武漢理工大學
WUHAN UNIVERSITY OF TECHNOLOGY

MATLAB 数值计算

方法与实践

第二章 线性方程组求解

主讲老师：余文昭

yuwenzhao1989@qq.com

2018/3/29

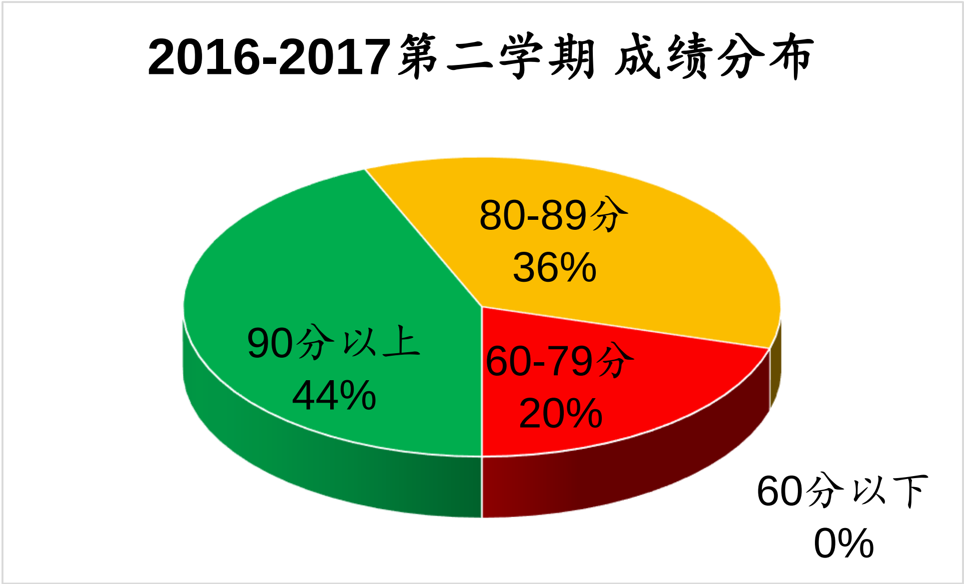
课程简介

- MATLAB数值计算方法与实践，1学分，1-8周
- 考试方式： 出勤（20%）、作业完成（80%）

课后作业（电子版本）

课后作业汇总（纸质版本）

30%　　70%



- 不及格：
  - 缺勤超过4次
  - 多次督促均未交作业
- 态度

课程简介（2）

- 课程内容

1. Matlab简介，基本语法及运算
2. 线性方程组求解（直接法、间接法）
3. 插值与拟合
4. 数值积分（※）

课程简介（2）

❿ 参考书目

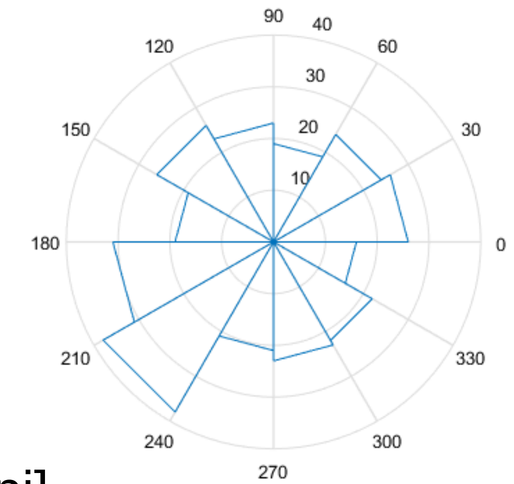
- ［美］John H.Mathews，Kurtis D. Fink著，《数值方法（Matlab版）》，电子工业出版社，2010。
- ［美］Shoichiro Nakamura著，《科学计算引论-基于MATLAB的数值分析》，电子工业出版社，2002。
- ［美］ Cleve B.Moler著 张志勇等编译，《MATLAB数值计算（2013修订版中译本）》

第一次课 Q & A

- 安装相关问题
  - 建议完全安装
  - 如空间不足，可勾选需要用的Toolbox安装，后期需要再添加
  - ……
- 程序或命令行中使用中文字符——报错
- function关键字
  - 用于定义函数
  - 不可用于命令行中
  - 分清所写代码是函数中的语句，还是脚本文件中的语句（函数中的代码一般不会再命令行中输入）
  - function .m文件中非全局变量均为该函数局部变量，在command window中直接运行该函数时，workspace里面无法直接查看
  - 全局变量定义： global 变量名1，变量名2 ……

武漢理工大學
WUHAN UNIVERSITY OF TECHNOLOGY

第一、二次课 Q & A



- 绘图相关
  - ezplot（符号函数绘图）
    - ezplot(y(x)) 默认[-2pi,2pi]
    - ezplot(x,y(x)) (参数函数形式) 默认[0,2pi]
    - 实际应用中较少使用
  - rose（极坐标形式的直方图）
    - rose(theta) 统计theta中的分布，默认20条
    - rose(theta,x) 按照向量x统计theta的分布（条数=length(x)）
    - 统计与角度相关的数据
- 作业中需要附上源代码
- 请所有同学重新补交第一次的作业
  - 已经上交的可以重新修改再提交
  - 补选的同学请根据第一章的内容自学补交
  - 时间：3月21日（下周二）晚上10点前交给各专业负责人
  - 负责人汇总后发送到邮箱（PPT首页）
  - 纳入平时成绩考虑（占30%）

本章主要内容

- 线性方程组的直接解法
- 线性方程组的迭代解法
- Matlab内置函数求解线性方程组

MATLAB 线性方程组直接解法

相关定义

- 线性方程组**直接解法**：将线性方程组转换为便于求解的三角线性方程组，再求三角线性方程组的数值解。
  - 理论上直接法可在有限步内运算得到方程组的精确解，但是由于舍入误差的存在，实际计算中得到的也是近似解；
  - 计算复杂度较高，适用于阶数比较低的线性方程组的求解
- 线性方程组**迭代解法**：把求解线性方程组的问题转化为构造一个无穷序列，并用这个序列去逐次逼近满足一定误差要求的方程组的数值解。
  - 适合于高阶线性方程组的求解

线性方程组的表示形式

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

记$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$

则方程组可写为矩阵形式$Ax = b$

A为系数矩阵，b为自由列矩阵，x待求未知向量。

线性方程组解的结构

- 齐次线性方程组（$Ax = 0$）
  - rank(A) < n，则存在非零解
  - rank(A) = n，只有零解

回顾！

- 非齐次线性方程组（$Ax = b$）
  - 增广矩阵B=[A b]
  - （1）恰定方程组，即rank(A) = rank(B)=n，存在唯一解向量
  - （2）欠定方程组，即rank(A) = rank(B)<n，存在无穷个解向量
  - （3）超定方程组，即rank(A) < rank(B)，一般意义下无解

Matlab求解齐次方程组

- 零空间：对于齐次方程组$Ax=0$，如果rank(A)<n，把满足方程组的全体解向量x称为矩阵A的零空间。
- Matlab矩阵零空间的求解：

$$\begin{cases} x+2y+2z+w=0 \\ 2x+y-2z-2w=0 \\ x-y-4z-3w=0 \end{cases}$$

```
function [x,r] = HomoEqu
A = [1 2 2 1; 2 1 -2 -2; 1 -1 -4 -3];
r = rank(A);
x = null(A);
```

方程组基础解系含有
n-r = 2个基向量

```
r =    2
x =
  -0.7072    0.1256
   0.6526    0.1360
  -0.2173   -0.5913
  -0.1636    0.7849
```

- 方程组的全部解向量为：

$$s = k_1 x(:,1) + k_2 x(:,2)$$

- 验证解的正确性

```
k1 = 2; k2 = -3;
b = A*(k1*x(:,1)+k2*x(:,2));
```

```
b =
  1.0e-14 *
   0.1332
   0.4441
   0.3553
```

非齐次方程组直接解法

- Gram法
- 高斯消元法
- 三角矩阵法

Gram法

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

$Ax = b$：当且仅当 $\det(A) \neq 0$ 时，有唯一的解，而且解为：

$$x_i = \frac{D_i}{D}, D = \det(A), D_i = \det\begin{pmatrix} a_{11} & \cdots & a_{1i-1} & b_1 & a_{1i+1} & \cdots & a_{1n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & \cdots & a_{ni-1} & b_n & a_{ni+1} & \cdots & a_{nn} \end{pmatrix}$$

Gram法（2）

- 在实际运算中，当矩阵的维数较高时，计算行列式是非常困难的。
- 计算行列式的计算复杂度随维数的增长非常快，对于一个n*n的矩阵，用初等的方法计算其行列式，需要的计算时间是O(n!)。
- 因此，克莱姆法则在实际计算中并未被采用。

❿ 课堂练习（3分钟）：

给定行列式函数det(X)编写Gram法求解线性方程组解的程序，其中：

```
A =

    64    -3    -1
     2   -90     1
     1     1    40

b =

    14
    -5
    20
```

```
A = [64 -3 -1; 2 -90 1; 1 1 40];
b = [14; -5; 20];

x = zeros(3,1);
detA = det(A);

for iDim = 1:size(A,2)
    B = A;
    B(:, iDim) = b;
    x(iDim) = det(B)/detA;
end
```

高斯消元法

- 基本思路：首先将A化为上三角阵，再回代求解 。

$$
\begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\
a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn} & b_n
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\
0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\
0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)}
\end{pmatrix}
$$

第一步：

$$\text{第1行}\times\frac{-a_{i1}}{a_{11}}+\text{第}i\text{行}, i=2,\cdots,n$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \Rightarrow \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix}$$

高斯消元法的缺陷

- 若主对角线上某元素为0则无法进行消元
- 若主对角线上某元素绝对值接近0，则用其作除数，会导致其他元素数量级的严重增长和舍入误差的扩散，将严重影响计算结果的精度。
- 例：$\begin{cases} 10^{-9}x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$

$$m_{21} = a_{21}/a_{11} = 10^9$$

$$a_{22} = 1 - m_{21} \times 1 = 0.\overbrace{0\ldots0}1 \times 10^9 - 10^9 \doteq -10^9$$

$$b_2 = 2 - m_{21} \times 1 \doteq -10^9$$

$$\Rightarrow \begin{bmatrix} 10^{-9} & 1 & 1 \\ 0 & -10^9 & -10^9 \end{bmatrix}$$

列主消元法

- 在Gauss消元第k步之前，做如下的事情：
  - 每一步选绝对值最大且不为零的元素作为主元素
  - 通过换行把它调到主元素位置，
  - 不改变方程组的解，同时又有效地克服了Gauss消元地缺陷
- 例：$\begin{cases} 10^{-9}x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$

$$\begin{bmatrix} 10^{-9} & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 2 \\ 10^{-9} & 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\Rightarrow \quad x_2 = 1, \quad x_1 = 1 \ \checkmark$$

三角矩阵法

- 高斯消元法的实质就是通过初等变换把待求方程组的系数矩阵A变换成三角矩阵，这个过程也叫矩阵的三角化。
- 常用的矩阵三角化方法：LU分解、QR分解、Cholesky分解

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \Rightarrow \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{pmatrix}$$

LU分解

- 如果方阵A的所有顺序主子式都不为零，则可以唯一的将其分解为两个三角矩阵的乘积*A*=*LU*
- 分解的理论由Gauss消元得出，因此分解能够进行的条件与Gauss消元一样

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

- 如果L为上三角矩阵，U为下三角矩阵，则称为Doolittle分解
- 如果L为下三角矩阵，U为上三角矩阵，则称为Crout分解

LU分解（2）

- 对一个线性方程组进行LU分解之后，可以得到

$$Ax = LUx = b$$
$$\Rightarrow Ux = L^{-1}b = y$$

则

$$\begin{cases} Ly = b & (1) \\ Ux = y & (2) \end{cases}$$

高斯消元法

- L是下三角矩阵，根据（1）很容易求出y，再将其带入（2），由于U是上三角矩阵，也很容易解出x。
- LU分解可以给线性方程组的求解带来很大的方便，特别是在系数矩阵相同、自由项不同时，可以显著减少计算量。

LU分解（3）

- Matlab实现LU分解的格式：

  [L,U] = lu(A)

  L是一个下三角矩阵，U是一个上三角矩阵。

  [l,u,p] = lu(A)

  l是一个单位下三角矩阵，u是一个上三角矩阵，p是代表选主元的置换矩阵，其中L等于p^{-1} l，u等于U，所以(p^{-1} l)U=A。

LU分解（4）


```
function EquLU
A=[1 2 3; 2 4 1; 4 6 7];
[l1,u1,p]=lu(A);
```

```
l1 =
    1.0000         0         0
    0.5000    1.0000         0
    0.2500    0.5000    1.0000
u1 =
    4.0000    6.0000    7.0000
         0    1.0000   -2.5000
         0         0    2.5000
p =
     0     0     1
     0     1     0
     1     0     0
```

LU分解（5）


```
function EquLU
A=[1 2 3; 2 4 1; 4 6 7];
[l1,u1,p]=lu(A);


[l2,u2]=lu(A);
l3=inv(p)*l1;
```

```
l2 =
    0.2500    0.5000    1.0000
    0.5000    1.0000         0
    1.0000         0         0
u2 =
    4.0000    6.0000    7.0000
         0    1.0000   -2.5000
         0         0    2.5000
l3 =
    0.2500    0.5000    1.0000
    0.5000    1.0000         0
    1.0000         0         0
```

QR分解

- QR分解是将任意矩阵A分解为正交方阵Q和一个上三角阵R
  - Q满足：$Q^TQ = E$
  - R为与A同维度的上三角阵

- Matlab实现QR分解的格式：

[Q,R] = qr(A)

Q是一个正交方阵，R是一个上三角矩阵。

[Q,R,P] = qr(A)

Q是一个正交方阵，R是一个对角线元素递减的上三角矩阵，P是换位矩阵，且满足AP = QR。

QR分解（2）


```
A=[1 2 3; 2 4 1; 4 6 7];

[Q,R] = qr(A)
[Q2,R2,P] = qr(A)
```

Q =

-0.2182	-0.3904	-0.8944
-0.4364	-0.7807	0.4472
-0.8729	0.4880	0.0000

R =

-4.5826	-7.4194	-7.2012
0	-0.9759	1.4639
0	0	-2.2361

Q2 =

-0.3906	0.2020	-0.8981
-0.1302	-0.9779	-0.1633
-0.9113	0.0531	0.4082

R2 =

-7.6811	-6.7698	-4.2962
0	-3.1890	-1.5413
0	0	0.4082

P =

0	0	1
0	1	0
1	0	0

Cholesky分解

- 若矩阵A为n阶对称正定阵,则存在唯一的对角元素为正的三角阵D，使得A=D^{T}D，称为Cholesky分解。

$$A = D^T D = \begin{bmatrix} d_{11} & & & \\ d_{21} & d_{22} & & \\ \cdots & & \cdots & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix} \begin{bmatrix} d_{11} & d_{21} & \cdots & d_{n1} \\ & d_{22} & \cdots & d_{n2} \\ & & \cdots & \cdots \\ & & & d_{nn} \end{bmatrix}$$

Cholesky分解（2）

- 例：

```
function EquChol

A = [1 1 2; 1 2 0; 2 0 9];
D = chol(A);
```

```
A =
     1     1     2
     1     2     0
     2     0     9
D =
     1     1     2
     0     1    -2
     0     0     1
```

- chol函数可以用来检验矩阵是否为正定方阵，若能分解则是正定的，否则为非正定。

⑩ 利用矩阵的LU、QR和Cholesky分解求方程组的解

- （1）LU分解（A为方阵）

$$Ax = b \quad \Rightarrow \quad LUx = b$$

$$\Rightarrow \quad x = U \backslash (L \backslash b)$$

- （2）Cholesky分解（A为对称正定矩阵）

$$Ax = b \quad \Rightarrow \quad L'Lx = b$$

$$\Rightarrow \quad x = L \backslash (L' \backslash b)$$

- （3）QR分解（A为任意矩阵）

$$Ax = b \quad \Rightarrow \quad QRx = b$$

$$\Rightarrow \quad x = R \backslash (Q \backslash b)$$

❿ 这三种分解，在求解大型方程组时很有用，优点是运算速度快、可以节省磁盘空间、节省内存。

⑩ 例子：

利用LU、QR分解求解线性方程组的解，其中：

```
A =

     1     3     5     7
     2    -1     3     5
     0     0     2     5
    -2    -6    -3     1

b =

     1
     2
     3
     4
```

```
A = [1 3 5 7;
     2 -1 3 5;
     0 0 2 5;
     -2 -6 -3 1];
 b = [1 2 3 4]';

[L,U] = lu(A);
x1 = U\(L\b);

[Q,R] = qr(A);
x2 = R\(Q\b);
```

直接求解非齐次方程组

- 关于线性方程组的直接解法，如Gauss消去法、选主元消去法、三角分解法等，在MATLAB中，只需用“／”或“\”就解决问题。
- 它内部实际包含着许许多多的自适应算法。对超定方程用最小二乘法，对欠定方程时它将给出范数最小的一个解，解三对角阵方程组时用追赶法等。

- 非齐次线性方程组：Ax=B

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2(n-1)} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} & a_{nn} \end{pmatrix}, B=\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1(p-1)} & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2(p-1)} & b_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{n(p-1)} & b_{np} \end{pmatrix}$$

⑩ 根据A和增广矩阵C来判断解的情况：

$$C=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1(n-1)} & a_{1n} & b_{11} & b_{12} & \cdots & b_{1(p-1)} & b_{1p} \\ a_{21} & a_{22} & \cdots & a_{2(n-1)} & a_{2n} & b_{21} & b_{22} & \cdots & b_{2(p-1)} & b_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} & a_{nn} & b_{n1} & b_{n2} & \cdots & b_{n(p-1)} & b_{np} \end{pmatrix}$$

恰定方程组

- rank(A) = rank(B)=n，此时$\det(A)\neq 0$，存在唯一解向量，matlab中可以采用两种方法求解：
  - 矩阵求逆：x=inv(A)*B
  - 矩阵左除：x=A\B
- 例：

$$\begin{bmatrix} -1 & -1 & 1 \\ 5 & -4 & 3 \\ 2 & 7 & 8 \end{bmatrix} X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

恰定方程组（2）


```
function NonhomoEqu
clear;clc;

A = [1 -1 1; 5 -4 3; 2 7 8];
B = [1 2; 3 4; 5 6];
C = [A B];

rankA = rank(A);
rankC = rank(C);
[rankA rankC]

X1 = inv(A)*B;
X2 = A\B;
```

```
ans =
     3     3
X1 =
   -0.1250   -1.6667
   -0.2500   -1.3333
    0.8750    2.3333
X2 =
   -0.1250   -1.6667
   -0.2500   -1.3333
    0.8750    2.3333
```

欠定方程组

- rank(A) = rank(B) < n，此时存在无穷多个解向量，由对应齐次方程组的通解加上非齐次方程组的特解。
  - 齐次方程组通解的求法同样采用化零空间，即X0=null(A)。
  - 特解的求解：
    - 矩阵左除：x=A\B

欠定方程组

- 例：求非齐次方程组的解

$$\begin{cases} x+y-3z-w=1 \\ 3x-y-3z+4w=4 \\ x+5y-9z-8w=0 \end{cases}$$

- （1）判定方程组解的结构

```
A = [1 1 -3 -1; 3 -1 -3 4; 1 5 -9 -8];
b = [1 4 0];
rankA = rank(A);
rankC = rank([A b]);
[rankA rankC]
```

```
ans =
     2     2
```

欠定方程组（2）

- （2）求其次方程组通解

```
u0 = null(A);
```

```
x0 =

    0.8308   -0.1937
    0.2534    0.8783
    0.4384    0.0853
   -0.2310    0.4288
```

- （3）求非齐次方程组特解

```
u1 = A\b;
```

```
Warning: Rank deficient, rank = 2, tol =  3.826647e-15.

x1 =

         0
         0
   -0.5333
    0.6000
```

欠定方程组（3）

- （4）非齐次方程组解

$$u = ku_0 + u_1$$

$$u=\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}=k_1\begin{bmatrix} 0.8308 \\ 0.2534 \\ 0.4384 \\ -0.2310 \end{bmatrix}+k_2\begin{bmatrix} -01937 \\ 0.8783 \\ 0.0853 \\ 0.4288 \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \\ -0.5333 \\ 0.6000 \end{bmatrix}$$

超定方程组

- 超定方程组：rank(A) ≠rank(B)，属于矛盾方程组，没有一般意义下的解
- 可以建立它的“正规方程组”，求出其最小二乘解
  - 最小二乘解不满足方程组中的每个方程
  - 但是把其带入方程组，左右两边差的平方和最小
- Matlab中采用左除得到的正是超定方程组的最小二乘解

超定方程组（2）

- 例：求非齐次方程组的解

$$\begin{cases} x-2y+3z-w=1 \\ 3x-y+5z-3w=2 \\ 2x+y+2z-2w=3 \end{cases}$$

- （1）判定方程组解的结构

```
A = [1 -2 3 -1; 3 -1 5 -3; 2 1 2 -2];
b = [1 2 3]';
rankA = rank(A);
rankC = rank([A b]);
[rankA rankC]
```

```
ans =
    2    3
```

超定方程组（3）

- （2）求最小二乘解

```
u = A\b;
```

```
Warning: Rank deficient, rank = 2,
tol =  2.370788e-15.


u =

     0

  0.9048

  0.7143

     0
```

- （3）检验结果

```
b1 = A*u;
```

```
b1 =

  0.3333

  2.6667

  2.3333
```

作业

- 把今天课堂上讲解的所有代码，亲自输入到Matlab应用程序并运行
- 自导方程组（恰定、欠定、超定）利用三角分解法及左除法完成方程组根结构判断、求解和检验。

附上源代码，切记！

线性方程组迭代法

迭代法

- 直接法得到的解是理论上准确的，但是它们的计算量和所需要的存储量均比较高，这在n比较小的时候还比较合适（一般n<400）
- 很多实际问题，往往要我们求解很大维数的矩阵，而且这些矩阵往往是系数矩阵含有大量的0元素。对于这类的矩阵，在用直接法时就会耗费大量的时间和存储单元。
- 引入一类新的方法：迭代法。

迭代法原理

- 对于线性方程组Ax = b，如果A是非奇异矩阵（方阵，且|A|≠0），则方程组有唯一解。我们可以把A分解成两个矩阵之差：

$$A = C - D$$

则有：$(C-D)x = b$

$$C^{-1}(C-D)x = C^{-1}b$$

$$x = C^{-1}Dx + C^{-1}b = Mx + g$$

于是，可以构造迭代公式：

$$x_{k+1} = Mx_k + g \qquad (1)$$

迭代法原理（2）

- 如果给定一个初始解向量x0，通过（1）式便可以得到一个迭代序列：

$$x_0，x_1，x_2，\ldots，x_k，\ldots$$

- 如果这个时间序列收敛于某个向量x*，即

$$\lim_{k\to\infty} x_k = x^*$$

则认为迭代公式收敛，否则发散。

- 这个收敛的向量x*即认为是方程组的解。

Jacobi（雅克比）迭代法

- 根据上述原理，假设线性方程组$Ax = b$，A是非奇异矩阵，把A分解成：

$$A = D + (-L) + (-U)$$

其中

（1）D是对角矩阵，$D = diag(a_{11},a_{22},\ldots,a_{nn})$

（2）-L是严格下三角矩阵，主对角元素为0

（3）-U是严格上三角矩阵，主对角元素为0

则有：

$$(D - L - U)x = b$$

$$Dx = (L+U)x + b$$

$$x = D^{-1}(L+U)x + D^{-1}b$$

Jacobi迭代法（4）

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2(n-1)} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} & a_{nn} \end{pmatrix} \qquad D=\begin{pmatrix} a_{11} & & 0 \\ & \ddots & \\ 0 & & a_{nn} \end{pmatrix}$$

$$L=\begin{pmatrix} 0 & & & & 0 \\ -a_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ & & & 0 & \\ -a_{n1} & & \cdots & -a_{nn-1} & 0 \end{pmatrix} \qquad U=\begin{pmatrix} 0 & -a_{12} & \cdots & & -a_{1n} \\ & 0 & \ddots & & \vdots \\ & & \ddots & & \\ & & & 0 & -a_{n-1n} \\ 0 & & & & 0 \end{pmatrix}$$

Jacobi迭代法（2）

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \qquad\qquad \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases} \Rightarrow \begin{cases} x_1 = \dfrac{-1}{a_{11}}(a_{12}x_2 + \cdots + a_{1n}x_n - b_1) \\ x_2 = \dfrac{-1}{a_{22}}(a_{21}x_1 + a_{23}x_3 + \cdots + a_{2n}x_n - b_2) \\ \quad \vdots \\ x_n = \dfrac{-1}{a_{nn}}(a_{n1}x_1 + \cdots + a_{n\,n-1}x_{n-1} - b_n) \end{cases}$$

Jacobi迭代法（3）

$$
\begin{cases}
x_1^{(k+1)} = \dfrac{-1}{a_{11}}\left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} - b_1\right) \\
x_2^{(k+1)} = \dfrac{-1}{a_{22}}\left(a_{21}x_1^{(k)} + \cdots + a_{2n}x_n^{(k)} - b_2\right) \\
\qquad \vdots \\
x_n^{(k+1)} = \dfrac{-1}{a_{nn}}\left(a_{n1}x_1^{(k)} + \cdots + a_{n\,n-1}x_{n-1}^{(k)} - b_n\right)
\end{cases}
$$

Jacobi迭代法（5）

- 由此构造雅克比迭代公式：

$$x_{k+1} = D^{-1}(L+U)x_k + D^{-1}b = Mx_k + g \qquad (2)$$

- 如果迭代公式收敛，则经过无穷次迭代之后即可得到方程组的解。实际应用中，只需要满足一定的误差我们就终止迭代：

$$\| x_{k+1} - x_k \| < \varepsilon$$

其中，ε为可容忍误差。

Jacobi迭代法（4）


```
function [y,n] = Jacobi(A,b,x0)
% Jacobi iteration algorithm for linear
equation
% input:
%        A - the coefficient of matrix
%        b - free item
%        x0 - initial value
% output:
%        y - the solution of the equation
%        n - the iteration times


D = diag(diag(A));
```

```
U = -triu(A,1);
L = -tril(A,-1);
M = D\(L+U);
g = D\b;


y = M*x0+g;
n = 1;
while norm(y-x0) >= 1.0e-6
    x0 = y;
    y = M*x0+g;
    n = n+1;
end
```

Jacobi迭代法（5）


```
function TestJacobi
A = [64 -3 -1; 2 -90 1; 1 1 40];
b = [14; -5; 20];
x0 = [0 0 0]';

[y, n] = Jacobi(A,b,x0);
y2 = A\b;
```

```
n =
    5
y =
    0.2295
    0.0661
    0.4926
y2 =
    0.2295
    0.0661
    0.4926
```

Gauss-Seidel迭代法

- 用已经计算出来的k+1步的分量代替k步的分量，以提高迭代公式的收敛速度和计算精度

$$
\begin{cases}
x_1^{(k+1)} = \dfrac{-1}{a_{11}}\left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} - b_1\right) \\
x_2^{(k+1)} = \dfrac{-1}{a_{22}}\left(a_{21}x_1^{(k+1)} + \cdots + a_{1n}x_n^{(k)} - b_2\right) \\
\vdots \\
x_n^{(k+1)} = \dfrac{-1}{a_{nn}}\left(a_{n1}x_1^{(k+1)} + \cdots + a_{n\,n-1}x_{n-1}^{(k+1)} - b_n\right)
\end{cases}
$$

Gauss-Seidel迭代法(2)

- 假设线性方程组$Ax = b$，A是非奇异矩阵，把A分解成：

$$A = D + (-L) + (-U)$$

则有：$(D - L)x = Ux + b$

$$x = (D - L)^{-1}Ux + (D - L)^{-1}b = Mx + g$$

Gauss-Seidel迭代法(3)


```
function [y,n] = Seidel(A,b,x0)
% Seidel iteration algorithm for linear
equation
% input:
%       A - the coefficient of matrix
%       b - free item
%       x0 - initial value
% output:
%       y - the solution of the equation
%       n - the iteration times


D = diag(diag(A));
U = -triu(A,1);
L = -tril(A,-1);
G = (D-L)\U;
f = (D-L)\b;


y = G*x0+f;
n = 1;
while norm(y-x0) >= 1.0e-6
    x0 = y;
    y = G*x0+f;
    n = n+1;
end
```

Gauss-Seidel迭代法(4)


```
function TestSeidal
A = [64 -3 -1; 2 -90 1; 1 1 40];
b = [14; -5; 20];
x0 = [0 0 0]';

tic;
[y, n] = Seidel(A,b,x0);
toc;

y2 = A\b;
```

```
y =
    0.2295
    0.0661
    0.4926
y2 =
    0.2295
    0.0661
    0.4926
>> testJacobi
Elapsed time is 0.000683 seconds.
>> testSeidel
Elapsed time is 0.000633 seconds.
```

迭代法的收敛性

迭代法是否收敛，取决于迭代公式中的迭代矩阵M：

$$x_{k+1} = Mx_k + g$$

- 通过迭代公式推出每步的迭代向量：

$$x_1 = Mx_0 + g$$

$$x_2 = Mx_1 + g = M^2x_0 + (M+E)g$$

……

$$x_k = Mx_{k-1} + g = M^kx_0 + (M^{k-1}+M^{k-2}+\ldots+E)\,g$$

$$= M^kx_0+(E-M)^{-1}(E-M^{k-1})$$

理论证明，解向量序列收敛的充要条件是：

$$k\to\infty\text{时，}\ M^k\to 0\text{，也即}\lim_{k\to\infty}M^k=0$$

迭代法的收敛性（2）

- 由矩阵特征值的定义可知，若为λ_i为矩阵M的第i个特征值，相应的特征向量为x_i，则有

$$Mx_i = \lambda_i x_i$$

$$\Rightarrow M^k x_i = \lambda_i^k x_i$$

于是：

$$\lim_{k\to\infty} M^k = 0 \quad \Leftrightarrow \quad |\lambda_i| < 1$$

- 只要保证矩阵M的谱半径（最大特征值的绝对值）ρ(M)满足：

$$\rho(M) = \max_{1\le i\le n} |\lambda_i| < 1$$

迭代法的收敛性（3）

- 二种方法都存在收敛性问题：
  - Gauss-Seidel法收敛时，Jacobi法可能不收敛；
  - 而Jacobi法收敛时， Gauss-Seidel法也可能不收敛。

矩阵特征值和特征向量

- MATLAB求矩阵特征值特性向量格式如下：

（1） r = eig (A)

只求解矩阵A的特征值。

（2） [x,r] = eig (A)

求解矩阵A的特征值r和特征向量x。

矩阵特征值和特征向量（2）


```
function EigValue
A = [64 -3 -1; 2 -90 1; 1 1 40];
D = diag(diag(A));

U = -triu(A,1);
L = -tril(A,-1);
MJ = D\(L+U);    % Jacobi迭代公式
MS = (D-L)\U;     % Gauss-Seidel迭代公式

r1 = eig(MJ);
r2 = eig(MS);

lamuda1 = max(max(abs(r1)));
```

```
lamuda2 = max(max(abs(r2)));
```

```
r1 =
  -0.0323
    0.0162 + 0.0203i
    0.0162 - 0.0203i
r2 =
        0
0.0002 + 0.0036i
0.0002 - 0.0036i
lamuda1 =    0.0323
lamuda2 =    0.0036
```

作业

❿ 自导方程组，采用Gauss-Seidel法、Jacobi法求解方程组的解，并判断敛散性。

附上源代码，切记！

第二次课 Q & A

- 新版本中 \(左除) inv( ) pinv()
  - \(左除）高级版本不再推荐使用（不可逆时求解会有问题）
  - inv(A) 要求A为可逆
  - pinv(A)矩阵A不要求一定可逆，伪逆（最小二乘意义上）
  - 一般求解方程，用inv/pinv
- MATLAB中LU分解A可不为方阵
  - A不为方阵时，分解出的L U可能不为标准三角矩阵
  - 分解后求解方程的解仍然有效（为什么？）
- [l,u,p] = lu(A)
  - l是一个单位下三角矩阵，u是一个上三角矩阵，p是代表选主元的置换矩阵，其中L等于p^{-1} l，u等于U，所以(p^{-1} l)U=A。

第二次课 Q & A

- 自定义function .m文件运行问题

```
1 -   A = [64 -3 -1; 2 -90 1; 1 1 40];
2 -   b = [14; -5; 20];
3 -   x0 = [0 0 0]';
4 -   [y, n] = Jacobi(A,b,x0);
5 -   y2 = A\b;
```

```
>> test
Undefined function or variable 'Jacobi'.

Error in test (line 4)
[y, n] = Jacobi(A,b,x0);
```

  - 调用未定义（undefined）function
  - 原因
    - 1.定义的function函数名与所在的.m文件文件名不一致
    - 2.定义的function .m文件不在当前工作目录下
- Gauss迭代与Gauss-Seidel迭代区别
- 希望大家积极完成评教工作
  - 希望大家积极评教，点赞给好评，谢谢大家
  - 对本课程有意见和建议，可以与PM我或私下交流
  - 办公室：国防楼 308

Matlab内置函数求解线性方程组

Matlab内置函数

- Matlab具有强大的数值计算能力，提供了很多内置的求解线性方程组的函数。
- 基于共轭梯度法的线性方程组求解

lsqr	共轭梯度法
bicg	双共轭梯度算法

这两种方法都从一组初始向量出发，通过定义代价函数，进而将线性方程组求解问题转化为最优化问题，最终找到一组向量使得计算的代价函数取得极值。最优化问题求解分别采用共轭梯度和双共轭梯度法。

共轭梯度法

[x,flag,relres,iter,resvec] = lsqr(A,b,tol,maxit,M,x0);

[x,flag,relres,iter,resvec] = bicg(A,b,tol,maxit,M,x0);

求解方程组形式：Ax = b

输入参数：

- tol：最大容忍误差
- maxit：最大迭代次数
- M：为了改善收敛速度而自定义的构造矩阵，常用构造方法有LU分解，Cholesky分解(对正定矩阵) 等
- x0：初值

共轭梯度法（2）

- 输出参数：

  x：线性方程组的解

  flag：计算成功与否标识

  relres：相对误差

  iter：迭代次数

  resvec：每次迭代的残差，也即||b-Ax||

共轭梯度法（3）

- 例：

$$\begin{cases} 5x_1+6x_2=1 \\ x_1+5x_2+6x_3=0 \\ x_2+5x_3+6x_4=0 \\ x_3+5x_4+6x_5=0 \\ x_4+5x_5=1 \end{cases}$$

```
function EquBicg
A = [5 6 0 0 0; 1 5 6 0 0 ; 0 1 5 6 0; 0 0 1 5
6; 0 0 0 1 5];
b = [1 0 0 0 1]';


tol = 1e-7;
[x1,flag1,relres1,iter1,resvec1] = lsqr(A,b,tol);
[x2,flag2,relres2,iter2,resvec2] = bicg(A,b,tol);
```

共轭梯度法（4）


```
x1 =
    2.2662
   -1.7218
    1.0571
   -0.5940
    0.3188
```

```
flag1 =  0
relres1 = 7.4556e-13
iter1 =   5
resvec1 =
    1.4142
    0.9578
    0.7755
    0.7720
    0.7567
    0.0000
```

```
x2 =
    2.2662
   -1.7218
    1.0571
   -0.5940
    0.3188
```

```
flag2 =  0
relres2 =  9.6340e-15
iter2 =  5
resvec2 =
    1.4142
    1.2166
    1.9474
    1.3807
    2.0930
    0.0000
```

最小残差法

- 基于最小残差法的线性方程组求解

minres	最小残差法
qmr	标准最小残差法
gmres	广义最小残差法

最小残差法（2）

❿ 例：

$$
\begin{cases}
5x_1 + 6x_2 = 1 \\
x_1 + 5x_2 + 6x_3 = 0 \\
x_2 + 5x_3 + 6x_4 = 0 \\
x_3 + 5x_4 + 6x_5 = 0 \\
x_4 + 5x_5 = 1
\end{cases}
$$

```
function EquMinres


A = [5 6 0 0 0; 1 5 6 0 0 ; 0 1 5 6 0; 0 0 1 5 6; 0 0 0 1 5];

b = [1 0 0 0 1]';


tol = 1e-7;

[x1,flag1,relres1,iter1,resvec1] = minres(A,b,tol);

[x2,flag2,relres2,iter2,resvec2] = qmr(A,b,tol);

[x3,flag3,relres3,iter3,resvec3] = gmres(A,b,[],tol);
```

最小残差法（3）


```
x1 =
    0.1525
    0.0074
   -0.0680
    0.0155
    0.1491
```

```
flag1 =    1
relres1 =
0.5370
iter1 =    5
resvec1 =
    1.4142
    0.9223
    0.9126
    0.8143
    0.7912
    0.7594
```

```
x2 =
    2.2662
   -1.7218
    1.0571
   -0.5940
    0.3188
```

```
flag2 =  0
relres2 =
4.4768e-15
iter2 =    5
resvec2 =
    1.4142
    0.9223
    0.8335
    0.8951
    0.8310
    0.0000
```

```
x3 =
    2.2662
   -1.7218
    1.0571
   -0.5940
    0.3188
```

```
flag3 =   0
relres3 = 0
iter3 =   1   5
resvec3 =
    1.4142
    0.9223
    0.8335
    0.8258
    0.7682
```

最小残差法（4）

- 例：

$$\begin{cases} x_1 + 3x_3 + 4x_4 = 3 \\ 2x_2 + 2x_3 + x_4 = 4 \\ 3x_1 + 2x_2 + 3x_3 = 0 \\ 4x_1 + x_2 + 4x_4 = 0 \end{cases}$$

```
A =

     1     0     3     4
     0     2     2     1
     3     2     3     0
     4     1     0     4
```

最小残差法（5）


```
x1 =
   -0.0181
    0.8696
    1.1051
    0.0507
```

```
flag1 =    0
relres1 =
2.5042e-16
iter1 =    4
resvec1 =
    7.3655
    2.5177
    2.2048
    0.3568
    0.0000
```

```
x2 =
   -0.0181
    0.8696
    1.1051
    0.0507
```

```
flag2 =    0
relres2 =
1.2889e-15
iter2 =    4
resvec2 =
    7.3655
    2.5177
    2.2048
    0.3568
    0.0000
```

```
x3 =
   -0.0181
    0.8696
    1.1051
    0.0507
```

```
flag3 =    0
relres3 = 0
iter3 =    1   4
resvec3 =
    7.3655
    2.5177
    2.2048
    0.3568
```

Matlab内置函数选择

- 这些内置迭代算法都有各自较适合的方程组，算法的选取主要由所求解的线性方程组的系数矩阵来决定。
- 一般情况下：
  - 系数矩阵为非对称方阵的大型线性方程组可以选择广义最小残差法（gmres），双共轭梯度算法（bicg），标准最小残差法（qmr）算法；
  - 系数矩阵为对称方阵但不要求正定的大型线性方程组可选择最小残差法（minres）算法；
  - 系数矩阵为非方阵的大型线性方程组可选择共轭梯度法（lsqr）算法。

作业

⑩ 自导方程组，采用Matlab内置函数求解方程组的解，并比较不同函数的求解精度。

附上源代码，切记！

Q & A