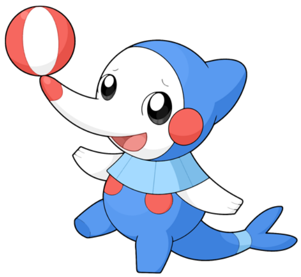
数值计算之美

SHU ZHI JI SUAN ZHI MEI

胡家威

http://hujiaweibujidao.github.io/



清华大学逸夫图书馆 · 北京

内容简介

本书是我对数值计算中的若干常见的重要算法及其应用的总结，内容还算比较完整。

本人才疏学浅，再加上时间和精力有限，所以本书不会详细介绍很多的概念，需要读者有一定的基础或者有其他的参考书籍，这里推荐参考文献中的几本关于数值计算的教材。

本书只会简单介绍下算法的原理，对于每个算法都会附上我阅读过的较好的参考资料以及算法的实现 (Matlab 或者其他语言)，大部分代码是来源于参考文献 [1] 或者是经过我改编而成的，肯定都是可以直接使用的，需要注意的是由于 Latex 对代码的排版问题，导致中文注释中的英文字符经常出现错位，对于这种情况请读者自行分析，不便之处还望谅解。写下这些内容的目的是让自己理解地更加深刻些，顺便能够作为自己的 HandBook，如有错误之处，还请您指正，本人邮箱地址是：hujiawei090807@gmail.com。

目　　录

第一章 线性方程组求解

1.1 高斯消去法

对于线性方程组 $Ax = b(A \in \mathbf{R}^{n*n}, b \in \mathbf{R}^n)$，一般的解法有高斯消去法 (Gauss Elimination) 和高斯 -若当消去法 (Gauss-Jordan Elimination，本书不讨论，详情请看参考文献 [2, 3])。

高斯消去法分为两个过程：第一步是前向消去过程 (forward elimination)，也就是将系数矩阵化成上三角矩阵的过程；第二步是回代过程 (back substitution)，也就是自底向上的求解方程组的解的过程。

需要考虑的是：(1) 主元可能为 0；(2) 主元可能相对于其他行的主元位置元素小很多，这两种情况可能会导致结果出现很大的误差，具体的例子可以看参考文献 [1, 3]，所以一般情况下，使用高斯消去法要进行选主元的操作，选主元的方式有四种 (可以查看参考文献 [1] 中的 NCM 程序包中的 $lugui$ 程序源码)，一般是使用部分选主元 (事实证明：*采用部分选主元的高斯消去法，可以保证得到相对较小的残差*)，选择主元所在列中绝对值最大的元素 (但有些时候为了简便可以就选择主元位置非 0 的那一行即可，参考文献 [3](P79) 中给出了例子)，然后将该行和主元行进行交换，再进行消去过程。

该算法很简单，所以直接附上实现源码 (Matlab)，改编自文献 [3]

code/gausseliminationnm.m

```
function x = gausselimination(A,b)
% 选主元的高斯消去法
% A=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
% b=[12; -6.5; 16; 17];
% x=gausselimination(A,b)

ab=[A,b];%系数矩阵和右端项组成的矩阵
[r,c]=size(ab);%行数，列数
for k=1:r-1 %消去过程
    %寻找主元列中对角线下面绝对值最大的元素
    [rm,im]=max(abs(ab(k:r,k)));%当前主元所在列的最大项和索引
    im=im+k-1;%修正为正确的行索引因为上面返回的是相对于对角线的索引号,
    %如果对应元素不是0 如果是的话那么[0就是奇异阵了A]
    if(ab(im,k) ~= 0)
        if(im ~= k)
            ab([k im],:)=ab([im k],:);%进行行交换
        end
    end
    %计算乘子更新主元行以下的行
    for i=k+1:r
        %注意是当前行元素减去乘子乘以主元行对应元素
        ab(i,k:c)=ab(i,k:c)-ab(i,k)/ab(k,k)*ab(k,k:c);
    end
end
%回代计算解
x=zeros(r,1);%解的列向量
x(r)=ab(r,c)/ab(r,r);%最后一个解
for i=r-1:-1:1 %自底向上回代
    %右端项减去已经求出的部分然后除以它的系数
    x(i)=(ab(i,c)-ab(i,i+1:r)*x(i+1:r))/ab(i,i);
end
```

1.2 LU 分解

线性方程组的实际应用中经常遇到系数矩阵不变，只是右端项发生变化的情况 (多右端项问题)，这个时候如果还是使用高斯消去法的话，对于每个右端项都要进行重复的消去和回代的过程，显然计算量很大。为了解决这类问题，于是就有了 LU 分解算法。

关系式 $LU = PA$ 即为矩阵 A 的 LU 分解 (或者三角分解)，其中矩阵 P 为排列阵 (单位阵经过行列交换得到的矩阵)，矩阵 L 为单位下三角矩阵 (对角线元素都为 1)，矩阵 U 为上三角矩阵，这样的话，一般的线性方程组 $Ax = b$ 可以等价为两个三角型线性方程组 $Ly = Pb, Ux = y$。首先对系数矩阵 A 进行 LU 分解，然后对于每个右端项只要先计算出 y 然后再计算 x 即可，两个都只是解很简单的三角型线性方程组。

LU 分解有两种方法，一种是使用前面的高斯消去法，另一种是 Crout 方法 (本书不介绍，详情请看参考文献 [3])。

下面给出使用高斯消去法的 LU 分解算法源码 [大部分内容和高斯消去法相同，只是它还要计算矩阵 L, U, P]，来源于 [1] 中的 $lutx$ 程序，它是 LU 分解常用的 K-I-J 版本。

code/lugsnm.m

```
function [L,U,p] = lugs(A)
% 使用高斯消去法的分解LU [ k−i−j 版本 ]
% 系数矩阵，单位下三角矩阵，上三角矩阵，排列阵
% A=[4 −2 −3 6; −6 7 6.5 −6; 1 7.5 6.25 5.5; −12 22 15.5 −1];
% b=[12; −6.5; 16; 17];
% [L,U,P]=lugs(A)
% A(P,:)
% L*U

[n,n] = size(A);
p = (1:n)';
for k = 1:n−1
    %查找主元列对角线以下的绝对值最大的元素和索引
    [r,m] = max(abs(A(k:n,k)));
    m = m+k−1;
    %如果对应元素是则跳过消去过程0
    if (A(m,k) ~= 0)
        %满足条件的话就交换行
        if (m ~= k)
            A([k m],:) = A([m k],:);
            p([k m]) = p([m k]);%对应的排列阵也要跟着变化
        end
        %计算乘子
        i = k+1:n;
        A(i,k) = A(i,k)/A(k,k);
        %更新矩阵的行
        j = k+1:n;
        A(i,j) = A(i,j) − A(i,k)*A(k,j);%就地存储!
    end
end
%得到分解的结果LU
L = tril(A,−1) + eye(n,n);%下三角部分，但是对角线上都是1
U = triu(A);%上三角部分
```

1.3 Cholesky 分解

对于对称矩阵 A，如果它的各阶顺序主子式 $\neq 0$，则它可以唯一分解为 $A = LDL^T$，其中 D 是对角阵，L 为单位下三角阵。若矩阵同时正定，那么存在实的非奇异的下三角矩阵 L，满足 $A = LL^T$，若限定 L 对角线元素 > 0，那么此分解唯一。

考虑对称正定矩阵 A，它的 LU 分解过程可以进一步简化，这也就是 Cholesky 分解！事实证明，对于对称正定矩阵，*Cholesky* 分解是稳定的，不需要进行选主元操作。另外，它的计算量和存储量都只是 LU 分解的一半。考虑到它的对称性，LU 分解的结果如下：(注意矩阵 L 不再是单位下三角矩阵了)

$$A = \begin{bmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} & \dots & l_{1n} \\ 0 & l_{22} & l_{23} & \dots & l_{2n} \\ 0 & 0 & l_{33} & \dots & l_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & l_{nn} \end{bmatrix}$$

一种就地存储的 Cholesky 分解算法的实现步骤 (对 $j = 2, 3 \cdots, n$ 重复执行 (3)(4) 步即可)：

(1) 求 l_{11}。$l_{11} = \sqrt{a_{11}}$

(2) 求 l_{i1}。$l_{i1} = a_{i1}/l_{11}$

(3) 假设矩阵 L 的前 $j-1$ 列都已经求出了，即 $l_{ik}(k \leq j-1, i = 1, \cdots, n)$ 都已知，考虑计算 $a_{i,j}$。

$$a_{jj} = \sum_{k=1}^{j} l_{ik}^2 \quad => \quad l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} {l_{jk}}^2}$$

(4) 求矩阵 L 中第 j 列剩余元素，即 $l_{ij}(i > j)$，考虑计算 a_{ij}。

$$a_{jj} = \sum_{k=1}^{n} l_{ik} l_{jk} = \sum_{k=1}^{j} l_{ik} l_{jk} = \sum_{k=1}^{j-1} l_{ik} l_{jk} + l_{ij} l_{jj} \quad => \quad l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk})/l_{jj}$$

code/cholnm.m

```
function [L,U] = cholnm(A)
% 简易版本的分解，就地存储，使用下三角Cholesky
% p=pascal(5)
% [l,u]=cholnm(p)
% l*u

[n,n] = size(A);
for j=1:n %当前操作的列
    for k=1:j-1
        A(j,j)=A(j,j)-A(j,k)^2;
    end
    A(j,j)=sqrt(A(j,j));%求出对角线元素
    for i=j+1:n %求解对角线以下的元素
        for k=1:j-1
            A(i,j)=A(i,j)-A(i,k)*A(j,k);
        end
        A(i,j)=A(i,j)/A(j,j);%求出对角线以下的元素
    end
end
%求出上三角和下三角
L=tril(A);
U=L';
```

1.4 矩阵求逆的方法

一般情况下，求矩阵的逆矩阵都是计算量非常大的，所以一般对于线性方程组 $Ax = b$ 不是去求 A^{-1} 来得到解的。下面给出 Matlab 中反斜线求逆的简易版本，来源于参考文献 [1] 中 NCM 程序包中的 *bslashtx* 程序。它首先判断是否是特殊的三种矩阵，如果是的话直接采用最优的算法求解，如果不是那就先进行 LU 分解再求解。

code/bslashnm.m

```
function x = bslashnm(A,b)
% 解线性方程组
% A=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
% b=[12; -6.5; 16; 17];
% A\b

[n,n] = size(A);
if isequal(triu(A,1),zeros(n,n))
    %下三角直接前向回代即可
    x = forward(A,b);
    return
elseif isequal(tril(A,-1),zeros(n,n))
    %上三角直接后向回代即可
    x = backsubs(A,b);
    return
elseif isequal(A,A')
    [R,fail] = chol(A);%进行分解Cholesky
    if ~fail
        %如果分解成功，求解两个三角型线性方程组即可
        y = forward(R',b);
        x = backsubs(R,y);
        return
    end
end
```

```
%三角分解
[L,U,p] = lugs(A);
%对右端项进行排列然后解下三角线性方程组
y = forward(L,b(p));
%解上三角线性方程组
x = backsubs(U,y);

% 前向消去———— 解下三角型线性方程组————
% For lower triangular L, x = forward(L,b) solves L*x = b.
function x = forward(L,x)
[n,n] = size(L);
x(1) = x(1)/L(1,1);
for k = 2:n
    j = 1:k-1;
    x(k) = (x(k) - L(k,j)*x(j))/L(k,k);
end

% 回代———— 解上三角型线性方程组——————
% For upper triangular U, x = backsubs(U,b) solves U*x = b.
function x = backsubs(U,x)
[n,n] = size(U);
x(n) = x(n)/U(n,n);
for k = n-1:-1:1
    j = k+1:n;
    x(k) = (x(k) - U(k,j)*x(j))/U(k,k);
end
```

1.5 Matlab 函数解析

Matlab 内置函数

(1)LU 分解 lu

$[L, U, P] = lu(A)$

(2)Cholesky 分解 $chol$

$[R, p] = chol(A)$

参考文献

[1] Numerical Computing with Matlab. Cleve B. Moler.
中文翻译版本《Matlab 数值计算》，喻文健，机械工业出版社，2006，6
网站资源：
(1)Cleve Moler 撰写的教科书
(2)数值计算交互演示网站

[2] 数值分析与算法，喻文健，清华大学出版社，2012，1

[3] Numerical Methods: An introduction with Applications Using Matlab. Amos Gilat. Vish Subramaniam，2010，10

[4] Data Mining Algorithms In R.
网站资源:
WikiBook：Data Mining Algorithms In R