差的危害

$$|x - x^*| \leq \varepsilon$$

ε称为近似值x^*的误差限或精度。

作业

1. 用LU分解法给出下列代数方程组$(2x_1 + 2x_2 + 3x_3 = 3, 4x_1 + 7x_2 + 7x_3 = 1, -2x_1 + 4x_2 + 5x_3 = -7)$的解及L,U矩阵;
2. 已知代数方程组$x_1 + 5x_2 - 3x_3 = 2; 5x_1 - 2x_2 + x_3 = 4; 2x_1 + x_2 - 5x_3 = -11$ (1). 写出该方程组的高斯-赛德尔迭代法的迭代格式; (2). 选$x^{(0)} = (0, 0, 0)^T$,求出$x^{(2)}$;写出相应的SOR迭代法的迭代格式(取$w = 1.5$);
3. 求解三阶三对角方程组;

$$\begin{pmatrix} 1 & 2 & \\ 2 & 2 & 3 \\ & 3 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

4. 用牛顿迭代法计算方程组$y\cos(xy) + 1 = 0$ and $\sin(xy)+x-y=0$在$x=1,y=2$附近的根；求解非线性方程组$x^2 - 2x - y + 0.5 = 0$ and $x^2 + 4y^2 - 4 = 0$在$x^{(0)},y^{(0)} = (0, 1)$附近的根。