第7章　线性方程系统

线性方程系统是最常见的计算问题，几乎在所有的应用中都把它作为子问题提出来。通常MATLAB用运算符\从左分开来求解线性方程系统，超定系统的解决方法同样可用于欠定系统。

在线性系统理论中重要的概念是行列式、逆和矩阵的秩。首先定义这些 MATLAB命令，再在7.2节中开始介绍求解的方法。在范数和条件数定义之后，介绍一些因数分解。最后一节涉及超定和欠定系统。

注意，本章中的所有命令只能用于二维矩阵。

7.1　行列式、逆和秩

下列命令用来计算矩阵**A**的行列式、逆和矩阵的秩。

命令集67　矩阵函数

`det(A)`	求方阵**A**的行列式。
`rank(A)`	求**A**的秩，即**A**中线性无关的行数和列数。
`inv(A)`	求方阵**A**的逆矩阵。如果**A**是奇异矩阵或者近似奇异矩阵，则会给出一个错误信息。
`pinv(A)`	求矩阵**A**的伪逆。如果**A**是$m\times n$的矩阵，则伪逆的大小为$n\times m$。对于非奇矩阵**A**来说，有`pinv(A)=inv(A)`。
`trace(A)`	求矩阵**A**的迹，也就是对角线元素之和。

■ **例**7.1

假设有下列矩阵：

$$\mathbf{A1}=\begin{pmatrix}1 & 3\\ 2 & 4\end{pmatrix}\quad \mathbf{A2}=\begin{pmatrix}1 & 3\\ 2 & 6\end{pmatrix}\quad \mathbf{A3}=\begin{pmatrix}1 & 3 & 2\\ 2 & 7 & 6\end{pmatrix}$$

将命令`det`、`inv`、`rank`和一些其他命令对上述矩阵进行操作。

```
(a) det1 = det(A1), det2 = det(A2), det3 = det(A3)
```

give:

```
det1 =
    -2

det2 =
     0

??? Error using ==> det
Matrix must be square.
```

仅为方阵定义行列式。

(b) 仅为方阵定义逆：

```
Inv1 = inv(A1), Inv2 = inv(A2), Inv3 = inv(A3)
```

结果为：

```
Inv1 =
   -2.0000    1.5000
    1.0000   -0.5000

Warning: Matrix is singular to working precision.
Inv2 =
   Inf   Inf
   Inf   Inf

??? Error using ==> inv
Matrix must be square.
```

对所有矩阵定义伪逆：

```
Pinv1 = pinv(A1), Pinv2 = pinv(A2), Pinv3 = pinv(A3)
```

结果为：

```
Pinv1 =
   -2.0000    1.5000
    1.0000   -0.5000

Pinv2 =
    0.0200    0.0400
    0.0600    0.1200

Pinv3 =
    0.9048   -0.3333
    1.0476   -0.3333
   -1.5238    0.6667
```

注意，**A1**的逆矩阵和它的伪逆是一样的。

(c) 如果**A**的逆矩阵存在，那么它的行列式 $\det(A^{-1})$就等于：$\frac{1}{\det(A)}$

```
detinv1 = det(inv(A1))

detinv1 =
   -0.5000
```

(d) 矩阵的秩和它的转置的秩相同：

```
rank1 = rank(A1), rank2 = rank(A2), rank3 = rank(A3)
```

结果为：

```
rank1 =
     2

rank2 =
     1
```

```
rankT3 =
     2
```

和

```
rankT1 = rank(A1'), rankT2 = rank(A2'), rankT3 = rank(A3')
```

结果为：

```
rankT1 =
     2

rankT2 =
     1

rankT3 =
     2
```

(e) 实数矩阵的行列式和它的转置的行列式相同：

```
detT1 = det(A1'), detT2 = det(A2'), detT3 = det(A3')
detT1 =
    -2

detT2 =
     0
??? Error using ==> det
Matrix must be square.
```

■

与线性系统相联系的两个子空间是值域和零空间。如果 **A**为$m\times n$的矩阵，它的秩为r，那么**A**的向量空间就是由**A**的列划分的线性空间，这个空间的维数是 r，也就是**A**的秩。如果$r=n$，则**A**的列线性无关。MATLAB命令orth用来求**A**的空间的正交基。

A的零空间是由满足条件**Ax=0**的所有向量**x**组成的线性子空间。在MATLAB中可以用命令null来求得零空间的正交基。

假设有一个向量集$\mathbf{v}_1$，$\mathbf{v}_2$，$\cdots\mathbf{v}_n$，可以通过定义矩阵**B**=($\mathbf{v}_1$ $\mathbf{v}_2$ $\cdots$ $\mathbf{v}_n$)来判断它们是否线性相关。例如，如果**B**的秩是$n-1$，那么其中的一个向量v_i可以用其他向量线性表示。

用命令subspace来求得两个向量或者两个子空间的夹角。

命令集68 值域、零空间和子空间的夹角

orth(A)	求**A**空间的正交基，它的列数等于**A**的秩。
null(A)	求**A**的零空间的正交基，它的列数等于零空间的维数。
subspace(x,y)	求列向量**x**和**y**的夹角，向量的长度必须一样。
subspace(A,B)	求由矩阵**A**和**B**的列划分的子空间的夹角，列的长度必须一样。

■ **例7.2**

用命令orth、null和subspace求解例7.1中的矩阵的正交基、零空间和夹角。

(a) 先求正交基：

```
Range1 = orth(A1), Range2 = orth(A2), Range3 = orth(A3)
```

结果为：

```
Range1 =
    0.5760    0.8174
    0.8174   -0.5760
Range2 =
    0.4472
    0.8944

Range3 =
    0.3667   -0.9303
    0.9303    0.3667
```

(b) 再求它们的秩：

```
rank1 = rank(orth(A1)), ...
rank2 = rank(orth(A2)), rank3 = rank(orth(A3))
```

结果为：

```
rank1 =
     2

rank2 =
     1

rank3 =
     2
```

当然，矩阵的向量空间的秩就等于矩阵本身的秩。

(c) 然后求它们的零空间：

```
nullSpace1 = null(A1), ...
nullSpace2 = null(A2), nullSpace3 = null(A3)
```

结果为：

```
nullSpace1 =
   Empty matrix: 2-by-0

nullSpace2 =
   -0.9487
    0.3162

nullSpace3 =
   -0.8729
    0.4364
   -0.2182
```

这里的Empty Matrix表示零空间是空的。对**A1**求它的零空间时结果就得到零向量。

(d) 求它们转置矩阵的零空间：

```
nullSpaceT1 = null(A1'), ...
nullSpaceT2 = null(A2'), nullSpaceT3 = null(A3')
```

结果为：

```
nullSpaceT1 =
   Empty matrix: 2-by-0

nullSpaceT2 =
   -0.8944
    0.4472

nullSpaceT3 =
   Empty matrix: 2-by-0
```

(e) 用orth求正交基：

```
RangeT1 = orth(A1'), ...
RangeT2 = orth(A2'), RangeT3 = orth(A3')
```

结果为：

```
RangeT1 =
    0.4046    0.9145
    0.9145   -0.4046

RangeT2 =
    0.3162
    0.9487

RangeT3 =
    0.2197   -0.4357
    0.7508   -0.4958
    0.6229    0.7513
```

(f) 最后求矩阵的夹角：

```
angle = subspace(null(A2),orth(A2'))
```

结果为：

```
angle =
    1.5708
```

■

角度为π/2，说明这两个空间是正交的。命令subspace要求列的长度相同。

7.2 线性系统的求解和LU因式分解

MATLAB中用运算符\求解线性系统，这个运算符的功能很强大，而且具有智能性。通常，仔细研究计算过程是有价值的，在MATLAB中有几个专门这样的命令。

令**A**是$n\times m$的矩阵，**b**和**x**是有n个元素的列向量，**B**和**X**是n行p列的矩阵。MATLAB用如下命令求解系统**Ax=b**：

```
x=A\b
```

求解更一般的系统**AX=B**，也用同样的方法，其中$\mathbf{B}=(\mathbf{b}_1\ \mathbf{b}_2\cdots\ \mathbf{b}_p)$：

```
X=A\B
```

如果A是一个奇异矩阵，或者是近似奇异矩阵，则会给出一个错误信息。

■ 例7.3

在MATLAB中求解下列方程组：

$$\begin{cases} 2x_1 + 3x_2 = 7 \\ 4x_1 + x_2 = 9 \end{cases}$$

先找出系数矩阵和右边的b：

$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 4 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 7 \\ 9 \end{pmatrix}$$

解向量是x=(x_1 x_2)′，可以用x=A\b来求得：

```
x =
     2
     1
```

■

MATLAB依据系数矩阵 **A**的不同而相应地使用不同的方法求解线性系统。如果可能，MATLAB先分析矩阵的结构。例如，如果**A**是对称且正定的，则使用Cholesky分解。

如果没有找到可以替代的方法，则采用高斯消元法和部分主元法。主要是对矩阵进行 LU因式分解或LU分解。这种方法就是令**A=LU**，其中**U**是一个上三角矩阵，**L**是一个带有单位对角线的下三角矩阵。

然而为了保证计算的稳定性可以使用部分主元法。也就是说，**L**通常是一个改变序列的下三角矩阵，即有些行进行互换。这样，**L**就可能显得结构不完整，将这些排列定义为交换矩阵**P**。

交换矩阵P的大小为$n \times n$，它实际上是一个单位矩阵，按照交换的顺序来交换列向量。交换矩阵的逆等于它本身的转置。

$$\mathbf{PA} = \mathbf{L}_l\mathbf{U}$$

LU因式分解可以用非交换下三角矩阵L_l表示出来：

即交换矩阵**L**由**L=P′L**$_l$给出。

MATLAB中用命令lu可以求得**U**和交换或非交换下三角矩阵**L**，后一种情况也可给出交换矩阵**P**。

命令集69 LU分解

[L,U]=lu(A)	求上三角矩阵**U**和交换下三角矩阵**L**。**L**是一个带有单位对角线的下三角矩阵和交换矩阵，即**P**的逆矩阵的乘积，见下个命令。
[L,U,P]=lu(A)	求上三角矩阵**U**、有单位对角线的下三角矩阵 **L**和交换矩阵**P**，满足**LU=PA**。

■ **例**7.4

如果矩阵**A** = [1 2 3 ; 4 5 6 ;7 8 0]，输入[L,U]=lu(A)，运行结果为：

```
L =
    0.1429    1.0000         0
    0.5714    0.5000    1.0000
    1.0000         0         0

U =
    7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000
```

这里的交换矩阵的逆为：

$$\mathbf{P}^{-1}=\mathbf{P}'=\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}$$

下面列出高斯消元法的详细步骤：

$$\mathbf{A}=\begin{pmatrix}1&2&3\\4&5&6\\7&8&0\end{pmatrix}\quad \mathbf{A1}=\begin{pmatrix}7&8&0\\4&5&6\\1&2&3\end{pmatrix}\quad \mathbf{A2}=\begin{pmatrix}7&8&0\\0&0.4286&6\\0&0.8571&3\end{pmatrix}$$

$$\mathbf{A3}=\begin{pmatrix}7&8&0\\0&0.8571&3\\0&0.4286&6\end{pmatrix}\quad \mathbf{A4}=\begin{pmatrix}7&8&0\\0&0.8571&3\\0&0&4.5\end{pmatrix}$$

先从矩阵**A**开始，第1个主元的位置在(1, 1)，通过第1行和第3行互换，可以得到最大的主元，也就是7。第1次交换后得到矩阵**A1**，第1次消元后得到矩阵**A2**。

第2个主元在(2, 2)上，将第2行和第3行交换后得到矩阵**A3**，消元后得到矩阵**A4**，和矩阵**U**是同一个矩阵。

这个过程进行了两次交换，第1次是第1行和第3行互换，也就是：

$$\mathbf{P1}=\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$$

第2次进行第2、3行互换，也就是：

$$\mathbf{P2}=\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}$$

P1和**P2**相乘，形成P：

$$\mathbf{P1}*\mathbf{P2}=\mathbf{P}=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}$$

P的逆为：

$$\mathbf{P}^{-1}=\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}$$

如果输入`[L,U,P]=lu(A)`，运行就会得到如下的结果：

```
L =
    1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000

U =
    7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000
```

```
P =
     0     0     1
     1     0     0
     0     1     0
```

■

有许多方法可求解 **Ax=b**这样的问题， **A**是$n\times n$的矩阵， **b**是一个长度为n的列向量。在本书中没有对算法进行完整的描述，详细信息可参见 MATLAB Help Desk。

命令集70 解方程组的方法

`x=bicg(A,b,tol, maxit,M)`	用双共轭梯度法解方程组。如果给定 *tol*，则用它来指定解的精度，也就是和`norm(b-A*x)/norm(b)`比较来判断解是否可以接受。如果给定 *maxit*，则用它作为最大的迭代数。要使用预处理因子，可在矩阵**M**中规定它。
`bicg(A,b,···,M1,M2,x0)`	操作同上，但是使用矩阵**M1**和**M2**作为预处理因子。矩阵**M1**、**M2**和预处理因子**M**的关系为**M=M1·M2**。如果给定**x0**，则将它作为初始化向量开始迭代。
`[x,flag,relres,iter, resvect]=bicg(···)`	求**x**中的问题解，有关*bigc*的收敛信息存放在*flag*中。结果的相对剩余范数保存在*iter*中。值*resvect*是每次迭代的范数，结果向量中的所有变量都可忽略。
`bicgstab(···)`	用稳定双共轭梯度法解方程组，调用方式和返回的结果形式和命令`bicg`一样。
`cgs(···)`	用复共轭梯度平方法解方程组，调用方式和返回的结果形式和命令`bicg`一样。
`gmres(A, b, restart, ···)`	用广义最小残量法解方程组，除了参数 *restart*可以给出外，调用方式和返回的结果形式和命令`bicg`一样。
`pcg(···)`	用预处理共轭梯度法解方程组，调用方式和返回的结果形式和命令`bicg`一样。
`qmr(···)`	用准最小残量法解方程组，调用方式和返回的结果形式和命令`bicg`一样。

7.3 行梯形矩阵

LU因式分解的另一种方法就是将系数矩阵 **A**降维成行梯形矩阵的形式。因为它能应用在长方矩阵上，所以这种方法更常用。这种矩阵能给出许多有关线性系统的信息。

对于每一个$m\times n$的矩阵**A**都有一个交换矩阵**P**，一个有单位对角线的下三角矩阵 **L**和一个$m\times n$的梯形矩阵**R**，它们满足**PA=LR**。

如果下列情况成立，则矩阵是梯形矩阵：

1) 如果有零行，就放在矩阵的底部；

2) 每行中第一非零元素是1，它作为每行的最主要元素；

3) 每行的最主要元素放在上一行最主要元素的右边；

4) 在有最主要元素1的列中，其他元素都是零。

可以用MATLAB中命令rref来分析系统**Ax=b**。

命令集71 缩减行阶梯矩阵

`rref(A)`	用高斯—约当消元法和行主元法求**A**的缩减行的阶梯矩阵。
`rref(A,tol)`	和rref(A)一样，但是使用精度 *tol*，它用来决定什么时候元素可以忽略不计。
`rrefmovie(A)`	求缩减行的阶梯矩阵，并给出每一步的求解过程。

通常使用的精度是`tol=max(size(A))*eps*norm(A, inf)` ，在7.6节中定义了命令`norm`。

■ **例**7.5

令

$$\mathbf{A}=\begin{pmatrix}1 & 3 & 3 & 2\\ 2 & 6 & 9 & 5\\ -1 & -3 & 3 & 0\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}2 & 1 & 1\\ 4 & 1 & 0\\ -2 & 2 & 1\end{pmatrix}$$

(a) 如果用命令rref(A)得到的矩阵中有一个或多个零行，则可以知道矩阵 **A**中有一些行线性相关。同样也可用命令`rref(A´)` 来知道矩阵**A**的列向量的相关性。

输入命令`Aref=rref(A)`，`Bref=rref(B)`，运行结果为：

```
Aref =
    1.0000    3.0000         0    1.0000
         0         0    1.0000    0.3333
         0         0         0         0

Bref =
     1     0     0
     0     1     0
     0     0     1
```

(b) 求矩阵**A**的秩可以在**A**的缩减行的阶梯矩阵中数非零行的数量。在这个例子中，可以知道**A**的秩是2，**B**的秩是3。可以用命令`rankA=rank(A)`，`rankB=rank(B)`来确认一下：

```
rankA =
     2

rankB =
     3
```

■

命令`rref`可以用来研究线性方程组，令**Ax=b**代表一个线性方程组，令矩阵**B**=(A b)。用命令`C=rref(B)`求得矩阵**B**的缩减行的阶梯矩阵**C**，然后有下列结论：

• 如果**C**中有一个或多个全零的行向量，则这个方程组中有冗余信息。这就意味着可以去掉一个或多个方程

• 如果**C**中有除了最后一个元素不是零外其余都是零的行，即(0, 0, ···,0, 1)，则这个方程组无解

• 如果方程组有唯一解，则在C的最后一个列可以得到该解。

7.4 Cholesky因式分解

如果矩阵**A**是一个对称正定矩阵，即**A**=**A**′且对于每个**x**≠0都有x′**Ax**>0，则存在一个上三角矩阵**G**，它的对角线元素是正数，且满足**G**′**G**=**A**。

这是LU因式分解的一种特殊情况称为Cholesky因式分解，它只有标准LU因式分解的大约一半的计算步骤。注意，在有些书中是按照下三角矩阵来定义Cholesky因式分解的。

在用左除\来解对称正定的方程组时，MATLAB会自动用Cholesky因式分解来求解。

命令chol可用来计算正定矩阵**A**的Cholesky因式分解。

命令集72 Cholesky因式分解

`chol(A)`	求矩阵**A**的Cholesky因子，是一个上三角矩阵。如果**A**不是一个正定矩阵，则给出一个错误信息。
`[G,err]=chol(A)`	求矩阵**A**的Cholesky因子**G**。如果**A**不是一个正定矩阵，则不给出错误信息，而是将err设为非零值。
`R1=cholupdate(R,x)`	如果**R**=chol(A)且x是一个和**A**的列长度一样的列向量，则求**A**+**xx**′的上三角Cholesky因子**R1**。
`R1=cholupdate(R,x,'-')`	如果R=chol(A)且x是一个和**A**的列长度一样的列向量，则求A－**xx**′的上三角Cholesky因子**R1**。

■ **例**7.6

```
(a) b = [-1 -1 -1]; A = 4*eye(4) + diag(b,-1) + diag(b,1),...
    G = chol(A)
```

结果为：

```
A =
     4    -1     0     0
    -1     4    -1     0
     0    -1     4    -1
     0     0    -1     4

G =
    2.0000   -0.5000         0         0
         0    1.9365   -0.5164         0
         0         0    1.9322   -0.5175
         0         0         0    1.9319
```

用Test=G′*G来检验这个结果，得：

```
Test =
    4.0000   -1.0000         0         0
   -1.0000    4.0000   -1.0000         0
         0   -1.0000    4.0000   -1.0000
         0         0   -1.0000    4.0000
```

这样就又得到矩阵**A**。

(b) 为了看清楚LU因式分解和Cholesky因式分解在计算步骤的不同，输入：

```
flops(0), lu(A); flops, flops(0), chol(A); flops
```

结果显示为：

```
ans =
       34
ans =
       30
```

■

对于较大的方程组来说，它们之间的差别更加明显。

7.5 QR因式分解

解线性方程组除了LU因式分解和Cholesky因式分解，还有第三种方法即QR因式分解或QR分解。

假设**A**是$n\times n$的矩阵，那么**A**就可以分解成：

```
A=QR
```

其中**Q**是一个正交矩阵，**R**是一个大小和**A**相同的上三角矩阵，因此**Ax=b**可以表示为**QRx=b**或者等同于：

```
Rx=Qb
```

这个方程组的系数矩阵是上三角的，因此容易求解。

和高斯消元法比较，QR因式分解的主要优点在于有更高的稳定性，然而它的数学运算更麻烦一些。

MATLAB中用命令qr来求QR因式分解，这个命令可以分解$m\times n$的矩阵，所以考虑一般情况，假设**A**是$m\times n$的矩阵。

命令集73 QR因式分解

`[Q,R]=qr(A)`	求得$m\times m$的矩阵**Q**和上三角矩阵**R**，**Q**的列形成了一个正交基，**Q**和**R**满足**A=QR**。
`[Q,R,P]=qr(A)`	求得矩阵**Q**、上三角矩阵**R**和交换矩阵**P**。**Q**的列形成一个正交基，**R**的对角线元素按大小降序排列，它们满足**AP=QR**。
`[Q,R]=qr(A,0)`	求矩阵**A**的QR因式分解。如果在$m\times n$的矩阵**A**中行数小于列数，则给出**Q**的前n列，因此**Q**的大小和**A**相同。也能得到交换矩阵，见上或输入help qr可获得帮助。
`[Q1,R1]=qrdelete(Q,R,j)`	求去掉矩阵**A**中第j列之后形成的矩阵的**QR**因式分解，矩阵**Q**和**R**是**A**的**QR**因子。
`[Q1,R1]=qrinsert(Q,R,b,j)`	求在矩阵**A**的第j列前插入一列向量**b**后形成的矩阵的**QR**因式分解，矩阵**Q**和**R**是**A**的**QR**因子。如果$j=n+1$，那么插入的一列放在最后。

■ **例**7.7

(a) 解**Ax=b**，其中**A**和**b**给出如下：

$$\mathbf{A}=\begin{pmatrix}1 & 2 & 2\\ 3 & 2 & 2\\ 1 & 1 & 2\end{pmatrix} \qquad \mathbf{b}=\begin{pmatrix}7\\ 9\\ 5\end{pmatrix}$$

```
[Q,R] = qr(A), x = R\Q'*b
```

结果为：

```
Q =
   -0.3015    0.9239   -0.2357
   -0.9045   -0.3553   -0.2357
   -0.3015    0.1421    0.9428

R =
   -3.3166   -2.7136   -3.0151
         0    1.2792    1.4213
         0         0    0.9428

x =
     1
     2
     1
```

这个结果和用命令A\b得到的结果一样。

(b) 下面比较一下用 QR因式分解法和左除法解线性方程组的计算步骤。为了比较明显，用它们来解一个较大的方程组：

```
A = rand(10,10); b = rand(10,1); flops(0); ...
x = A\b; lureq = flops
```

结果为：

```
lureq =
        1506
```

和

```
flops(0), [Q,R] = qr(A); x = R\Q'*b; qrreq = flops
```

结果为：

```
qrreq =
        5285
```

■

QR因式分解能用来解超定方程组，这样的方程组中方程的个数比未知变量的个数多 (见7.7节)，还能用来求特征值和特征向量。

计算矩阵QR因式分解的一种方法可以应用在 **Givens旋转**上。用命令planerot来求长度为2的向量的Givens平面旋转。必须用冒号表达式才能将 Givens平面旋转应用在矩阵上。

命令集74 Givens和Jacobi旋转

planerot(x)	求去掉有两个元素的向量 **x**中第1个元素的2×2矩阵的Givens旋转。在MATLAB函数qrinsert和qrdelete中使用这个命令。
[G,y]=planerot(x)	在**G**中返回Givens旋转，结果为**y**=**Gx**。
rjr(A)	求**A**的Jacobi旋转，旋转的角度和平面都是随机数。保存有特征向量、奇异值和对称性。

■ **例**7.8

Givens旋转可以描述成平面旋转，假设2×2的矩阵**A**有列向量**x**和**y**。

$$\mathbf{A}=\begin{pmatrix}1 & 1\\ -1 & 2\end{pmatrix}\qquad \mathbf{x}=\begin{pmatrix}1\\ -1\end{pmatrix}\qquad \mathbf{y}=\begin{pmatrix}1\\ 2\end{pmatrix}$$

如果输入`G=planerot(x)`，则**G**旋转向量**x**(**A**的第1列)成*x*轴，并用`Anew=G*A`对**A**操作，得到的结果为：

```
G =
    0.7071   -0.7071
    0.7071    0.7071

Anew =
    1.4142   -0.7071
         0    2.1213
```

用下列命令求出**A**和**Anew**的列向量范数是一样的：

```
xnorm = norm(x), ynorm = norm(y), ...
xnewnorm = norm(Anew(:,1)), ynewnorm = norm(Anew(:,2))


xnorm =
    1.4142

ynorm =
    2.2361

xnewnorm =
    1.4142

ynewnorm =
    2.2361
```

每列的范数(也称长度)是不变的，在7.6节定义了命令`norm`，从图7-1可以看出对于A中的两列旋转是一样的。

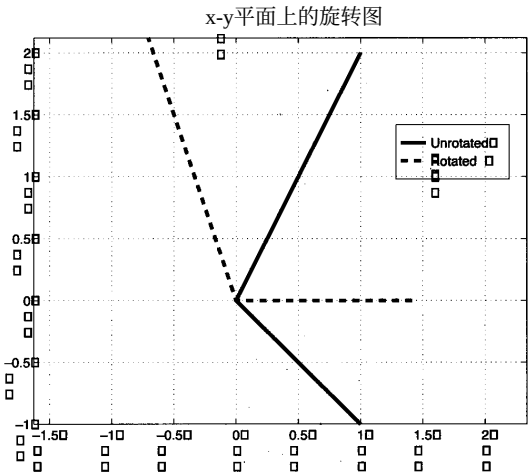

图7-1 在*x*-*y*平面上的两个向量的Givens旋转图

■

要编写一些程序才能将Givens旋转应用在矩阵上和进行缩减矩阵元素，可以参见 12.2节中关于命令planerot的例题。

7.6 范数和条件数

向量的范数是一个标量，用来衡量向量的长度，不要和向量中元素的个数相混淆。MATLAB中可以用命令norm得到不同的范数。

命令集75 向量范数

norm(x)	求欧几里得范数，即 $\|\mathbf{x}\|_2=\sqrt{\sum\|x_k\|^2}$ 。
norm(x,inf)	求∞-范数，即||x||=max(abs(x))。
norm(x,1)	求1-范数，即 $\|\mathbf{x}\|_1=\sum_k\|x_k\|$ 。
norm(x,p)	求p-范数，即 $\|\mathbf{x}\|_p=\sqrt{\sum_k\|x_k\|^p}$，所以norm(x, 2)=norm(x)
norm(x,－inf)	求向量x的元素的绝对值的最小值，即min(abs(x))。注意，这不是向量的范数。

■ **例**7.9

令

x=[3 4 5]

```
norm1   = norm(x,1),   norm2  = norm(x,2), ...
norminf = norm(x,inf), nonorm = norm(x,-inf)
```

结果为：

```
norm1 =
    12

norm2 =
    7.0711

norminf =
     5

nonorm =
     3
```

■

矩阵范数用来衡量矩阵大小，和矩阵的行、列数的概念是不一样的。由于数据或数字计算中的扰动，矩阵范数常用来估计误差。

用向量的范数来定义方阵的p-范数：

$$\|\mathbf{A}\|_p=\sup_{\mathbf{x}\neq\mathbf{0}}\frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p}$$

定义的这些范数并不真正用在计算中，而是用命令集 76中的表达式，它们可以求方阵范数，也可以用来求非方阵的范数。

由于欧几里得范数计算很复杂，所以用命令 normest来计算欧几里得范数的估计值。

命令集76 矩阵范数

`norm(A)`	求欧几里得范数$\|A\|_2$，等于**A**的最大奇异值，参见8.3节。
`norm(A,1)`	求列范数$\|A\|_1$，等于**A**的列向量的1-范数的最大值。
`norm(A,2)`	求欧几里得范数$\|A\|_2$，和`norm(A)`一样。
`norm(A,inf)`	求行范数$\|A\|_\infty$，等于**A**的行向量的1-范数的最大值。
`norm(A,'fro')`	求Frobenius范数$\|\mathbf{A}\|_F=\sqrt{\sum_i\sum_j\|a_{ij}\|^2}$，这不能用矩阵$p$-范数的定义来求。
`normest(A)`	求欧几里得范数的估计值，相对误差小于10^6。
`normest(A,tol)`	求欧几里得范数的估计值，相对误差小于*tol*。

■ **例**7.10

(a) 令

```
A = [1 1;2 3];

norm1   = norm(A,1),   norm2 = norm(A,2), ...
norminf = norm(A,inf), normf = norm(A,'fro')
```

结果为：

```
norm1 =
     4

norm2 =
    3.8643

norminf =
     5

normf =
    3.8730
```

(b) 求一个大矩阵的欧几里得范数和欧几里得范数估计值，并对它们进行比较：

```
A = rand(100);
flops(0), norm2 = norm(A), expensive = flops
```

结果为：

```
norm2 =
   49.8483

expensive =
     2948409
```

和

```
flops(0), normapprox = normest(A), cheaper = flops
```

结果为：

```
normapprox =
```

```
    50.6701

cheaper =
      133211
```

■

令**Ax=b**表示线性方程组。方程组的条件数是一个大于或者等于 1的实数，用来衡量关于数据中的扰动，也就是 **A**和/或**b**，对解**x**的灵敏度。一个差条件的方程组的条件数很大。条件数的定义为：

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$$

命令cond(A)可求出欧几里得范数的条件数，就是**A**的最大奇异数和最小奇异数的商；参见8.3节。

命令集77 条件数

cond(A)	求**A**的欧几里得范数的条件数。
cond(A,p)	求*p*-范数的条件数， *p*的值可以是1、2、*inf*或者'*fro*'；见命令集76中命令norm。
condest(A)	求矩阵**A**条件数的1-范数中的下界估计值。
[c,v]= condest(A)	求**A**的1-范数中条件数c的下界估计值，同时还计算向量v，使得它们满足条件 $\|\mathbf{A}v\| = \frac{\|A\|\|v\|}{c}$ 。
[c,v]= condest(A,tr)	求如上的*c*和**v**，同时显示出关于计算的步骤信息。如果*tr*=1，则计算的每步都显示出来；如果*tr*=—1，则给出商c/rcond(A)。
rcond(A)	求矩阵**A**定义的方程组的敏感度的另一个估计值。对于差条件矩阵**A**来说，给出一个接近于 0的数；对于好条件矩阵**A**，则给出一个接近于1的数。

■ **例**7.11

求Hilbert矩阵的条件数(见4.4节)：

```
bad=cond(hilb(5))
```

结果为：

```
bad =
   4.7661e+05
```

这表明在最坏情况下右边或者系数矩阵的一个扰动和数bad相乘以后，可能会丢失五位小数。

■

7.7 超定方程组和欠定方程组

对于一个系数矩阵是*m*×*n*的线性方程组**Ax=b**来说，如果*m*>*n*，也就是说方程的个数多于未知数，则称为超定方程组。通常这些方程组是矛盾的，所以方程组没有精确解。在拟合实验数据的曲线时，常会遇到这个问题。

关键是要找到一个向量**x**使它对*m*个方程的总误差最小。有几种方法可以求得，但是最常用的方法是最小二乘法：

$$e=\sum_{i=1}^{m}\left(b_i-\sum_{j=1}^{n}a_{ij}x_j\right)^2=\|\mathbf{b}-\mathbf{Ax}\|_2^2$$

这就是最小二乘法，最小二乘解可以用除法运算符 \或命令nnls和lscov来解得。这种方法适用于在第9章中讨论的稀疏矩阵，下面提到了在计算中创建稀疏矩阵的命令 spaugment。

命令集78 最小二乘解

A\b	求最小二乘解，也可参见 3.3节。如果**b**=**B**是一个矩阵，则是对和**B**中每一列相对应的方程求解。
spaugment(A,c)	生成一个对称的稀疏方阵Γ=[c*I A; A´0]，可以用Γ\z来解超定方程组**Ax=b**，其中**z**是带有尾随零的向量**b**。参数c可以省去，MATLAB就根据spparms中的设定来使用值。输入help spparms 可得更多帮助。
nnls(A,b)	求非负最小二乘解，输入help nnls可得到更多帮助。
lscov(A,b,v)	求在已知协方差V情况下的最小二乘解，这意味着(**b**−**Ax**′)′ ($\mathbf{V}^{-1}$(**b**−**Ax**)最小，输入help lscov 可得到更多帮助。

MATLAB在用左除解超定方程组时都用 QR因式分解。如果方程组的系数矩阵不满秩，就没有唯一的最小二乘解，将不定值设为零。然后给出一个警告信息。

■ **例**7.12

令

$$\mathbf{A1}=\begin{pmatrix}1&1\\1&2\\1&3\end{pmatrix}\qquad \mathbf{A2}=\begin{pmatrix}1&3\\1&3\\1&3\end{pmatrix}\qquad \mathbf{b}=\begin{pmatrix}2\\0\\-1\end{pmatrix}$$

```
fullrank = A1\b, notfullrank = A2\b
```

结果为：

```
fullrank =
    3.3333
   -1.5000

Warning: Rank deficient, rank = 1  tol = 3.4613e-15.

notfullrank =
         0
    0.1111
```

■

如果$n>m$，则方程组**Ax=b**是欠定方程组。这样的方程组通常有无穷组解， MATLAB在求解时给出一组解，不给出警告信息。

■ **例**7.13

(a) 令

$$\mathbf{A}=\begin{pmatrix}1&1&1\\1&2&3\end{pmatrix}\qquad \mathbf{b}=\begin{pmatrix}2\\3\end{pmatrix}$$

```
x = A\b
```

解为：

```
x =
    1.5000
         0
    0.5000
```

不幸的是MATLAB不能给出它的通解，用命令rref来研究方程组的可解性，参见7.3节。输入：

```
UnderSol = rref([A~b])
```

得到结果为：

```
UnderSol =
     1     0    -1     1
     0     1     2     1
```

从中可以找出它的通解，如：

$$\mathbf{x}=\begin{pmatrix}1+t\\1-2t\\t\end{pmatrix}$$ 对所有的t

t=0.5时就是MATLAB求得的解。

(b) 令

$$\mathbf{a}=\begin{pmatrix}1\\2\\3\end{pmatrix}\qquad \mathbf{b}=\begin{pmatrix}2\\-2\\2\end{pmatrix}$$

下面比较一下矩阵的各种除法所得结果， **a**除以**b**。

左除的结果是：

```
left = a\b
left =
    0.2857
```

或者2/7，这就是超定方程组**ax=b**的解。

右除的结果是：

```
Right = b/a
Right =
         0         0    0.6667
         0         0   -0.6667
         0         0    0.6667
```

这和(a ' \b ')结果一样，这是欠定方程组**a** ' **x** ' =**b** ' 的一个解。

数组的左除：

```
elementWiseLeft = a.\b
elementWiseLeft =
     2.0000
    -1.0000
     0.6667
```

结果和右除是一样的：

```
elementWiseRight = b./a
elementWiseRight =
     2.0000
    -1.0000
     0.6667
```

■