

第4章 创建新矩阵

在2.2节中讨论了矩阵的定义和分配。但是也还有可能要建立新的矩阵，例如通过函数返回一个新的矩阵或者使用已存在的矩阵。

4.1 建立新矩阵

建立1矩阵使用ones命令，这种矩阵的元素全部都是1。相应的建立0矩阵使用zeros命令，这种矩阵的元素全部都是0。单位矩阵的对角线元素全部是1，而其他元素全部是0。建立单位矩阵使用eye命令。在矩阵乘方运算中， n 阶的单位矩阵就相对应于在标量运算中的数字1。

命令集32 1矩阵、零矩阵和单位矩阵

ones(n)	建立一个 $n \times n$ 的1矩阵。
ones(m,n,...,p)	建立一个 $m \times n \times \dots \times p$ 的1矩阵。
ones(size(A))	建立一个和矩阵A同样大小的1矩阵。
zeros(n)	建立一个 $n \times n$ 的0矩阵。
zeros(m,n,...,p)	建立一个 $m \times n \times \dots \times p$ 的0矩阵。
zeros(size(A))	建立一个和矩阵A同样大小的0矩阵。
eye(n)	建立一个 $n \times n$ 的单位矩阵。注意eye命令只能用来建立二维矩阵。
eye(m, n)	建立一个 $m \times n$ 的单位矩阵。注意eye命令只能用来建立二维矩阵。
eye(size(A))	建立一个和矩阵A同样大小的单位矩阵。

例4.1

命令组：

```
OneMatrix = ones(3,4,2)
ZeroMatrix = zeros(size(OneMatrix))
Identity = eye(2)
Identity23 = eye(2,3)
Identity32 = eye(3,2)
```

运行上述命令后在屏幕上会得到以下结果：

```
OneMatrix(:,:,1) =
    1    1    1    1
    1    1    1    1
    1    1    1    1
```

```
OneMatrix(:,:,2) =
    1    1    1    1
    1    1    1    1
    1    1    1    1
```

```
ZeroMatrix(:,:,1) =
```

```
0    0    0    0
0    0    0    0
0    0    0    0
```

```
ZeroMatrix(:, :, 2) =
```

```
0    0    0    0
0    0    0    0
0    0    0    0
```

```
Identity =
```

```
1    0
0    1
```

```
Identity23 =
```

```
1    0    0
0    1    0
```

```
Identity32 =
```

```
1    0
0    1
0    0
```

矩阵中的所有元素都是随机数，这样的矩阵称为随机矩阵。可以用 `rand` 命令来产生在 0 ~ 1 之间均匀分布的随机数。也可以用 MATLAB 中的 `randn` 命令来产生服从零均值、单位方差正态分布的随机数。

命令集33 随机数和随机矩阵

<code>rand</code>	产生在 0 ~ 1 之间均匀分布的随机数；每调用一次给一个新的数值。
<code>rand + i*rand</code>	产生一个复数随机数。
<code>rand(n)</code>	产生一个 $n \times n$ 的矩阵，其元素为 0 ~ 1 之间均匀分布的随机数。
<code>rand(m,n,...,p)</code>	产生一个 $m \times n \times \dots \times p$ 的矩阵，其元素为 0 ~ 1 之间均匀分布的随机数。
<code>rand</code>	产生零均值、单位方差的正态分布随机数。
<code>randn(n)</code>	产生一个 $n \times n$ 的矩阵，其元素为零均值、单位方差的正态分布随机数。
<code>randn(m,n,...,p)</code>	产生一个 $m \times n \times \dots \times p$ 的矩阵，其元素为零均值、单位方差的正态分布随机数。

MATLAB 5 使用一种新的随机数发生器，可以设置几个随机数的种子。它能够产生在闭区间 $[2^{-53}, 1 - 2^{-53}]$ 上所有的浮点数。理论上它能够产生 $2^{1492} - 10^{449}$ 多个不重复的数。而 MATLAB 4 中的随机数发生器只能使用一个随机数的种子。

命令集34 随机数种子

<code>rand('state')</code>	返回一个有 5 个元素的向量，其中包含随机状态发生器的当前状态。
<code>rand('state',s)</code>	设置随机种子发生器的状态为 s 。

<code>rand('state',0)</code>	设置随机种子发生器为它的原始状态。
<code>rand('state', j)</code>	设置随机种子发生器为它的第 j 种状态, j 为整数。
<code>rand('state', sum(100*clock))</code>	使用 <code>clock</code> 命令(见命令集15), 使得随机种子发生器在每个不 同时刻都设置为一种不同的状态。
<code>rand('seed',arg)</code>	使用 MATLAB 4 中的随机种子发生器, 见帮助可以得到更多的 信息。
<code>randn('state')</code>	返回一个有两个元素的向量, 其中包含正态随机种子发生器的 状态。
<code>randn('state',arg)</code>	根据 <code>arg</code> 设置正态随机种子发生器, 见 <code>rand</code> 。

例4.2

(a) 例如, 随机种子发生器可以给出以下的结果。这里只列出了这个状态向量 (35个元素) 中的前五个元素的值。

```

astate = rand('state'); astate(1:5), Random = rand(2,3)
ans =
    0.6923
    0.1646
    0.5676
    0.3609
    0.8557
Random =
    0.4565    0.8214    0.6154
    0.0185    0.4447    0.7919

```

(b) 为了避免总是从相同的随机种子开始而得到相同的随机数序列, 可以使用 MATLAB 中的 `clock` 函数。

```

rand('state',sum(100*clock)); R = rand('state'); R(1:5)
ans =
    0.8010
    0.4701
    0.5052
    0.0707
    0.4643

```

`clock` 命令定义在 2.5 节中。

在 MATLAB 中还有利用已存在的矩阵建立新矩阵的命令。假设矩阵 A 是 $m \times n$ 的矩阵, x 是一个有 n 个元素的向量。用命令集 35 中的命令 `diag` 来生成一个新的矩阵。

命令集 35 从已存在的矩阵中生成新的矩阵 (一)

<code>diag(A)</code>	生成一个由矩阵 A 主对角线元素组成的列向量。主对角线总是从矩阵左上角开始。对于方阵来说它结束于矩阵的右下角。
----------------------	---

<code>diag(x)</code>	生成一个 n 维的方阵，它的主对角线元素值取自向量 x ，其余元素的值都为0。
<code>diag(A,k)</code>	生成一个由矩阵 A 第 k 条对角线的元素组成的列向量。 $k=0$ 为主对角线； $k<0$ 为下第 k 对角线； $k>0$ 为上第 k 对角线。
<code>diag(x,k)</code>	生成一个 $(n+\text{abs}(k)) \times (n+\text{abs}(k))$ 维的矩阵，该矩阵的第 k 条对角线元素取自向量 x ，其余元素都为零。关于参数 k 可参考上个命令。

例4.3

假设：

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \quad x = (-5 \quad -10 \quad -15)$$

(a) 命令 `diag_element=diag(A)` 给出：

```
diag_element =
     1
     6
    11
    16
```

(b) `Diag_matrix=diag(diag(A))` 返回：

```
Diag_matrix =
     1     0     0     0
     0     6     0     0
     0     0    11     0
     0     0     0    16
```

(c) 命令 `Dmatrixx=diag(x)` 或者 `Dmatrixx=diag(x')` 给出：

```
Dmatrixx =
    -5     0     0
     0    -10     0
     0     0    -15
```

(d) 如果输入 `superDiagElement=diag(A,2)`，那么输出：

```
superDiagElement =
     3
     8
```

(e) `NewMatrix=diag(diag(A,2))` 返回：

```
NewMatrix =
     3     0
     0     8
```

注意，该矩阵的大小由命令 `diag(A, 2)` 生成的向量决定。(f) `SuperDiagonalMatrix =diag(diag(A,2))` 返回下列矩阵：

SuperDiagonalMatrix =

```
0    0    3    0
0    0    0    8
0    0    0    0
0    0    0    0
```

矩阵A的上第2对角线的长度为2，因此建立的矩阵大小为4×4。

在MATLAB中使用命令triu和tril来建立三角矩阵。

命令集36 从已存在的矩阵中生成新的矩阵(二)

triu(A)	生成一个和A大小相同的上三角矩阵。该矩阵的主对角线及以上元素取自A中相应元素，其余元素都为零。
triu(A,k)	生成一个和A大小相同的上三角矩阵。该矩阵的第k条对角线及以上元素取自A中相应元素，其余元素都为零。命令triu(A,0)等同于命令triu(A)。
tril(A)	生成一个和A大小相同的下三角矩阵。该矩阵的主对角线及以下元素取自A中相应元素，其余元素都为零。
tril(A,k)	生成一个和A大小相同的下三角矩阵。该矩阵的第k条对角线及以下元素取自A中相应元素，负数k表示主对角线下的对角线。其余元素都为零。命令tril(A,0)等同于命令tril(A)。

对于每一个方阵A都有下列关系：

```
A=triu(A) + tril(A) - diag(diag(A))
```

严格的上三角矩阵A应该使用triu(A,1)来定义；而严格的下三角矩阵A则用tril(A, -1)来定义。因此，对于每一个方阵A都有下列关系：

```
A=triu(A, 1) + tril(A,-1) + diag(diag(A))
```

当通过迭代的方法来求解线性方程系统（例如Gauss-Seidel, Jacobi 或者Successive Over Relaxation(SOR)）时，以这种方式来分解矩阵是很重要的。

例4.4

假设：

$$B = \begin{pmatrix} 9 & 8 & 7 & 6 \\ 1 & 3 & 0 & 7 \\ -4 & 7 & 1 & 9 \end{pmatrix}$$

(a) UpperTriangular =triu(B) 返回：

UpperTriangular =

```
9    8    7    6
0    3    0    7
0    0    1    9
```

(b) LowerTriangular =tril(-BL) 返回：

LowerTriangular =

```
0    0    0    0
1    0    0    0
-4   7    0    0
```

还有一些命令可以用来变换矩阵结构。

命令集37 矩阵旋转和矩阵变维

<code>fliplr(A)</code>	通过二维矩阵A的行元素按照 $b_{ij}=a_{i,n-j+1}$ 交换位置生成一个新矩阵。这里的‘lr’是‘left-right’的缩写。
<code>flipud(A)</code>	通过二维矩阵A的列元素按照 $b_{ij}=a_{m-i+1,j}$ 交换位置生成一个新矩阵。这里的‘ud’是‘up-down’的缩写。
<code>flipdim(A, dim)</code>	生成一个在dim维矩阵A内的元素交换位置的多维矩阵。命令 <code>flipdim(A, 1)</code> 等同于命令 <code>flipud(A)</code> ，命令 <code>flipdim(A, 2)</code> 等同于命令 <code>fliplr(A)</code> 。
<code>rot90(A)</code>	生成一个由矩阵A逆时针旋转90°而得的新阵。也就是将矩阵A中的左上角的元素和右下角的元素交换位置，也可见3.5节。
<code>rot90(A, k)</code>	生成一个由矩阵A逆时针旋转 $k \times 90^\circ$ 而得到的新阵，也可见13.5节。
<code>reshape(A,m,n,...p)</code>	生成一个 $m \times n \times \dots \times p$ 维的矩阵，它的元素以线性索引的顺序(见图2-2)从矩阵A中取来。如果矩阵A中没有 $m \times n \times \dots \times p$ 个元素，将返回一个错误信息。
<code>repmat(A,[m n ...p])</code>	创建一个和矩阵A有相同元素的 $m \times n \times \dots \times p$ 块的多维矩阵。
<code>repmat(x,[m n ...p])</code>	创建一个 $m \times n \times \dots \times p$ 的多维矩阵，所有元素的值都为标量x。使用该命令要比用命令 <code>x*ones([m n ...])</code> 来创建同一个大矩阵的速度要快。
<code>shiftdim(A,n)</code>	矩阵的列移动n步。n为正数，矩阵向左移；n为负数，向右移。
<code>squeeze(A)</code>	返回没有空维的矩阵A。
<code>cat(dim,A,B)</code>	将矩阵A和B组合成一个dim维的多维矩阵。
<code>permute(A,order)</code>	根据向量order来改变矩阵A中的维数顺序。
<code>ipermute(A,order)</code>	进行命令permute的逆变换。命令 <code>ipermute(permute(A, order), order)</code> 得到的结果就是矩阵A本身。

例4.5

(a) 假设有如例4.1中的多维矩阵 **OneMatrix**，使用命令 `B=reshape(OneMatrix,8)` 可以使它变维而成为二维矩阵，结果如下：

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(b) 为了在矩阵B中增加一层零元素，可以先使用命令 `C=zeros(3,8)` 创建一个零阵。然后通过下列命令来得到一个合并的矩阵：

```
D = cat (3, B, C)
```

```
D(:, :, 1) =
```

```
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
```

```
D(:, :, 2) =
```

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

(c) 为了快速对矩阵D进行变维以便它可以响应从命令cat(3,C,B)返回的结果, 可以使用:

```
flipdim(D,3)
```

```
ans(:, :, 1) =
```

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

```
ans(:, :, 2) =
```

```
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
```

(d) 使用命令permute和shiftdim对矩阵变维的结果如下:

```
size(shiftdim(D,2))
```

```
ans =
```

```
2 3 8
```

```
size(permute(D,[2 1 3]))
```

```
ans =
```

```
8 3 2
```

在MATLAB中可以通过增加元素、行和列将一个矩阵或者向量进行扩展。由于 MATLAB 可以自动地改变矩阵的大小, 所以使用已存在的矩阵的一部分来创建一个新矩阵是很容易的, 这在许多应用中都很有用。

从已存在的矩阵中建立一个矩阵就和定义一个新矩阵一样。元素用空格或逗号分隔, 行用分号或回车分隔; 见2.2节。在4.3节中给出了其相反过程, 从大矩阵中定义子矩阵。

例4.6

假设下列矩阵已经定义为:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \quad \mathbf{x} = (9 \quad 10) \quad \mathbf{y} = \begin{pmatrix} 11 \\ 12 \end{pmatrix} \quad \mathbf{z} = (13 \quad 14)$$

(a) 有几种方式可以将向量x扩展成 1×4 。假设想要的新向量是:

```
xnew=(9 10 0 5)
```

下列的三种方法都可以给出想要的结果：

- (i) `xnew = x; xnew(3) = 0; xnew(4) = 15;`
- (ii) `xnew = [x 0 15];`
- (iii) `temp = [0 15]; xnew = [x temp];`

(b) 以下两种方法可以对矩阵A扩展一个新行，如向量z：

- (i) `Anew1 = [A; z];`
- (ii) `Anew1 = [A; [13 14]];`

它们在屏幕上显示的结果如下：

```
Anew1 =
     1     2
     3     4
    13    14
```

有时还可以对矩阵添加多个新行：

```
Anew2 = [A; x; z; [0 0]]
Anew2 =
     1     2
     3     4
     9    10
    13    14
     0     0
```

对矩阵A扩展一个新列，如y，可以这样做：

```
Anew3=[A y] 或者 Anew3=[A [11; 12]]
```

它们在屏幕上显示的结果如下：

```
Anew3 =
     1     2    11
     3     4    12
```

扩展一个矩阵的操作是相似的。输入命令：

```
ABvert=[A; B] 和 ABhoriz=[A B]
```

就可以得到：

```
ABvert =
     1     2
     3     4
     5     6
     7     8

ABhoriz =
     1     2     5     6
     3     4     7     8
```

对于ABvert来说，它的列数一定等于矩阵A和B的列数；而对于ABhoriz来说，它的行数一定等于矩阵A和B的行数。

为了生成规则的矩阵块可以下列的方式使用命令 `repmat`。

例4.7

(a) 命令 `repmat([1 0; 0 1],3)` 将返回：

```
ans =
    1     0     1     0     1     0
    0     1     0     1     0     1
    1     0     1     0     1     0
    0     1     0     1     0     1
    1     0     1     0     1     0
    0     1     0     1     0     1
```

(b) 在例3.2中使用命令 `repmat([1 0],1)`，得到：

```
ans =
    1     0     1     0     1     0     1     0     1     0
```

(c) 如果要创建一个所有元素都是同一个值的矩阵，可以使用命令 `repmat(42,[22])` 来创建，给出的结果如下：

```
ans =
    42     42
    42     42
```

4.2 空矩阵

在MATLAB中对空矩阵的定义是 `A=[]`。有时创建一个多维的矩阵，但是这个矩阵中可能有几维是空的，比如 $0 \times 1 \times 0$ 矩阵。也可参见命令集31中的命令 `isempty`。

例4.8

要创建一个大小为 $1 \times 2 \times 0 \times 0 \times 2$ 的矩阵，可以使用命令 `zeros(1,2,0,0,2)`，结果为：

```
ans =
    Empty array: 1-by-2-by-0-by-0-by-2
```

空的行向量和列向量用命令 `zeros` 来定义：

```
rowvect = zeros(1,0)
```

```
rowvect =
    Empty matrix: 1-by-0
```

```
colvect = zeros(0,1)
```

```
colvect =
    Empty matrix: 0-by-1
```

现在这些向量的大小为：

```
size(rowvect), size(colvect)
```

```
ans =
     1     0
ans =
     0     1
```

一个空矩阵可以这样来创建：

```
A = []
```

```
A =  
[]
```

通过命令 `whos` 来查看在内存中的驻留变量的详细信息：

Name	Size	Bytes	Class
A	0x0	0	double array
colvect	0x1	0	double array
rowvect	1x0	0	double array

Grand total is 0 elements using 0 bytes

一些函数对空矩阵操作返回一个常量，在编写程序时这常常是有用的。在命令集 38 中 `E` 是一个空矩阵，为了清除矩阵中的空维可以使用命令 `squeeze`。

命令集38 空矩阵函数

```
squeeze(A)    返回没有空维的矩阵A。  
sum(E)        返回0。  
prod(E)       返回1。  
max(E)        返回E。  
min(E)        返回E。
```

4.3 向量和子矩阵的生成

在 MATLAB 中可以使用冒号 ‘:’ 来代表一系列数值。有时也使用它来定义一个子矩阵。我们先给出用冒号来定义向量的方法。

命令集39 数字序列(一)

```
i:k          创建从i开始、步长为1、到k结束的数字序列，即i, i+1, i+2, ..., k。如果  
              i>k，MATLAB则返回一个空矩阵，也就是 []。数字i和k不必是整数，该  
              序列的最后一个数是小于或等于k。  
  
i:j:k        创建从i开始、步长为j、到k结束的数字序列，即i, i+j, i+2j, ..., k。对于  
              j=0，则返回一个空矩阵。数字i、j和k不必是整数，该序列的最后一个数  
              是小于或等于k。
```

例 4.9

(a) 如果输入 `vect=2:7` 或者 `vect=2:7.7`，MATLAB 返回相同结果：

```
vect =  
     2     3     4     5     6     7
```

(b) 负步长: vect2=6:-1:1, 结果为:

```
vect2 =
     6     5     4     3     2     1
```

(c) 实数: realVect=1.2:-0.8:-3.2, 结果为:

```
realVect =
     1.2000     0.4000    -0.4000    -1.2000    -2.0000    -2.8000
```

注意最后一个数值为 - 2.8。

(d) 命令 realVect2=0:pi/4:pi, 结果为:

```
realVect2 =
     0     0.7854     1.5708     2.3562     3.1416
```

(e) 冒号可以用来定义矩阵:

```
Mat1=[2:4 0.1:1:2.1; 1:6]
```

结果为:

```
Mat1 =
     2.0000     3.0000     4.0000     0.1000     1.1000     2.1000
     1.0000     2.0000     3.0000     4.0000     5.0000     6.0000
```

(f) 冒号能够生成函数表, 比如 sine:

```
a = 0.0; b = 2*pi; n = 11; x = (a:(b-a)/(n-1):b)';
y = sin(x); Ftable = [x y]
```

结果为:

```
Ftable =
     0     0
     0.6283     0.5878
     1.2566     0.9511
     1.8850     0.9511
     2.5133     0.5878
     3.1416     0.0000
     3.7699    -0.5878
     4.3982    -0.9511
     5.0265    -0.9511
     5.6549    -0.5878
     6.2832    -0.0000
```

还有一些预定义函数也可以用来创建线性序列和逻辑序列。在绘图函数中这些序列都是有用的。

命令集40 数字序列(二)

linspace(a,b)	在区间[a, b]上创建一个有100个元素的向量, 这100个数把整个区间线性分隔。
linspace(a,b,n)	在区间[a, b]上创建一个有n个元素的向量。这个命令和冒号表示形式相近, 但是它直接定义了数据的个数。

`logspace(a,b)` 在区间 $[10^a, 10^b]$ 上创建一个有50个元素的向量，这50个数把整个区间对数分隔。特例：如果 $b=\pi$ ，则函数返回一个在区间 $[10^a, \pi]$ 上服从对数分布的向量。

`logspace(a,b,n)` 在区间 $[10^a, 10^b]$ 上创建一个有 n 个元素的向量，这 n 个数把整个区间对数分隔，特例情况同上。

如果从矩阵C中抽取行和/或列组成矩阵D，那么D就称为C的子阵，C中的行和列也可以称为C的子阵。所以一个矩阵可以有許多子阵。这可以推广到多维数组中去。在命令集41中列出了对二维数组操作的命令。

要取出其中一维的最后一个元素值，可以用值`end`来取。例如，A是一个 $4 \times 3 \times 2$ 的数组，`A(end,2,1)`就可以得到元素 a_{421} 的值，`A(end,end,end)`得到元素 a_{432} 的值。

当用冒号来定义矩阵A的子阵时，要使用在命令集41中列出的表达式。

命令集41 定义子阵

<code>A(i,j,...,k)</code>	返回多维数组A中下标为 (i, j, \dots, k) 的元素值，也可参见2.3节。
<code>A(:,j)</code>	返回二维矩阵A中第j列列向量。
<code>A(i,:)</code>	返回二维矩阵A中第i行行向量。
<code>A(:,j:k)</code>	返回由二维矩阵A中的第j列，第j+1列，直到第k列列向量组成的子阵。
<code>A(i:k,:)</code>	返回由二维矩阵A中的第i行，第i+1行，直到第k行行向量组成的子阵。
<code>A(i:k,j:l)</code>	返回由二维矩阵A中的第i行到第k行行向量和第j列到第l列列向量组成的子阵。
<code>A(:, :, ..., :)</code>	返回矩阵A本身。
<code>A(:)</code>	将矩阵A中的每列合并成一个长的列向量。
<code>A(j:k)</code>	返回一个行向量，其中的元素为A(:)中的从第j个元素到第k个元素。
<code>A([j1 j2...])</code>	返回一个行向量，其中的元素为A(:)中的第j1、j2...元素。
<code>A(:, [j1 j2...])</code>	返回矩阵A的第j1列、第j2列等的列向量。
<code>A([i1 i2...], :)</code>	返回矩阵A的第i1行、第i2行等的行向量。
<code>A([i1 i2...], [j1 j2...])</code>	返回矩阵A的第i1行、第i2行等和第j1列、第j2列等的元素。

也可参见`help colon`

例4.10

假设矩阵Ftable如例4.8(f)中一样定义。

(a) 语句`Submatrix=Ftable(2:4,:)`输出的结果为：

```
Submatrix =
    0.6283    0.5878
    1.2566    0.9511
    1.8850    0.9511
```

也就是它的每一列是从矩阵Ftable的第2到第4行。

(b) 为了使得从一个 $i \times j \times k$ 的立方体原点到 $3 \times 3 \times 3$ 的立方体的中心最远，可以使用的命令是：

```
A(end-2: end , end2: end , end2: end )
```

(c) 冒号表达式能够与关系运算符一起使用，见 3.8 节。通过下面简短的命令可以挑选出矩阵 **Ftable** 第 2 列中大于 0 的元素所在的行向量：

```
Selected=Ftable(Ftable(:, 2) > 0 ,: )
```

所得的结果为：

```
Selected =
    0.6283    0.5878
    1.2566    0.9511
    1.8850    0.9511
    2.5133    0.5878
    3.1416    0.0000
```

利用冒号表达式可以写出复合表达式，用 `help lis` 可得更多的信息。

4.4 MATLAB 中的特殊矩阵

在 4.1 节提到的零矩阵、单位矩阵和 1 矩阵都属于特殊矩阵，在同一节中还列出了对随机矩阵的操作命令。

另外，MATLAB 中还有一些命令用于生成试验矩阵。希尔伯特 (Hilbert) 矩阵，也称 H 阵，其元素为 $h_{ij}=1/(i+j-1)$ 。由于它是一个条件数差的矩阵，所以将它用来作为试验矩阵；见 7.6 节。

命令集 42 希尔伯特矩阵

```
hilb(n)      生成一个  $n \times n$  的希尔伯特矩阵。
invhilb(n)   生成一个  $n \times n$  的希尔伯特矩阵的逆矩阵，其元素都为整数。
```

例 4.11

如果输入 `H=hilb(3)`，`Hinv=invhilb(3)`，MATLAB 就会相应地输出：

```
H =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000

Hinv =
     9    -36     30
   -36    192   -180
     30   -180    180
```

可以看出希尔伯特矩阵和它的逆矩阵都是对称矩阵；见附录 B。

托普利兹 (Toeplitz) 矩阵由两个向量来定义，一个行向量和一个列向量。对称的托普利兹矩阵由单一向量来定义。

命令集 43 托普利兹矩阵

```
toeplitz(k,r) 生成一个非对称的托普利兹矩阵，将  $k$  作为第 1 列，将  $r$  作为第 1 行。
```

其余的元素与左上角相邻元素相等。

`toeplitz(c)` 用向量`c`生成一个对称的托普利兹矩阵。

例4.12

已知 $x=[1 \ 2 \ 3 \ 4]$ $y=[9 \ 8 \ 7 \ 6]$ 那么

`Toeppmatrix1 = toeplitz(x,y)`, `Toeppmatrix2 = toeplitz(y,x)`

给出结果为：

Column wins diagonal conflict.

`Toeppmatrix1 =`

```
1      8      7      6
2      1      8      7
3      2      1      8
4      3      2      1
```

Column wins diagonal conflict.

`Toeppmatrix2 =`

```
9      2      3      4
8      9      2      3
7      8      9      2
6      7      8      9
```

可以通过命令`gallery`来调用特殊矩阵库。输入`help gallery`可以找到哪些矩阵族是可用的，输入`help private/fami`可以得到特殊矩阵族的有关信息。旧版本中保留的有关特殊命令集应该避免使用。

在下面的命令集44中给出了MATLAB中其他特殊矩阵。

命令集44 其他特殊矩阵

<code>compan(p)</code>	生成一个 p 多项式的友矩阵，也就是它的特征多项式是 p 。 p 是一个包含多项式系数的向量，见 10.1节。
<code>gallery(n)</code>	生成一个在数字分析中有名的 $n \times n$ 试验矩阵。比如，只有 $n=3$ 和 $n=5$ 时该矩阵才存在： $n=3$ 时是一个条件数差的矩阵； $n=5$ 时是一个有意义的特征值问题矩阵。
<code>gallery family</code>	从 <code>family</code> 族中返回一个矩阵；见表 4-1。
<code>hadamard(k)</code>	返回一个阶数为 $n=2^k$ 的Hadamard矩阵。只有当 n 能被4整除时Hadamard矩阵才存在。
<code>hankel(x)</code>	返回一个由向量 x 定义的Hankel方阵。该矩阵是一个对称矩阵，它的元素为 $h_{ij}=x_{i+j-1}$ 。第1列为向量 x ，反三角以下的元素为0。
<code>hankel(x,y)</code>	返回一个 $m \times n$ 的Hankel矩阵，它的第 i 列为向量 x ，最后一行为向量 y 。
<code>magic(n)</code>	给出一个 $n \times n$ 的魔方矩阵。
<code>pascal(n)</code>	返回一个 $n \times n$ 的Pascal矩阵，它是对称、正定的矩阵，它的元素由Pascal三角组成。它的逆矩阵的所有元素是整数。

<code>pascal(n, k)</code>	$k=1$ 时给出一个由下三角的 Cholesky 系数组成的 Pascal 矩阵。要注意的是通常情况下在 MATLAB 中使用上三角的 Cholesky 系数。 $k=2$ 时是 <code>pascal(n,1)</code> 的变换和重新排列的形式。
<code>rosser</code>	给出 Rosser 矩阵, 这是一个经典对称特征测试问题, 它的大小是 8×8 。
<code>vander(x)</code>	返回一个倒数第 2 列为向量 x 的 Vandermonde 矩阵。它的元素 $v_{i,j} = x_i^{n-j}$, n 为向量 x 的长度值。
<code>wilkinson(n)</code>	返回一个 $m \times n$ 的 Wilkinson 特征值测试矩阵。

例4.13

$m \times n$ 的魔方矩阵由 $1 \sim n^2$ 的整数作为其元素, 并且矩阵任一行和列的元素之和相等。要创建一个 3×3 的魔方矩阵, 可以输入 `magic(3)`, 得到的结果为:

```
ans =
     8     1     6
     3     5     7
     4     9     2
```

例4.14

为了能够得到一个 Householder 矩阵, 可以运行命令 `help private/house` 这将给出指定参数的信息。下面的语句可用来创建一个 Householder 矩阵 H 。

```
x = [2; 5; 3];
[V, BETA] = gallery('house', x);
H = eye(3,3) - BETA*V*V'

H =
   -0.3244   -0.8111   -0.4867
   -0.8111    0.5033   -0.2980
   -0.4867   -0.2980    0.8212
```

表4-1中通过命令 `gallery` 列出了最常用的矩阵族。

表4-1 通过命令 `gallery` 得到的可用矩阵族

<code>circul</code>	给出循环序列矩阵, 第 1 列作为参数给出, 然后用相同的循环值建立矩阵的其他列
<code>dorr</code>	给出一个条件数差的对角三角矩阵
<code>house</code>	给出一个 Householder 矩阵
<code>invhess</code>	给出一个上 Hessenberg 矩阵的逆矩阵
<code>jordblock</code>	给出一个 Jordan 矩阵
<code>poisson</code>	给出一个稀疏对角三角块矩阵, 在求解带有有限差分的 Poisson 方程时要用到该矩阵
<code>vander</code>	给出一个 Vandermonde 矩阵
<code>wilk</code>	给出一个 Wilkinson 矩阵