矩阵及其基本运算

MATLAB，即“矩阵实验室”，它是以矩阵为基本运算单元。因此，从最基本的运算单元出发，介绍 MATLAB 的命令及其用法。

1.1 矩阵的表示

1.1.1 数值矩阵的生成

1．实数值矩阵输入

MATLAB 的强大功能之一体现在能直接处理向量或矩阵。当然首要任务是输入待处理的向量或矩阵。

不管是任何矩阵（向量），我们可以直接按行方式输入每个元素：同一行中的元素用逗号（,）或者用空格符来分隔，且空格个数不限；不同的行用分号（;）分隔。所有元素处于一方括号（[ ]）内；当矩阵是多维（三维以上），且方括号内的元素是维数较低的矩阵时，会有多重的方括号。如：

```
>> Time = [11   12   1   2   3   4   5   6   7   8   9   10]
    Time =
            11   12   1   2   3   4   5   6   7   8   9   10
>> X_Data = [2.32   3.43; 4.37   5.98]
    X_Data =
            2.43   3.43
            4.37   5.98
>> Null_M = [ ]              %生成一个空矩阵
```

2．复数矩阵输入

复数矩阵有两种生成方式：

第一种方式

例 1-1

```
>> a=2.7;b=13/25;
>> C=[1,2*a+i*b,b*sqrt(a); sin(pi/4),a+5*b,3.5+1]
    C=
       1.0000                5.4000 + 0.5200i    0.8544
       0.7071                5.3000                4.5000
```

第 2 种方式

例 1-2

```
>> R=[1 2 3;4 5 6], M=[11 12 13;14 15 16]
    R =
        1     2     3
        4     5     6
    M =
       11    12    13
       14    15    16
>> CN=R+i*M
    CN =
        1.0000 +11.0000i    2.0000 +12.0000i    3.0000 +13.0000i
        4.0000 +14.0000i    5.0000 +15.0000i    6.0000 +16.0000i
```

1.1.2 符号矩阵的生成

在 MATLAB 中输入符号向量或者矩阵的方法和输入数值类型的向量或者矩阵在形式上很相像，只不过要用到符号矩阵定义函数 sym，或者是用到符号定义函数 syms，先定义一些必要的符号变量，再像定义普通矩阵一样输入符号矩阵。

1．用命令 sym 定义矩阵：

这时的函数 sym 实际是在定义一个符号表达式，这时的符号矩阵中的元素可以是任何的符号或者是表达式，而且长度没有限制，只是将方括号置于用于创建符号表达式的单引号中。如下例：

例 1-3

```
>> sym_digits = sym('[1 2 3; a b c; sin(x) cos(y) tan(z)]')
    sym_digits =
    [      1,      2,      3]
    [      a,      b,      c]
    [ sin(x), cos(y), tan(z)]
```

1.1.3 大矩阵的生成

对于大型矩阵，一般创建 M 文件，以便于修改：

例 1-6　用 M 文件创建大矩阵，文件名为 example.m

```
exm=[ 456    468    873     2   579    55
       21     687    54    488     8    13
       65    4567    88     98    21     5
      456     68   4589    654     5   987
     5488     10      9      6    33    77]
```

在 MATLAB 窗口输入：

```
>>example;
>>size(exm)     %显示 exm 的大小
ans=
     5   6          %表示 exm 有 5 行 6 列。
```

1.1.4 多维数组的创建

函数　**cat**

格式　A=cat(n,A1,A2,…,Am)

说明　n=1 和 n=2 时分别构造[A1；A2]和[A1，A2]，都是二维数组，而 n=3 时可以构造出三维数组。

例 1-7

```
>> A1=[1,2,3;4,5,6;7,8,9];A2=A1';A3=A1-A2;
>> A4=cat(3,A1,A2,A3)
A4(:,:,1) =
      1      2      3
      4      5      6
      7      8      9
A4(:,:,2) =
      1      4      7
      2      5      8
      3      6      9
A4(:,:,3) =
      0     -2     -4
      2      0     -2
      4      2      0
```

或用另一种原始方式可以定义：

例 1-8

```
>> A1=[1,2,3;4,5,6;7,8,9];A2=A1';A3=A1-A2;
>> A5(:,:,1)=A1, A5(:,:,2)=A2, A5(:,:,3)=A3
A5(:,:,1) =
     1     2     3
     4     5     6
     7     8     9
A5(:,:,2) =
     1     4     7
     2     5     8
     3     6     9
A5(:,:,3) =
     0    -2    -4
     2     0    -2
     4     2     0
```

1.1.5 特殊矩阵的生成

命令　全零阵

函数　**zeros**

格式　B = zeros(n)　　%生成 n×n 全零阵

B = zeros(m,n)　　%生成 m×n 全零阵

B = zeros([m n])　　%生成 m×n 全零阵

B = zeros(d1,d2,d3…)　　%生成 d1×d2×d3×…全零阵或数组

B = zeros([d1 d2 d3…])　　%生成 d1×d2×d3×…全零阵或数组

B = zeros(size(A))　　%生成与矩阵 A 相同大小的全零阵

命令　单位阵

函数　**eye**

格式　Y = eye(n)　　%生成 n×n 单位阵

Y = eye(m,n)　　%生成 m×n 单位阵

Y = eye(size(A))　　%生成与矩阵 A 相同大小的单位阵

命令　全 1 阵

函数　**ones**

格式　Y = ones(n)　　%生成 n×n 全 1 阵

Y = ones(m,n)　　%生成 m×n 全 1 阵

Y = ones([m n])　　%生成 m×n 全 1 阵

Y = ones(d1,d2,d3…)　　%生成 d1×d2×d3×…全 1 阵或数组

Y = ones([d1 d2 d3…])　　%生成 d1×d2×d3×…全 1 阵或数组

Y = ones(size(A))　　%生成与矩阵 A 相同大小的全 1 阵

命令　均匀分布随机矩阵

函数　**rand**

格式　Y = rand(n)　　%生成 n×n 随机矩阵，其元素在（0，1）内

Y = rand(m,n)　　%生成 m×n 随机矩阵

Y = rand([m n])　　%生成 m×n 随机矩阵

Y = rand(m,n,p,…)　　%生成 m×n×p×…随机矩阵或数组

Y = rand([m n p…])　　%生成 m×n×p×…随机矩阵或数组

Y = rand(size(A))　　%生成与矩阵 A 相同大小的随机矩阵

rand　　%无变量输入时只产生一个随机数

s = rand('state')　　%产生包括均匀发生器当前状态的 35 个元素的向量

rand('state', s)　　%使状态重置为 s

rand('state', 0)　　%重置发生器到初始状态

```
rand('state', j)                    %对整数 j 重置发生器到第 j 个状态
rand('state', sum (100*clock))      %每次重置到不同状态
```

例 1-9　产生一个 3×4 随机矩阵

```
>> R=rand(3,4)
R =
    0.9501    0.4860    0.4565    0.4447
    0.2311    0.8913    0.0185    0.6154
    0.6068    0.7621    0.8214    0.7919
```

例 1-10　产生一个在区间[10, 20]内均匀分布的 4 阶随机矩阵

```
>> a=10;b=20;
>> x=a+(b-a)*rand(4)
x =
   19.2181   19.3547   10.5789   11.3889
   17.3821   19.1690   13.5287   12.0277
   11.7627   14.1027   18.1317   11.9872
   14.0571   18.9365   10.0986   16.0379
```

命令　正态分布随机矩阵

函数　**randn**

格式

```
Y = randn(n)                %生成 n×n 正态分布随机矩阵
Y = randn(m,n)              %生成 m×n 正态分布随机矩阵
Y = randn([m n])            %生成 m×n 正态分布随机矩阵
Y = randn(m,n,p,…)          %生成 m×n×p×…正态分布随机矩阵或数组
Y = randn([m n p…])         %生成 m×n×p×…正态分布随机矩阵或数组
Y = randn(size(A))          %生成与矩阵 A 相同大小的正态分布随机矩阵
randn                       %无变量输入时只产生一个正态分布随机数
s = randn('state')          %产生包括正态发生器当前状态的 2 个元素的向量
s = randn('state', s)       %重置状态为 s
s = randn('state', 0)       %重置发生器为初始状态
s = randn('state', j)       %对于整数 j 重置状态到第 j 状态
s = randn('state', sum(100*clock))      %每次重置到不同状态
```

例 1-11　产生均值为 0.6，方差为 0.1 的 4 阶矩阵

```
>> mu=0.6; sigma=0.1;
>> x=mu+sqrt(sigma)*randn(4)
x =
    0.8311    0.7799    0.1335    1.0565
    0.7827    0.5192    0.5260    0.4890
    0.6127    0.4806    0.6375    0.7971
    0.8141    0.5064    0.6996    0.8527
```

命令　产生随机排列

函数　**randperm**

格式

```
p = randperm(n)         %产生 1～n 之间整数的随机排列
```

例 1-12

```
>> randperm(6)
ans =
     3     2     1     5     4     6
```

命令　产生线性等分向量

函数　**linspace**

格式

```
y = linspace(a,b)       %在(a, b)上产生 100 个线性等分点
y = linspace(a,b,n)     %在(a, b)上产生 n 个线性等分点
```

命令　产生对数等分向量

函数　**logspace**

格式　y = logspace(a,b)　　%在（10^a，10^b）之间产生 50 个对数等分向量

y = logspace(a,b,n)

y = logspace(a,pi)

命令　计算矩阵中元素个数

n = numel(a)　　%返回矩阵 A 的元素的个数

命令　产生以输入元素为对角线元素的矩阵

函数　**blkdiag**

格式　out = blkdiag(a,b,c,d,…)　　%产生以 a,b,c,d,…为对角线元素的矩阵

例 1-13

```
>> out = blkdiag(1,2,3,4)
out =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

1.2　矩阵运算

1.2.1　加、减运算

运算符：“＋”和“－”分别为加、减运算符。

运算规则：对应元素相加、减，即按线性代数中矩阵的“十”，“一”运算进行。

例 1-22

```
>>A=[1, 1, 1; 1, 2, 3; 1, 3, 6]
>>B=[8, 1, 6; 3, 5, 7; 4, 9, 2]
>>C=A+B
>>D=A-B
C =
     9     2     7
     4     7    10
     5    12     8
D =
    -7     0    -5
    -2    -3    -4
    -3    -6     4
```

1.2.2　乘法

运算符：*

运算规则：按线性代数中矩阵乘法运算进行，即放在前面的矩阵的各行元素，分别与放在后面的矩阵的各列元素对应相乘并相加。

1．两个矩阵相乘

例 1-23

```
>>X= [2   3   4   5; 1   2   2   1];
>>Y=[0   1   1;
     1   1   0;
     0   0   1;
     1   0   0];
Z=X*Y
结果显示为：
Z=
   8   5   6
   3   3   3
```

2．矩阵的数乘：数乘矩阵

上例中：a=2*X

则显示：a =

4 6 8 10

2 4 4 2

向量的点乘（内积）：维数相同的两个向量的点乘。

数组乘法：

A.*B 表示 A 与 B 对应元素相乘。

>> X.*X

ans =

4 9 16 25

1 4 4 1

3．向量点积

函数 dot

格式 C = dot(A,B) %若 A、B 为向量，则返回向量 A 与 B 的点积，A 与 B 长度相同；若为矩阵，则 A 与 B 有相同的维数。

C = dot(A,B,dim) %在 dim 维数中给出 A 与 B 的点积

例

```
>>X=[-1  0  2];
>>Y=[-2  -1  1];
>>Z=dot(X, Y)
```

则显示：Z =

4

还可用另一种算法：

```
sum(X.*Y)
ans=
     4
```

4．向量叉乘

在数学上，两向量的叉乘是一个过两相交向量的交点且垂直于两向量所在平面的向量。在 Matlab 中，用函数 cross 实现。

函数 cross

格式 C = cross(A,B) %若 A、B 为向量，则返回 A 与 B 的叉乘，即 C=A×B，A、B 必须是 3 个元素的向量；若 A、B 为矩阵，则返回一个 3×n 矩阵，其中的列是 A 与 B 对应列的叉积，A、B 都是 3×n 矩阵。

C = cross(A,B,dim) %在 dim 维数中给出向量 A 与 B 的叉积。A 和 B 必须具有相同的维数，size(A,dim)和 size(B,dim)必须是 3。

例 1-24 计算垂直于向量(1, 2, 3)和(4, 5, 6)的向量。

```
>>a=[1  2  3];
>>b=[4  5  6];
>>c=cross(a,b)
```

结果显示：

```
c=
    -3     6    -3
```

可得垂直于向量(1, 2, 3)和(4, 5, 6)的向量为±(-3, 6, -3)

5．混合积

混合积由以上两函数实现：

例 1-25 计算向量 a=(1, 2, 3)、b=(4, 5, 6)和 c=(-3, 6, -3) 的混合积 $a\cdot(b\times c)$

解：

```
>>a=[1  2  3]; b=[4  5  6]; c=[-3  6  -3];
>>x=dot(a, cross(b, c))
```

结果显示：x =

54

注意：先叉乘后点乘，顺序不可颠倒。

6．矩阵的卷积和多项式乘法

函数　conv

格式　w = conv(u,v)　　%u、v 为向量，其长度可不相同。

说明　长度为 m 的向量序列 u 和长度为 n 的向量序列 v 的卷积(Convolution)定义为：

$w(k)=\sum_{j=1}^{k} u(j)\,v(k+1-j)$ 式中：w 向量序列的长度为(m+n-1)，当 m=n 时，

w(1) = u(1)*v(1)
w(2) = u(1)*v(2)+u(2)*v(1)
w(3) = u(1)*v(3)+u(2)*v(2)+u(3)*v(1)
…
w(n) = u(1)*v(n)+u(2)*v(n-1)+ … +u(n)*v(1)
…
w(2*n-1) = u(n)*v(n)

例 1-26　展开多项式 $(s^2+2s+2)(s+4)(s+1)$

解：>> w=conv([1,2,2],conv([1,4],[1,1]))

```
w =
     1     7    16    18     8
>> P=poly2str(w,'s')      %将 w 表示成多项式
P =
  s^4 + 7 s^3 + 16 s^2 + 18 s + 8
```

7．反褶积（解卷）和多项式除法运算

函数　deconv

格式　[q,r] = deconv(v,u)　%多项式 v 除以多项式 u，返回商多项式 q 和余多项式 r。

注意：v、u、q、r 都是按降幂排列的多项式系数向量。

例 1-27　$(x^3+2x^2+3x+4)(10x^2+20x+30)$，则其卷积为

```
>>u = [1     2     3     4]
>>v = [10     20     30]
>>c = conv(u,v)
c =
    10    40   100   160   170   120
```

则反褶积为

```
>>[q,r] = deconv(c,u)
q =
    10    20    30
r =
     0     0     0     0     0     0
```

1.2.3　集合运算

1．两个集合的交集

函数　intersect

格式　c = intersect(a,b)　　%返回向量 a、b 的公共部分，即 c= a∩b。

c = intersect(A,B,'rows')　%A、B 为相同列数的矩阵，返回元素相同的行。

[c,ia,ib] = intersect(a,b)　%c 为 a、b 的公共元素，ia 表示公共元素在 a 中的位置，ib 表示公共元素在 b 中位置。

例 1-29

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

```
     1     2     4     6
     6     7     1     4
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
B =
     1     2     3     8
     1     1     4     6
     6     7     1     4
>> C=intersect(A,B,'rows')
C =
     6     7     1     4
```

例 1-30

```
>> A = [1 9 6 20]; B = [1 2 3 4 6 10 20];
>> [c,ia,ib] = intersect(A,B)
c =
     1     6    20
ia =
     1     3     4
ib =
     1     5     7
```

2．检测集合中的元素

函数　**ismember**

格式　k = ismember(a,S)　　%当 a 中元素属于 S 时，k 取 1，否则，k 取 0。

k = ismember(A,S,'rows')　%A、S 有相同的列，返回行相同 k 取 1，不相同取 0 的列向量。

例 1-31

```
>> S=[0   2   4   6   8   10   12   14   16   18   20];
>> a=[1   2   3   4   5   6];
>> k=ismember(a,S)
k =
     0     1     0     1     0     1     %1 表示相同元素的位置
```

例 1-32

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
>> k=ismember(A,B,'rows')
k =
     0
     0
     1          %1 表示元素相同的行
```

3．两集合的差

函数　**setdiff**

格式　c = setdiff(a,b)　　%返回属于 a 但不属于 b 的不同元素的集合，C = a-b。

c = setdiff(A,B,'rows')　%返回属于 A 但不属于 B 的不同行

[c,i] = setdiff(…)　　%c 与前面一致，i 表示 c 中元素在 A 中的位置。

例 1-33

```
>> A = [1 7 9 6 20]; B = [1 2 3 4 6 10 20];
>> c=setdiff(A,B)
c =
     7     9
```

例 1-34

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
>> c=setdiff(A,B,'rows')
c =
     1     2     3     4
     1     2     4     6
```

4．两个集合交集的非（异或）

函数　**setxor**

格式　c = setxor(a,b)　　%返回集合 a、b 交集的非

　　c = setxor(A,B,'rows')　%返回矩阵 A、B 交集的非，A、B 有相同列数。

　　[c,ia,ib] = setxor(…)　%ia、ib 表示 c 中元素分别在 a (或 A)、b(或 B)中位置

例 1-35

```
>> A=[1   2   3   4];
>> B=[2   4   5   8];
>> C=setxor(A,B)
     C =
         1     3     5     8
```

例 1-36

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
     A =
         1     2     3     4
         1     2     4     6
         6     7     1     4
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
     B =
         1     2     3     8
         1     1     4     6
         6     7     1     4
>> [C,ia,ib]=setxor(A,B,'rows')
     C =
         1     1     4     6
         1     2     3     4
         1     2     3     8
         1     2     4     6
     ia =
         1
         2
     ib =
         2
         1
```

5．两集合的并集

函数　**union**

格式　c = union(a,b)　　%返回 a、b 的并集，即 c = a∪b。

　　c = union(A,B,'rows')　%返回矩阵 A、B 不同行向量构成的大矩阵，其中相同行向量只取其一。

　　[c,ia,ib] = union(…)　%ia、ib 分别表示 c 中行向量在原矩阵(向量)中的位置

例 1-37

```
>> A=[1 2 3 4];
>> B=[2 4 5 8];
>> c=union(A,B)
```

则结果为

```
     c =
         1     2     3     4     5     8
```

例 1-38

```
>> A=[1 2 3 4;1 2 4 6]
     A =
         1     2     3     4
         1     2     4     6
>> B=[1 2 3 8;1 1 4 6]
     B =
         1     2     3     8
         1     1     4     6
>> [c,ia,ib]=union(A,B,'rows')
     c =
```

```
     1     1     4     6
     1     2     3     4
     1     2     3     8
     1     2     4     6
ia =
     1
     2
ib =
     2
     1
```

6．取集合的单值元素

函数

格式 b = unique (a) %取集合 a 的不重复元素构成的向量

b = unique (A,'rows') %返回 A、B 不同行元素组成的矩阵

[b,i,j] = unique (…) %i、j 体现 b 中元素在原向量（矩阵）中的位置

例 1-39

```
>> A=[1 1 2 2 4 4 6   4   6]
A =
     1     1     2     2     4     4     6     4     6
>> [c,i,j]=unique(A)
c =
     1     2     4     6
i =
     2     4     8     9
j =
     1     1     2     2     3     3     4     3     4
```

例 1-40

```
>> A=[1 2 2 4;1 1 4 6;1 1 4 6]
A =
     1     2     2     4
     1     1     4     6
     1     1     4     6
>> [c,i,j]=unique(A,'rows')
c =
     1     1     4     6
     1     2     2     4
i =
     3
     1
j =
     2
     1
     1
```

1.2.4 除法运算

Matlab 提供了两种除法运算：左除（\）和右除（/）。一般情况下，x=a\b 是方程 a*x =b 的解，而 x=b/a 是方程 x*a=b 的解。

例：

```
a=[1   2   3; 4   2   6; 7   4   9]
b=[4; 1; 2];
x=a\b
```

则显示：x=

```
   -1.5000
    2.0000
    0.5000
```

数组除法：

A./B 表示 A 中元素与 B 中元素对应相除。

1.2.5 矩阵乘方

运算符：^

运算规则：

（1）当 A 为方阵，P 为大于 0 的整数时，A^P 表示 A 的 P 次方，即 A 自乘 P 次；P 为小于 0 的整数时，A^P 表示 A^{-1} 的 P 次方。

（2）标量的数组乘方 P.^A，标量的数组乘方定义为 $P.^A=\begin{bmatrix} p^{a_{11}} & \cdots & p^{a_{1n}} \\ \vdots & & \vdots \\ p^{a_{m1}} & \cdots & p^{a_{mn}} \end{bmatrix}$ 数组乘方：

A.^P：表示 A 的每个元素的 P 次乘方。

1.2.7 矩阵转置

运算符：′

运算规则：若矩阵 A 的元素为实数，则与线性代数中矩阵的转置相同。

若 A 为复数矩阵，则 A 转置后的元素由 A 对应元素的共轭复数构成。

若仅希望转置，则用如下命令：A.′ 。

1.2.8 方阵的行列式

函数 **det**

格式 d = det(X)　　%返回方阵 X 的多项式的值

例 1-42

```
>> A=[1 2 3;4 5 6;7 8 9]
     A =
         1     2     3
         4     5     6
         7     8     9
>> D=det(A)
     D =
         0
```

1.2.9 逆与伪逆

命令 逆

函数 **inv**

格式 Y=inv(X)　%求方阵X的逆矩阵。若X为奇异阵或近似奇异阵，将给出警告信息。

例 1-43 求 $A=\begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 4 & 3 \end{pmatrix}$ 的逆矩阵

方法一

```
>>A=[1   2   3; 2   2   1; 3   4   3];
>>Y=inv(A)或 Y=A^(-1)
则结果显示为
    Y =
        1.0000    3.0000   -2.0000
       -1.5000   -3.0000    2.5000
        1.0000    1.0000   -1.0000
```

方法二：由增广矩阵 $B=\begin{pmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 & 1 & 0 \\ 3 & 4 & 3 & 0 & 0 & 1 \end{pmatrix}$ 进行初等行变换

```
>>B=[1, 2, 3, 1, 0, 0; 2, 2, 1, 0, 1, 0; 3, 4, 3, 0, 0, 1];
```

```
>>C=rref(B)        %化行最简形
>>X=C(:, 4:6)      %取矩阵 C 中的 A^(-1)部分
显示结果如下：
  C =
       1.0000            0            0       1.0000       3.0000      -2.0000
            0       1.0000            0      -1.5000      -3.0000       2.5000
            0            0       1.0000       1.0000       1.0000      -1.0000
  X =
       1.0000       3.0000      -2.0000
      -1.5000      -3.0000       2.5000
       1.0000       1.0000      -1.0000
```

例 1-44

```
>> A=[2 1 -1;2 1 2;1 -1 1];
>> format rat       %用有理格式输出
>> D=inv(A)
     D =
          1/3             0           1/3
            0           1/3          -2/3
         -1/3           1/3             0
```

命令　伪逆

函数　**pinv**

格式　B = pinv(A)　　%求矩阵 A 的伪逆

　　B = pinv(A, tol)　%tol 为误差：max(size(A))*norm(A)*eps

说明　当矩阵为长方阵时，方程 AX=I 和 XA=I 至少有一个无解，这时 A 的伪逆能在某种程度上代表矩阵的逆，若 A 为非奇异矩阵，则 pinv(A) = inv(A)。

例 1-45

```
>> A=magic(5);     %产生 5 阶魔方阵。
>> A=A(:,1:4)      %取 5 阶魔方阵的前 4 列元素构成矩阵 A。
     A =
          17      24       1       8
          23       5       7      14
           4       6      13      20
          10      12      19      21
          11      18      25       2
>> X=pinv(A)       %计算 A 的伪逆
     X =
         -0.0041       0.0527      -0.0222      -0.0132       0.0069
          0.0437      -0.0363       0.0040       0.0033       0.0038
         -0.0305       0.0027      -0.0004       0.0068       0.0355
          0.0060      -0.0041       0.0314       0.0211      -0.0315
```

1.2.11　矩阵和向量的范数

命令　向量的范数

函数　**norm**

格式　n = norm(X)　　%X 为向量，求欧几里德范数，即 $\|X\|_2=\sqrt{\sum|x_k|^2}$。

　　n = norm(X,inf)　%求 ∞ -范数，即 $\|X\|=\max(\mathrm{abs}(X))$。

　　n = norm(X,1)　　%求 1-范数，即 $\|X\|_1=\sum_k|x_k|$。

　　n = norm(X,-inf)　%求向量 X 的元素的绝对值的最小值，即 $\|X\|=\min(\mathrm{abs}(X))$。

　　n = norm(X, p)　　%求 p-范数，即 $\|X\|_p=\sqrt[p]{\sum_k|x_k|^p}$，所以 norm(X,2) = norm(X)。

1.2.13 矩阵的秩

函数 **rank**

格式 k = rank (A) %求矩阵 A 的秩

k = rank (A,tol) %tol 为给定误差

1.2.14 特殊运算

1．矩阵对角线元素的抽取

函数 **diag**

格式 X = diag(v,k) %以向量 v 的元素作为矩阵 X 的第 k 条对角线元素，当 k=0 时，v 为 X 的主对角线；当 k>0 时，v 为上方第 k 条对角线；当 k<0 时，v 为下方第 k 条对角线。

X = diag(v) %以 v 为主对角线元素，其余元素为 0 构成 X。

v = diag(X,k) %抽取 X 的第 k 条对角线元素构成向量 v。k=0：抽取主对角线元素；k>0：抽取上方第 k 条对角线元素；k<0 抽取下方第 k 条对角线元素。

v = diag(X) %抽取主对角线元素构成向量 v。

例 1-46

```
>> v=[1 2 3];
>> x=diag(v,-1)
     x =
         0     0     0     0
         1     0     0     0
         0     2     0     0
         0     0     3     0
>> A=[1 2 3;4 5 6;7 8 9]
     A =
         1     2     3
         4     5     6
         7     8     9
>> v=diag(A,1)
     v =
         2
         6
```

2．上三角阵和下三角阵的抽取

函数 **tril** %取下三角部分

格式 L = tril(X) %抽取 X 的主对角线的下三角部分构成矩阵 L

L = tril(X,k) %抽取 X 的第 k 条对角线的下三角部分；k=0 为主对角线；k>0 为主对角线以上；k<0 为主对角线以下。

函数 **triu** %取上三角部分

格式 U = triu(X) %抽取 X 的主对角线的上三角部分构成矩阵 U

U = triu(X,k) %抽取 X 的第 k 条对角线的上三角部分；k=0 为主对角线；k>0 为主对角线以上；k<0 为主对角线以下。

例 1-47

```
>> A=ones(4)      %产生 4 阶全 1 阵
     A =
         1     1     1     1
         1     1     1     1
         1     1     1     1
         1     1     1     1
>> L=tril(A,1)      %取下三角部分
     L =
```

```
     1     1     0     0
     1     1     1     0
     1     1     1     1
     1     1     1     1
>> U=triu(A,-1)      %取上三角部分
   U =
     1     1     1     1
     1     1     1     1
     0     1     1     1
     0     0     1     1
```

6．矩阵元素的数据变换

对于小数构成的矩阵 A 来说，如果我们想对它取整数，有以下几种方法：

（1）按-∞方向取整

函数　**floor**

格式　floor(A)　%将 A 中元素按-∞方向取整，即取不足整数。

（2）按+∞方向取整

函数　**ceil**

格式　ceil(A)　%将 A 中元素按+∞方向取整，即取过剩整数。

（3）四舍五入取整

函数　**round**

格式　round (A)　%将 A 中元素按最近的整数取整，即四舍五入取整。

（4）按离 0 近的方向取整

函数　**fix**

格式　fix (A)　%将 A 中元素按离 0 近的方向取整

例 1-55

```
>> A=-1.5+4*rand(3)
   A =
      2.3005    0.4439    0.3259
     -0.5754    2.0652   -1.4260
      0.9274    1.5484    1.7856
>> B1=floor(A),B2=ceil(A),B3=round(A),B4=fix(A)
   B1 =
      2     0     0
     -1     2    -2
      0     1     1
   B2 =
      3     1     1
      0     3    -1
      1     2     2
   B3 =
      2     0     0
     -1     2    -1
      1     2     2
   B4 =
      2     0     0
      0     2    -1
      0     1     1
```

1.2.16　矩阵元素个数的确定

函数　**numel**

格式　n = numel(a)　%计算矩阵 A 中元素的个数

例 1-65

```
>> A=[1 2 3 4;5 6 7 8];
>> n=numel(A)
```

```
n =
     8
```

1.3 矩阵分解（先了解用到时再研究）

1.3.1 Cholesky 分解

函数 **chol** A 的全部顺序主子式>0 A 能够作 Cholesky 分解的充要条件

格式 R = chol(X) %如果 X 为 n 阶对称正定矩阵（A 的特征值全为正等），则存在一个实的非奇异上三角阵 R，满足 R'*R = X；若 X 非正定，则产生错误信息。

[R,p] = chol(X) %不产生任何错误信息，若 X 为正定阵，则 p=0，R 与上相同；若 X 非正定，则 p 为正整数，R 是有序的上三角阵。

例 1-66

```
>> X=pascal(4)      %产生 4 阶 pascal 矩阵
     X =
         1     1     1     1
         1     2     3     4
         1     3     6    10
         1     4    10    20
>> [R,p]=chol(X)
     R =
         1     1     1     1
         0     1     2     3
         0     0     1     3
         0     0     0     1
     p =
         0
```

1.3.2 LU 分解

矩阵的三角分解又称 LU 分解，它的目的是将一个矩阵分解成一个下三角矩阵 L 和一个上三角矩阵 U 的乘积，即 A=LU。(当 A 的所有顺序主子式都不为 0 时，矩阵 A 可以分解为 A=LU)

函数 **lu**

格式 [L,U] = lu(X) %U 为上三角阵，L 为下三角阵或其变换形式，满足 LU=X。

[L,U,P] = lu(X) %U 为上三角阵，L 为下三角阵，P 为单位矩阵的行变换矩阵，满足 LU=PX。

例 1-67

```
>> A=[1 2 3;4 5 6;7 8 9];
>> [L,U]=lu(A)
     L =
         0.1429    1.0000         0
         0.5714    0.5000    1.0000
         1.0000         0         0
     U =
         7.0000    8.0000    9.0000
              0    0.8571    1.7143
              0         0    0.0000
>> [L,U,P]=lu(A)
     L =
         1.0000         0         0
         0.1429    1.0000         0
         0.5714    0.5000    1.0000
     U =
```

```
    7.0000    8.0000    9.0000
         0    0.8571    1.7143
         0         0    0.0000
P =
     0     0     1
     1     0     0
     0     1     0
```

1.3.3 QR 分解

将矩阵 A 分解成一个正交矩阵 Q 与一个上三角矩阵的乘积 R。A=QR, 原矩阵 A 不必为正方矩阵；如果矩阵 A 大小为 n*m，则矩阵 Q 大小为 n*n，矩阵 R 大小为 n*m。

函数 **qr**

格式 [Q,R] = qr(A) %求得正交矩阵 Q 和上三角阵 R，Q 和 R 满足 A=QR。

[Q,R,E] = qr(A) %求得正交矩阵 Q 和上三角阵 R，E 为单位矩阵的变换形式，R 的对角线元素按大小降序排列，满足 AE=QR。

[Q,R] = qr(A,0) %产生矩阵 A 的“经济大小”分解

[Q,R,E] = qr(A,0) %E 的作用是使得 R 的对角线元素降序，且 Q*R=A(:, E)。

R = qr(A) %稀疏矩阵 A 的分解，只产生一个上三角阵 R，满足 R'*R = A'*A，这种方法计算 A'*A 时减少了内在数字信息的损耗。

[C,R] = qr(A,b) %用于稀疏最小二乘问题：minimize||Ax-b||的两步解：[C,R] = qr(A,b)，x = R\c。

R = qr(A,0) %针对稀疏矩阵 A 的经济型分解

[C,R] = qr(A,b,0) %针对稀疏最小二乘问题的经济型分解

例 1-68

```
>>A =[ 1    2    3;4    5    6; 7    8    9; 10    11    12];
>>[Q,R] = qr(A)
    Q =
       -0.0776    -0.8331     0.5444     0.0605
       -0.3105    -0.4512    -0.7709     0.3251
       -0.5433    -0.0694    -0.0913    -0.8317
       -0.7762     0.3124     0.3178     0.4461
    R =
       -12.8841    -14.5916    -16.2992
              0     -1.0413     -2.0826
              0           0      0.0000
              0           0           0
```

1.3.4 Schur 分解

函数 **schur**

格式 T = schur(A) %产生 schur 矩阵 T，即 T 的主对角线元素为特征值的三角阵。

T = schur(A,flag) %若 A 有复特征根，则 flag='complex'，否则 flag='real'。

[U,T] = schur(A,…) %返回正交矩阵 U 和 schur 矩阵 T，满足 A = U*T*U'。

例 1-71

```
>> H = [ -149   -50   -154; 537   180   546; -27   -9   -25 ];
>> [U,T]=schur(H)
    U =
        0.3162    -0.6529     0.6882
       -0.9487    -0.2176     0.2294
        0.0000     0.7255     0.6882
    T =
        1.0000    -7.1119 -815.8706
             0     2.0000   -55.0236
             0          0     3.0000
```

1.3.6 特征值分解

函数 **eig**

格式 d = eig(A) %求矩阵 A 的特征值 d，以向量形式存放 d。

d = eig(A,B) %A、B 为方阵，求广义特征值 d，以向量形式存放 d。

[V,D] = eig(A) %计算 A 的特征值对角阵 D 和特征向量 V，使 AV=VD 成立。

[V,D] = eig(A,'nobalance') %当矩阵 A 中有与截断误差数量级相差不远的值时，该指令可能更精确。'nobalance'起误差调节作用。

[V,D] = eig(A,B) %计算广义特征值向量阵 V 和广义特征值阵 D，满足 AV=BVD。

[V,D] = eig(A,B,flag) % 由 flag 指定算法计算特征值 D 和特征向量 V，flag 的可能值为：'chol' 表示对 B 使用 Cholesky 分解算法，这里 A 为对称 Hermitian 矩阵，B 为正定阵。'qz' 表示使用 QZ 算法，这里 A、B 为非对称或非 Hermitian 矩阵。

说明 一般特征值问题是求解方程： $Ax=\lambda x$ 解的问题。广义特征值问题是求方程： $Ax=\lambda Bx$ 解的问题。

```
>> A=[1 2 3;4 5 6;7 8 9];
>> d = eig(A)
d =
    16.1168
    -1.1168
    -0.0000
```

1.4 线性方程的组的求解

我们将线性方程的求解分为两类：一类是方程组求唯一解或求特解，另一类是方程组求无穷解即通解。可以通过系数矩阵的秩来判断：

若系数矩阵的秩 r=n（n 为方程组中未知变量的个数），则有唯一解；

若系数矩阵的秩 r<n，则可能有无穷解；

线性方程组的无穷解 = 对应齐次方程组的通解+非齐次方程组的一个特解；**其特解的求法属于解的第一类问题，通解部分属第二类问题。**

1.4.1 求线性方程组的唯一解或特解（第一类问题）

这类问题的求法分为两类：一类主要用于解低阶稠密矩阵 —— 直接法；另一类是解大型稀疏矩阵 —— 迭代法。

1．利用矩阵除法求线性方程组的特解（或一个解）

方程：AX=b

解法：X=A\b

例 1-76 求方程组
$$\begin{cases} 5x_1+6x_2 = 1 \\ x_1+5x_2+6x_3 = 0 \\ x_2+5x_3+6x_4 = 0 \\ x_3+5x_4+6x_5 = 0 \\ x_4+5x_5 = 1 \end{cases}$$
的解。

解：

```
>>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]';
R_A=rank(A)     %求秩
X=A\B       %求解
```

运行后结果如下

```
R_A =
     5
X =
    2.2662
   -1.7218
    1.0571
   -0.5940
    0.3188
```

这就是方程组的解。

或用函数 rref 求解：

```
>> C=[A,B]     %由系数矩阵和常数列构成增广矩阵 C
>> R=rref(C)     %将 C 化成行最简行
    R =
         1.0000         0         0         0         0    2.2662
              0    1.0000         0         0         0   -1.7218
              0         0    1.0000         0         0    1.0571
              0         0         0    1.0000         0   -0.5940
              0         0         0         0    1.0000    0.3188
```

则 R 的最后一列元素就是所求之解。

例 1-77　求方程组 $\begin{cases} x_1 + x_2 - 3x_3 - x_4 = 1 \\ 3x_1 - x_2 - 3x_3 + 4x_4 = 4 \\ x_1 + 5x_2 - 9x_3 - 8x_4 = 0 \end{cases}$ 的一个特解。

解：

```
>>A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
>>B=[1    4    0]';
>>X=A\B     %由于系数矩阵不满秩，该解法可能存在误差。
    X =[ 0    0    -0.5333    0.6000]'（一个特解近似值）。
```

若用 rref 求解，则比较精确：

```
>> A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
    B=[1    4    0]';
>> C=[A,B];      %构成增广矩阵
>> R=rref(C)
    R =
         1.0000         0   -1.5000    0.7500    1.2500
              0    1.0000   -1.5000   -1.7500   -0.2500
              0         0         0         0         0
```

由此得解向量 X=[1.2500　−0.2500　0　0]'（一个特解）。

2．利用矩阵的 LU、QR 和 cholesky 分解求方程组的解

（1）LU 分解：

LU 分解又称 Gauss 消去分解，可把任意方阵分解为下三角矩阵的基本变换形式（行交换）和上三角矩阵的乘积。即 A=LU，L 为下三角阵，U 为上三角阵。

则：A*X=b　　变成 L*U*X=b

所以 X=U\(L\b)　这样可以大大提高运算速度。

命令　[L，U]=lu (A)

例 1-78　求方程组 $\begin{cases} 4x_1 + 2x_2 - x_3 = 2 \\ 3x_1 - x_2 + 2x_3 = 10 \\ 11x_1 + 3x_2 = 8 \end{cases}$ 的一个特解。

解：

$$A=\begin{pmatrix}4 & 2 & -1\\ 3 & -1 & 2\\ 11 & 3 & 0\end{pmatrix} \qquad b=[2,10,8]'$$

```
>>A=[4 2 -1;3 -1 2;11 3 0];
>>B=[2 10 8]';
>>D=det(A)
>>[L,U]=lu(A)
>>X=U\(L\B)
```

显示结果如下：

```
D =
     0
L =
    0.3636   -0.5000    1.0000
    0.2727    1.0000         0
    1.0000         0         0
U =
   11.0000    3.0000         0
         0   -1.8182    2.0000
         0         0    0.0000
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.018587e-017.
> In D:\Matlab\pujun\lx0720.m at line 4
X =
   1.0e+016 *
   -0.4053
    1.4862
    1.3511
```

说明 结果中的警告是由于系数行列式为零产生的。可以通过 A*X 验证其正确性。

（2）Cholesky 分解

若 A 为对称正定矩阵，则 Cholesky 分解可将矩阵 A 分解成上三角矩阵和其转置的乘积，即：$A = R' * R$ 其中 R 为上三角阵。

方程 A*X=b 变成 $R' * R * X = b$

所以 $X = R \setminus (R' \setminus b)$

（3）QR 分解

对于任何长方矩阵 A，都可以进行 QR 分解，其中 Q 为正交矩阵，R 为上三角矩阵的初等变换形式，即：A=QR

方程 A*X=b 变形成 QRX=b

所以 X=R\(Q\b)

上例中 [Q, R]=qr(A)

X=R\(Q\B)

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

1.4.2 求线性齐次方程组的通解

在 Matlab 中，函数 null 用来求解零空间，即满足 A・X=0 的解空间，实际上是求出解空间的一组基（基础解系）。

格式 z = null % z 的列向量为方程组的正交规范基，满足 $Z' \times Z = I$。

z = null(A,'r') % z 的列向量是方程 AX=0 的有理基

例 1-79　求解方程组的通解：$\begin{cases} x_1 + 2x_2 + 2x_3 + x_4 = 0 \\ 2x_1 + x_2 - 2x_3 - 2x_4 = 0 \\ x_1 - x_2 - 4x_3 - 3x_4 = 0 \end{cases}$

解：

```
>>A=[1   2   2   1;2   1   -2   -2;1   -1   -4   -3];
>>format   rat        %指定有理式格式输出
>>B=null(A,'r')       %求解空间的有理基
```

运行后显示结果如下：

```
B =
     2      5/3
    -2     -4/3
     1      0
     0      1
```

即可写出其基础解系。

写出通解：

```
syms   k1   k2
X=k1*B(:,1)+k2*B(:,2)          %写出方程组的通解
pretty(X)        %让通解表达式更加精美
```

运行后结果如下：

```
X =
[   2*k1+5/3*k2]
[ -2*k1-4/3*k2]
[              k1]
[              k2]
 % 下面是其简化形式
[2k1 + 5/3k2 ]
[            ]
[-2k1 - 4/3k2]
[            ]
[      k1    ]
[            ]
[      k2    ]
```

或通过行最简行得到基：

```
>> B=rref(A)
    B =
         1.0000         0   -2.0000   -1.6667
              0    1.0000    2.0000    1.3333
              0         0         0         0
```

1.4.3　求非齐次线性方程组的通解

非齐次线性方程组需要先判断方程组是否有解，若有解，再去求通解。

因此，步骤为：

第一步：判断 AX=b 是否有解，若有解则进行第二步

第二步：求 AX=b 的一个特解

第三步：求 AX=0 的通解

第四步：AX=b 的通解= AX=0 的通解+AX=b 的一个特解。

例 1-80　求解方程组$\begin{cases} x_1 - 2x_2 + 3x_3 - x_4 = 1 \\ 3x_1 - x_2 + 5x_3 - 3x_4 = 2 \\ 2x_1 + x_2 + 2x_3 - 2x_4 = 3 \end{cases}$

解：在 Matlab 中建立 M 文件如下：

```
A=[1   -2   3   -1;3   -1   5   -3;2   1   2   -2];
b=[1   2   3]';
B=[A b];
n=4;
R_A=rank(A)
```

```
R_B=rank(B)
format rat
if R_A==R_B&R_A==n          %判断有唯一解
    X=A\b
elseif R_A==R_B&R_A<n       %判断有无穷解
    X=A\b          %求特解
    C=null(A,'r')       %求 AX=0 的基础解系
else X='equition no solve'         %判断无解
end
```

运行后结果显示：

```
R_A =
     2
R_B =
     3
X =
     equition no solve
```

说明　该方程组无解

例 1-81　求解方程组的通解：$\begin{cases} x_1+x_2-3x_3-x_4=1 \\ 3x_1-x_2-3x_3+4x_4=4 \\ x_1+5x_2-9x_3-8x_4=0 \end{cases}$

解法一：在 Matlab 编辑器中建立 M 文件如下：

```
A=[1   1   -3   -1;3   -1   -3   4;1   5   -9   -8];
b=[1 4 0]';
B=[A b];
n=4;
R_A=rank(A)
R_B=rank(B)
format rat
if R_A==R_B&R_A==n
   X=A\b
elseif R_A==R_B&R_A<n
   X=A\b
   C=null(A,'r')
else X='Equation has no solves'
end
```

运行后结果显示为：

```
R_A =
     2
R_B =
     2
Warning: Rank deficient, rank = 2    tol =      8.8373e-015.
> In D:\Matlab\pujun\lx0723.m at line 11
X =
     0
     0
     -8/15
     3/5
C =
     3/2              -3/4
     3/2               7/4
     1                 0
     0                 1
```

所以原方程组的通解为 $X=k_1\begin{pmatrix} 3/2 \\ 3/2 \\ 1 \\ 0 \end{pmatrix}+k_2\begin{pmatrix} -3/4 \\ 7/4 \\ 0 \\ 1 \end{pmatrix}+\begin{pmatrix} 0 \\ 0 \\ -8/15 \\ 3/5 \end{pmatrix}$

解法二：用 rref 求解

```
A=[1   1   -3   -1;3   -1   -3   4;1   5   -9   -8];
```

```
b=[1 4 0]';
B=[A b];
C=rref(B)      %求增广矩阵的行最简形，可得最简同解方程组。
```

运行后结果显示为：

```
C =
    1     0    -3/2     3/4     5/4
    0     1    -3/2    -7/4    -1/4
    0     0     0       0       0
```

对应齐次方程组的基础解系为：$\xi_1=\begin{pmatrix}3/2\\3/2\\1\\0\end{pmatrix}$，$\xi_2=\begin{pmatrix}-3/4\\7/4\\0\\1\end{pmatrix}$ 非齐次方程组的特解为：$\eta^*=\begin{pmatrix}5/4\\-1/4\\0\\0\end{pmatrix}$ 所以，原方程组的通解为：$X=k_1\xi_1+k_2\xi_2+\eta^*$。

1.5 特征值与二次型

工程技术中的一些问题，如振动问题和稳定性问题，常归结为求一个方阵的特征值和特征向量。

1.5.1 特征值与特征向量的求法

设 A 为 n 阶方阵，如果数“λ”和 n 维列向量 x 使得关系式 $Ax=\lambda x$ 成立，则称λ为方阵 A 的特征值，非零向量 x 称为 A 对应于特征值“λ”的特征向量。

详见 1.3.5 和 1.3.6 节：特征值分解问题。

例 1-89 求矩阵 $A=\begin{pmatrix}-2&1&1\\0&2&0\\-4&1&3\end{pmatrix}$ 的特征值和特征向量

解：

```
>>A=[-2   1   1;0   2   0;-4   1   3];
>>[V,D]=eig(A)
```

结果显示：

```
V =
   -0.7071   -0.2425    0.3015
         0         0    0.9045
   -0.7071   -0.9701    0.3015
D =
   -1    0    0
    0    2    0
    0    0    2
```

即：特征值-1 对应特征向量$(-0.7071\quad 0\quad -0.7071)^T$

特征值 2 对应特征向量$(-0.2425\quad 0\quad -0.9701)^T$ 和$(-0.3015\quad 0.9045\quad -0.3015)^T$

例 1-90 求矩阵 $A=\begin{pmatrix}-1&1&0\\-4&3&0\\1&0&2\end{pmatrix}$ 的特征值和特征向量。

解：

```
>>A=[-1 1 0;-4 3 0;1 0 2];
>>[V,D]=eig(A)
```

结果显示为

```
V =
     0         0.4082   -0.4082
     0         0.8165   -0.8165
1.0000        -0.4082    0.4082
D =
     2     0     0
     0     1     0
     0     0     1
```

说明　当特征值为 1 (二重根)时，对应特征向量都是 k (0.4082　0.8165　-0.4082)^T，k 为任意常数。

1.5.4　正交基

命令　orth

格式　B=orth(A)　%将矩阵 A 正交规范化，B 的列与 A 的列具有相同的空间，B 的列向量是正交向量，且满足：B'*B = eye(rank(A))。

例 1-92　将矩阵 $A=\begin{pmatrix}4&0&0\\0&3&1\\0&1&3\end{pmatrix}$ 正交规范化。

解：

```
>>A=[4   0   0; 0   3   1; 0   1   3];
>>B=orth(A)
>>Q=B'*B
```

则显示结果为

```
B =
1.0000         0         0
     0    0.7071   -0.7071
     0    0.7071    0.7071
Q =
1.0000         0         0
     0    1.0000         0
     0         0    1.0000
```

1.5.5　二次型

例 1-93　求一个正交变换 X=PY，把二次型

$f = 2x_1x_2 + 2x_1x_3 - 2x_1x_4 - 2x_2x_3 + 2x_2x_4 + 2x_3x_4$ 化成标准形。

解：先写出二次型的实对称矩阵

$$A=\begin{pmatrix}0&1&1&-1\\1&0&-1&1\\1&-1&0&1\\-1&1&1&0\end{pmatrix}$$

在 Matlab 编辑器中建立 M 文件如下：

```
A=[0 1 1 -1;1 0 -1 1;1 -1 0 1;-1 1 1 0];
[P,D]=schur(A)          % 返回正交矩阵 P 和 schur 矩阵 D，满足 A = P*D*P', 使得 P^-1AP=P^TAP=diag(λ1, λ2,..., λn)
syms y1 y2   y3 y4
y=[y1;y2;y3;y4] ;
X=vpa(P,2)*y            %vpa 表示可变精度计算，这里取 2 位精度
f=[y1 y2 y3 y4]*D*y
```

运行后结果显示如下：

```
P =
 780/989        780/3691         1/2         -390/1351
 780/3691       780/989         -1/2          390/1351
 780/1351      -780/1351        -1/2          390/1351
    0              0             1/2          1170/1351
```

```
D =
     1     0     0     0
     0     1     0     0
     0     0    -3     0
     0     0     0     1
X =
    [ .79*y1+.21*y2+.50*y3-.29*y4]
    [ .21*y1+.79*y2-.50*y3+.29*y4]
    [ .56*y1-.56*y2-.50*y3+.29*y4]
    [                .50*y3+.85*y4]
f =
    y1^2+y2^2-3*y3^2+y4^2
```

即 $f = y_1^2 + y_2^2 - 3y_3^2 + y_4^2$

1.6 秩与线性相关性

1.6.1 矩阵和向量组的秩以及向量组的线性相关性

矩阵 A 的秩是矩阵 A 中最高阶非零子式的阶数；向量组的秩通常由该向量组构成的矩阵来计算。

函数 rank

格式 k = rank(A) %返回矩阵 A 的行（或列）向量中线性无关个数

k = rank(A,tol) %tol 为给定误差

例 1-94 求向量组 $\alpha_1=(1\quad -2\quad 2\quad 3)$，$\alpha_2=(-2\quad 4\quad -1\quad 3)$，$\alpha_3=(-1\quad 2\quad 0\quad 3)$，$\alpha_4=(0\quad 6\quad 2\quad 3)$，$\alpha_5=(2\quad -6\quad 3\quad 4)$ 的秩，并判断其线性相关性。

```
>>A=[1 -2 2 3;-2 4 -1 3;-1 2 0 3;0 6 2 3;2 -6 3 4];
>>k=rank(A)
```

结果为

```
k =
     3
```

由于秩为 3 < 向量个数，因此向量组线性相关。

1.6.2 求行阶梯矩阵及向量组的基

行阶梯使用初等行变换，矩阵的初等行变换有三条：

1．交换两行 $r_i \leftrightarrow r_j$ （第 i、第 j 两行交换）

2．第 i 行的 K 倍 $k\,r_i$

3．第 i 行的 K 倍加到第 j 行上去 $r_j + k\,r_i$

通过这三条变换可以将矩阵化成行最简形，从而找出列向量组的一个最大无关组，Matlab 将矩阵化成行最简形的命令是 rref 或 rrefmovie。

函数 rref 或 rrefmovie

格式 R = rref(A) %用高斯—约当消元法和行主元法求 A 的行最简行矩阵 R

[R,jb] = rref(A) %jb 是一个向量，其含义为：r = length(jb)为 A 的秩；A(:, jb)为 A 的列向量基；jb 中元素表示基向量所在的列。

[R,jb] = rref(A,tol) %tol 为指定的精度

rrefmovie(A) %给出每一步化简的过程

例 1-95 求向量组 a1=(1,-2,2,3)，a2=(-2,4,-1,3)，a3=(-1,2,0,3)，a4=(0,6,2,3)，a5=(2,-6,3,4)的一个最大无关组。

```
>> a1=[1   -2   2   3]';
>>a2=[-2   4   -1   3]';
```

```
>>a3=[-1    2    0    3]';
>>a4=[0    6    2    3]';
>>a5=[2    -6    3    4]';
A=[a1    a2    a3    a4    a5]
A =
     1    -2    -1     0     2
    -2     4     2     6    -6
     2    -1     0     2     3
     3     3     3     3     4
>> [R,jb]=rref(A)
R =
    1.0000         0    0.3333         0    1.7778
         0    1.0000    0.6667         0   -0.1111
         0         0         0    1.0000   -0.3333
         0         0         0         0         0
jb =
     1     2     4
>> A(:,jb)
ans =
     1    -2     0
    -2     4     6
     2    -1     2
     3     3     3
```

即：a1　a2　a4 为向量组的一个基。