第3章 MATLAB数值计算

MATLAB在矩阵分析和运算方面提供了强大的函数和命令功能。由于MATLAB中的所有数据都是通过矩阵形式存在的，因此，MATLAB的数值计算主要分为两类：一类是针对整个矩阵的数值计算，即矩阵运算；另一类则是针对矩阵中的元素进行的，可以称为矩阵元素的计算。

3.1 矩阵运算

矩阵是MATLAB数据组织和运算的基本单元，矩阵的加、减、乘、除、幂运算等代数运算是MATLAB数值计算最基础的部分。

3.1.1 矩阵的算术运算

矩阵算术运算的书写格式与普通算术运算相同，包括优先顺序规则，但其乘法和除法的定义和方法与标量截然不同。

表3-1为MATLAB矩阵的算术运算符及说明。

表3-1 MATLAB矩阵的算术运算符及说明

运算符	名称	实例	说　明
+	加	A+B	如果A、B为同维数矩阵，则表示A与B对应元素相加；如果其中一个矩阵为标量，则表示另一矩阵的所有元素加上该标量
−	减	A−B	如果A、B为同维数矩阵，则表示A与B对应元素相减；如果其中一个矩阵为标量，则表示另一矩阵的所有元素减去该标量
*	矩阵乘	A*B	矩阵A与B相乘，A和B均可为向量或标量，但A和B的维数必须符合矩阵乘法的定义
\	矩阵左除	A\B	方程A*X=B的解X
/	矩阵右除	A/B	方程X*A=B的解X
^	矩阵乘方	A^B	当A、B均为标量时，表示A的B次方幂；当A为方阵，B为正整数时，表示矩阵A的B次乘积；当A、B均为矩阵时，无定义

【例 3-1】 矩阵的算术运算。

```
>> A = [1 4 7;2 5 8;3 6 9];
>> B = [0 2 75;12 3 8;10 5 3];
>> C = A + B                    % 矩阵的加法运算
C =
      1          6         82
     14          8         16
     13         11         12
>> D = A - B                    % 矩阵的减法运算
D =
      1          2        -68
    -10          2          0
     -7          1          6
>> E = A\B                      % 矩阵的左除运算
Warning: Matrix is singular to working precision.
E =
     NaN        NaN        NaN
    -Inf        Inf        Inf
     Inf       -Inf       -Inf
>> F = A/B                      % 矩阵的右除运算
F =
    0.1518   -1.0654    1.3785
    0.1702   -1.2198    1.6638
    0.1886   -1.3743    1.9491
>> H = A * B                    % 矩阵的乘方运算
H =
     118         49        128
     140         59        214
     162         69        300
>> J = A ^ 3                    % 矩阵的乘方运算
J =
     468        1062       1656
     576        1305       2034
     684        1548       2412
```

注意：

(1) 如果 A、B 两矩阵进行加、减运算，则 A、B 必须维数相同，否则系统提示出错。

(2) 如果 A、B 两矩阵进行乘运算，则前一矩阵的列数必须等于后一矩阵的行数(内维数相等)。

(3) 如果 A、B 两矩阵进行右除运算，则两矩阵的列数必须相等(实际上，X＝B/A＝B×A^{-1})。

(4) 如果 A、B 两矩阵进行左除运算，则两矩阵的行数必须相等(实际上，X＝A\B＝A^{-1} · B)。

3.1.2 矩阵的转置

矩阵的转置计算在 MATLAB 中非常简单，就是运算符“'”，此运算符的运算级别比

加、减、乘、除等运算要高。

【例 3-2】 矩阵的转置运算。

```
>> clear all;
>> A = [1 1 1 1;2 2 2 2;3 3 3 3;4 4 4 4]
A =
     1     1     1     1
     2     2     2     2
     3     3     3     3
     4     4     4     4
>> A'
ans =
      1     2     3     4
      1     2     3     4
      1     2     3     4
      1     2     3     4
```

如果一个矩阵与其转置矩阵相等，则称其为对称矩阵；如果一个矩阵与其转置矩阵的和为零矩阵，则称其为反对称矩阵。在 MATLAB 中，判断对称和反对称的方法非常简单，如果 isequal(A,A')返回 1，则矩阵 A 为对称矩阵；如果 isequal(A,-A')返回 1，则矩阵 A 为反对称矩阵。例如：

```
>> B = [1 2 3;2 2 2;3 2 4]
B =
     1     2     3
     2     2     2
     3     2     4
>> isequal(B,B')
ans =
     1
```

3.1.3 方阵的行列式

对于方阵的行列式，手工计算是非常烦琐的，尤其是对于高阶方阵。而在 MATLAB 中，利用 det(A)函数就可以非常简单地计算矩阵行列式的值。由线性代数的知识可以知道，如果方阵 A 的元素为整数，则计算结果也为整数。

【例 3-3】 矩阵行列式的值。

```
>> A = [1 5 9;3 4 7;6 8 2]
A =
     1     5     9
     3     4     7
     6     8     2
>> det(A)
ans =
      132
```

注意：利用 det(X)＝0 来检验方阵是否为奇异矩阵，并不是任何时候都有效，在某些情况下就会判断错误。利用 abs(det(X))<＝tol 来判断矩阵的奇异性也并不可靠，因为误差 tol 是很难确定的，通常利用 cond(X)来判断奇异或接近奇异的矩阵比较合理。

3.1.4 矩阵的逆与伪逆

对于方阵 A，如果为非奇异方阵，则存在逆矩阵 inv(A)，使得 A＊inv(A)＝I。手工计算矩阵的逆是非常烦琐的，而利用函数 inv(A)则会非常方便地求出方阵的逆。函数 inv(A)在求方程的逆时采用高斯消去法。如果 A 不是方阵或 A 为奇异或接近奇异的方阵，函数会给出警告信息，计算结果将都为 Inf。在不是采用 IEEE 算法的机器上，上述情况会出错。

【例 3-4】 矩阵的逆。

```
>> A = [1 4 8 7;3 5 7 9;11 5 7 8;0 2 7 12];
>> B = inv(A)                    % 求矩阵的逆
B =
    -0.0246    -0.1088     0.1228     0.0140
    -0.1719     0.5719    -0.1404    -0.2351
     0.3860    -0.3860     0.0702     0.0175
    -0.1965     0.1298    -0.0175     0.1123
>> A * B
ans =
     1.0000     0.0000    -0.0000     0.0000
    -0.0000     1.0000    -0.0000          0
    -0.0000          0     1.0000     0.0000
    -0.0000          0    -0.0000     1.0000
>> C = [0 0 0;0 2 0;0 0 0]
C =
     0     0     0
     0     2     0
     0     0     0
>> inv(C)
```

警告：矩阵为奇异工作精度。

```
ans =
     Inf     Inf     Inf
     Inf     Inf     Inf
     Inf     Inf     Inf
```

对于奇异方阵或非方阵的矩阵，并不存在逆矩阵，但可以用函数 pinv(A)求其伪逆。其调用格式为：

```
B = pinv(A)
B = pinv(A,tol)
```

函数返回一个与矩阵 A 具有相同大小的矩阵 B，并且满足 ABA＝A，BAB＝B，且 AB

和 BA 都是 Hermintain 矩阵。计算方法是基于奇异值分解 svd(A),并且任何小于公差的奇异值都被看作零,其默认的公差为 max(size(A)) * norm(A) * eps。如果 A 为非奇异方阵,函数的计算结果与 inv(A)相同,但却会耗费大量的计算时间,相比较而言,inv(A)花费更少的时间。在其他情况下,pinv(A)具有 inv(A)的部分特性,但却不是与 inv(A)完全等同。例如:

```
>> pinv(A)
ans =
    -0.0246    -0.1088     0.1228     0.0140
    -0.1719     0.5719    -0.1404    -0.2351
     0.3860    -0.3860     0.0702     0.0175
    -0.1965     0.1298    -0.0175     0.1123
```

pinv(A)也可以用来求线性方程 Ax=b,x 可以由 pinv(A) * b 和 A/b 分别解得,其中 A 可以是方阵,也可以是奇异方阵。

【例 3-5】 利用矩阵的逆运算求解线性方程组。

```
>> pinv(A)
ans =
    -0.0246    -0.1088     0.1228     0.0140
    -0.1719     0.5719    -0.1404    -0.2351
     0.3860    -0.3860     0.0702     0.0175
    -0.1965     0.1298    -0.0175     0.1123
>> rank(A)
ans = >> b=[0.3;0.4;0.5;0.6]
b =
    0.3000
    0.4000
    0.5000
    0.6000
>> pinv(A) * b
ans =
    0.0189
   -0.0340
    0.0070
    0.0516
>> A\b
ans =
    0.0189
   -0.0340
    0.0070
    0.0516
```

3.1.5 矩阵或向量的范数

矩阵或向量范数是度量矩阵或向量在某种意义下的长度。范数有多种定义,其定义不同,范数值也就不同。在线性代数方程组的数值解法中,经常需要分析解向量的误差,需要比较误差向量的“大小”或“长度”,那么怎样定义向量的长度呢? 在初等数学里知

道，定义向量的长度，实际上就是对每一个向量按一定的法则规定一个非负实数与之对应，这一思想推广到 n 维线性空间里，就是向量的范数或模。

1）向量的 3 种常用范数及其计算函数

设向量 $V=(v_1,v_2,\cdots,v_n)$，下面讨论向量的 3 种范数。

（1）1 范数。

$$\|V\|_1=\sum_{i=1}^{n}|v_i|$$

（2）2 范数。

$$\|V\|_2=\sqrt{\sum_{i=1}^{n}v_i^2}$$

（3）∞范数。

$$\|V\|_\infty=\max_{1\leqslant i\leqslant n}\{|v_i|\}$$

在 MATLAB 中，求这 3 种向量范数的函数分别为：

- norm(V, 1)：计算向量 V 的 1 范数。
- norm(V)或 norm(V,2)：计算向量 V 的 2 范数。
- norm(V, inf)：计算向量 V 的∞(无穷)范数。

【例 3-6】 计算向量的 3 种范数。

```
>> clear all;
X = randperm(5)                          %产生向量
y = X.^2;
N2 = sqrt(sum(y));                       %利用范数定义求解向量范数
Ninf = max(abs(X));
Nneg_inf = min(abs(X));
%利用范数函数求解向量范数
n2 = norm(X);
ninf = norm(X,inf);
nneg_inf = norm(X, - inf);
%输出计算结果
disp('根据定义计算的范数结果:');
fprintf('2 范数:%8.6f\n',N2)
fprintf('无穷范数:%8.6f\n',Ninf)
fprintf('负无穷范数:%8.6f\n',Nneg_inf)
disp('根据 norm 函数计算的范数结果:')
fprintf('2 范数:%8.6f\n',n2)
fprintf('无穷范数:%8.6f\n',Ninf)
fprintf('负无穷范数:%8.6f\n',Nneg_inf)
```

运行程序，输出如下：

```
X =
     1     2     4     3     5
根据定义计算的范数结果:
2 范数:7.416198
无穷范数:5.000000
```

```
负无穷范数:1.000000
根据 norm 函数计算的范数结果:
2 范数:7.416198
无穷范数:5.000000
负无穷范数:1.000000
```

2) 矩阵的范数

设 A 为 n 阶方阵，R^n 中已定义了向量范数 $\|\cdot\|$，则称 $\sup_{\|X\|=1}\|AX\|$ 为矩阵 A 的范数或模，记为 $\|A\|$。

$$\|A\| = \sup_{\|X\|=1}\|AX\|$$

矩阵 A 的任一特征值的绝对值不超过 A 的范数 $\|A\|$。

矩阵 A 的特征值的最大绝对值称为 A 的谱半径，记为：

$$\rho(A) = \max_{-1\leqslant i\leqslant 1}|\lambda_i|$$

MATLAB 提供了 4 种求矩阵范数的函数。

- norm(A)或 norm(A,2)：计算矩阵 A 的 2 范数。
- norm(A,1)：计算矩阵 A 的 1 范数。
- norm(A,inf)：计算矩阵 A 的∞(无穷)范数。
- norm(A,'fro')：计算矩阵 A 的 Frobenius 范数。

【例 3-7】 求解 Hilbert 矩阵的范数。

```
>> clear all;
X = hilb(5) % 产生 Hilbert 矩阵,其中 H(i,j) = 1/(i + j - 1)
% 利用范数定义求解向量范数
N1 = max(sum(abs(X)));
N2 = norm(X);
Ninf = max(sum(abs(X')));
Nfro = sqrt(sum(diag(X' * X)));
% 利用范数函数求解向量范数
n1 = norm(X,1);
n2 = norm(X,2);
ninf = norm(X,inf);
nfro = norm(X,'fro');
% 输出计算结果
disp('根据定义计算的范数结果');
fprintf('1 范数:%8.6f\n',N1)
fprintf('2 范数:%8.6f\n',N2)
fprintf('无穷范数:%8.6f\n',Ninf)
fprintf('Frobenius 范数:%8.6f\n',Nfro)
disp('根据 norm 函数计算的范数结果')
fprintf('2 范数:%8.6f\n',n1)
fprintf('2 范数:%8.6f\n',n2)
fprintf('无穷范数:%8.6f\n',ninf)
fprintf('负无穷范数:%8.6f\n',nfro)
```

运行程序，输出如下：

```
X =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

根据定义计算的范数结果为：

```
1 范数:2.283333
2 范数:1.567051
无穷范数:2.283333
Frobenius 范数:1.580906
根据 norm 函数计算的范数结果
2 范数:2.283333
2 范数:1.567051
无穷范数:2.283333
负无穷范数:1.580906
```

3.1.6 矩阵的条件数

在线性代数中，描述线性方程 $Ax=b$ 的解对 b 的误差或不确定性的敏感度度量就是矩阵 A 的条件数，其对应的数学定义为：

$$k = \| A^{-1} \| \cdot \| A \|$$

根据数学基础知识可知，矩阵的条件数总是大于或等于 1。其中，正交矩阵的条件数为 1，奇异矩阵的条件数为∞，而病态矩阵的条件数则比较大。

依据条件数，方程解的相对误差可以由以下不等式来估算：

$$\frac{1}{k}\left(\frac{|\delta b|}{|x|}\right) \leqslant \frac{|\delta x|}{|x|} \leqslant k\left(\frac{|\delta b|}{|b|}\right)$$

在 MATLAB 中，提供了 cond(A)求取矩阵 A 的条件数。函数的调用格式为：

c=cond(X)：计算矩阵 X 的 2 范数下的条件数。

c=cond(X,p)：计算矩阵的 p 范数下的条件数，p 的取值有：

- p=1 时，即为计算矩阵 X 的 1 范数下的条件数。
- p=2 时，即为计算矩阵 X 的 2 范数下的条件数。
- p=inf 时，即为计算矩阵 X 的∞范数下的条件数。
- p=fro 时，即为计算矩阵 X 的 Frobenius 范数下的条件数。

【例 3-8】 计算矩阵的条件数。

```
>> A=[1 5 9 4;3 6 7 9;12 0 8 4];
>> C1=norm(A)                    % 矩阵 2 范数下的条件数
C1 =
     20.4664
>> C2=norm(A,1)                  % 矩阵 1 范数下的条件数
C2 =
     24
```

```
>> C3 = norm(A,inf)                    % 矩阵∞范数下的条件数
C3 =
     25
>> C4 = norm(A,'fro')                  % 矩阵 Frobenius 范数下的条件数
C4 =
     22.8473
```

在 MATLAB 中，采用 rcond 函数来计算矩阵条件数的倒数值。函数的调用格式为：

c＝rcond(A)：计算矩阵 A 条件数的倒数。当矩阵 A 为病态时，该函数的返回值接近 0；当矩阵为良态时，返回值接近 1。

【例 3-9】 计算矩阵条件数的倒数值。

```
>> A = magic(3)
A =
     8          1          6
     3          5          7
     4          9          2
>> r1 = rcond(A)        % 魔方矩阵的条件数的倒数值
r1 =
     0.1875
>> B = hilb(3)          % 3 阶病态矩阵
B =
     1.0000     0.5000     0.3333
     0.5000     0.3333     0.2500
     0.3333     0.2500     0.2000
>> r2 = rcond(B)        % 病态矩阵的条件数的倒数值
r2 =
     0.0013
```

此外，在 MATLAB 中，采用 condest 函数计算矩阵 1 范数下的条件数的估计值，该函数的调用格式为：

c＝condest(A)：计算矩阵 A 的 1 范数下的条件数的估计值。

【例 3-10】 计算矩阵 1 范数下的条件数的估计值。

```
>> M = rand(1000);
t1 = clock;
M_cond = cond(M,1)
t2 = clock;
t_norm = etime(t2,t1)
t3 = clock;
M_condest = condest(M)
t4 = clock;
t_condest = etime(t4,t3)
```

运行程序，输出如下：

```
M_cond =
     1.2773e+05
```

```
t_norm =
     0.3590
M_condest =
     1.2773e+05
t_condest =
     0.1880
```

3.1.7 矩阵的秩

矩阵的秩用函数 rank 来求取，该函数返回矩阵的行向量或列向量的不相关个数。

函数 rank 返回矩阵 A 的奇异值中比误差 tol 大的奇异值的个数，tol 的默认值为 max(size(A)) * norm(A) * eps。rank(A,tol)将指定误差 tol。

在 MATLAB 中，计算矩阵的秩的算法是以奇异值分解为基础的，用奇异值分解的方法来求解矩阵的秩非常耗时，但却是最有效的方法。

【例 3-11】 求矩阵的秩。

```
>> A=[3 8;7 9;2 1;3 4];
>> rank(A)
ans =
     2
>> B=[1 3 7;2 5 8];
>> rank(B)
ans =
     2
>> C=rank(magic(3))
C =
     3
```

3.1.8 矩阵的迹

矩阵的迹等于矩阵的对角线元素之一，也等于矩阵的特征值之和。在 MATLAB 中，提供了 trace 函数求矩阵的迹。函数的调用格式为：

b=trace(A)：求矩阵 A 的迹，返回值为 b。

【例 3-12】 求矩阵的迹。

```
>> A=magic(3)
    A =
     8     1     6
     3     5     7
     4     9     2
>> trace(A)                    %矩阵的迹
ans =
     15
```

3.1.9 矩阵的正交基

在 MATLAB 中，提供了 orth 函数求矩阵的标准正交基。函数的调用格式为：

B＝orth(A)：矩阵 B 的列向量组成了矩阵 A 的标准正交基，于是 B' * B＝eye(rank(A))。

【例 3-13】 求矩阵的标准正交基。

```
>> A = [1 2 3;3 8 9;7 4 5];
>> O1 = orth(A)              % 矩阵的标准正交基
O1 =
    -0.2374    -0.1357    -0.9619
    -0.7880    -0.5521     0.2724
    -0.5680     0.8227     0.0241
>> O1' * O1                  % 检验
ans =
     1.0000         0    -0.0000
          0    1.0000     0.0000
    -0.0000    0.0000     1.0000
>> O2 = orth(B)              % 矩阵的标准正交基
O2 =
    -0.5774     0.7071     0.4082
    -0.5774     0.0000    -0.8165
    -0.5774    -0.7071     0.4082
>> O2' * O2                  % 检验
ans =
     1.0000     0.0000    -0.0000
     0.0000     1.0000    -0.0000
    -0.0000    -0.0000     1.0000
```

3.1.10 矩阵化零

对于非满秩的矩阵 A，存在某矩阵 Z，满足 $A \cdot Z=0$，同时矩阵 Z 是一个正交矩阵，也就是说 $Z' \cdot Z=I$，则矩阵 Z 被称为矩阵 A 的化零矩阵。在 MATLAB 中，提供了 null 函数实现矩阵的化零矩阵求解。函数的调用格式为：

Z＝null(A)：返回矩阵 A 的化零矩阵，如果化零矩阵不存在，则返回空矩阵。

Z＝null(A,'r')：返回有理形式的化零矩阵。

【例 3-14】 求非满秩矩阵 A 的化零矩阵。

```
>> clear all;
A = [1  2  3;1  2  3;1  2  3];
Z = null(A)
Z =
         0    0.9636
   -0.8321   -0.1482
    0.5547   -0.2224
```

```
>> A * Z
ans =
  1.0e-015 *
    0.2220              0.2220
    0.2220              0.2220
    0.2220              0.2220
>> Z' * Z
ans =
     1.0000             0.0000
     0.0000             1.0000
>> ZR = null(A,'r')                    % 求解有理数形式化零矩阵
ZR =
    -2                  -3
     1                   0
     0                   1
>> RZ = A * ZR
RZ =
     0                   0
     0                   0
     0                   0
```

3.1.11 矩阵的特征向量

特征值和特征向量的求解和运算问题是线性代数中一个重要的课题，它们在工程和科学实践中应用非常广泛。

对 $n\times n$ 方阵 A，其特征值 λ(标量)和特征值对应的特征向量 x(向量)满足关系式：

$$Ax = \lambda x$$

如果把矩阵 A 的 n 个特征值放在矩阵的对角线上，得到 D，即 $D=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n)$时，如果把特征值对应的向量按照和特征值对应的次序排列，可以得到矩阵 V 的数据列。如果矩阵 V 是可逆的，那么，特征值问题可以转化为 $A=VD$。进一步可表示为：

$$A = VDV^{-1}$$

关于特征值和特征向量，MATLAB 中提供了多个命令来进行分析。

在 MATLAB 中，提供了 eig 函数用于求矩阵特征值和特征向量。函数的调用格式为：

d=eig(A)：求矩阵 A 的全部特征值，构成向量 d。

d=eig(A,B)：求矩阵 A 与矩阵 B 的特征值，且 A、B 为方阵。

[V,D]=eig(A)：求矩阵 A 的全部特征值，构成对角阵 D，并求 A 的特征向量构成 V 的列向量。

[V,D]=eig(A,'nobalance')：直接求矩阵 A 的特征值与特征向量。

[V,D]=eig(A,B)：求方阵 A、B 的特征值，构成对角阵 D，并求特征向量构成 V 的列向量，并且 A * V=B * V * D。

[V,D]=eig(A,B,flag)：指定算法 flag 计算特征值和特征向量，flag 取值如下：

- chol：利用 Cholesky 分解法求解矩阵的特征值与特征向量。

• qz：利用 QZ 分解法求解矩阵的特征值与特征向量。

【例 3-15】 求矩阵的特征值与特征向量，并求出相似矩阵 T 与平衡矩阵 B。

```
>> clear all;
A = [ 3 -2 -.9 2*eps; -2 4 1 -eps;
     -eps/4    eps/2    -1    0; -.5    -.5    .1    1];
>> [V,E] = eig(A)
V =
       0.6153       -0.4176       -0.0000       -0.1437
      -0.7881       -0.3261       -0.0000        0.1264
      -0.0000       -0.0000       -0.0000       -0.9196
       0.0189        0.8481        1.0000        0.3432
E =
    5.5616        0        0         0
         0   1.4384        0         0
         0        0   1.0000         0
         0        0        0   -1.0000
>> A = [ 3 -2 -.9 2*eps; -2 4 1 -eps;
     -eps/4    eps/2    -1    0; -.5    -.5    .1    1];
>> [T,B] = balance(A)
T =
     1.0e+07 *
    0.0000        0        0        0
         0   0.0000        0        0
         0        0   0.0000        0
         0        0        0   3.3554
B =
      3.0000    -2.0000    -0.0000     0.0000
     -2.0000     4.0000     0.0000    -0.0000
     -0.0000     0.0000    -1.0000          0
     -0.0000    -0.0000     0.0000     1.0000
>> [V,E] = eig(B)
V =
      0.6154    -0.7882    -0.0000    -0.0000
     -0.7882    -0.6154    -0.0000     0.0000
     -0.0000    -0.0000    -0.0000    -1.0000
      0.0000     0.0000     1.0000     0.0000
E =
    5.5616        0        0         0
         0   1.4384        0         0
         0        0   1.0000         0
         0        0        0   -1.0000
```

在一些工程及物理问题中，通常只需要求出矩阵 A 的按模最大的特征值（称为 A 的主特征值）和相应的特征向量，这些求部分特征值的问题可以利用 eigs 函数来实现。函数的调用格式为：

d=eigs(A)：求矩阵 A 的 6 个最大特征值，并以向量 d 形式存放。

[V,D]=eigs(A)：求矩阵 A 的 6 个最大特征值，并返回部分对角矩阵 D 和部分特征向量 V。

[V,D,flag]=eigs(A)：flag 为返回的收敛标志。如果 flag=0，即表示融合所有的

特征值；否则，即不能融合所有的特征值。

[V,D,flag]=eigs(A,k)：返回矩阵 A 的 k 个最大特征值。

[V,D,flag]=eigs(A,k,sigma)：根据 sigma 的取值来求 A 的部分特征值，其中 sigma 的取值及说明如表 3-2 所示。

表 3-2　sigma 取值及说明

sigma 取值	说　　明
'lm'	求按模最大的 k 个特征值
'sm'	求按模最小的 k 个特征值
'la'	对实对称问题求 k 个最大特征值
'sa'	对实对称问题求 k 个最小特征值
'be'	同时返回实对称问题 k 个最大及最小特征值
'lr'	对非实对称和复数问题求 k 个最大实部特征值
'sr'	对非实对称和复数问题求 k 个最小实部特征值
'li'	对非实对称和复数问题求 k 个最大虚部特征值
'si'	对非实对称和复数问题求 k 个最小虚部特征值

[V,D,flag]=eigs(A,K,sigma,opts)：opts 为指定的结构体属性，默认值为{}。

[V,D,flag]=eigs(Afun,n,…)：根据 sigma 的取值来求由 M 文件 Afun.m 生成的矩阵 A 的 n 个最大特征值。

【例 3-16】 求矩阵的按模最大与最小特征值。

```
>> A=[1 3 -2 0;-5 7 12 -9;-2 0 3 6;1 1 0 1];
>> dmax=eigs(A,1)                    %求按模最大特征值
dmax =
    4.0329 - 4.9181i
>> [V,D]=eigs(A,1)
V =
    -0.3596 - 0.3056i
     0.1488 - 0.8406i
     0.0169 + 0.0867i
    -0.1880 - 0.0731i
D =
    4.0329 + 4.9181i
>> dmin=eigs(A,1,'sm')               %求按模最小特征值
dmin =
     -1.7818
>> [V1,D1]=eigs(A,1,'sm')            %求按模最小特征值
V1 =
     -0.7679
      0.3818
     -0.4953
      0.1388
D1 =
     -1.7818
```

3.1.12 矩阵的指数和对数

矩阵的指数运算用 expm 函数来实现，expm(X)＝V＊diag(exp(diag(D)))/V(其中X为已知矩阵，[V,D]＝eig(X))。对数运算用 logm 函数来实现，L＝logm(A)，与矩阵的指数运算互为逆运算。

【例 3-17】 矩阵的指数和对数运算。

```
>> clear all;
>> A = rand(4)
A =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
>> B = expm(A)
B =
    4.7204    2.4561    4.1836    3.6737
    2.9289    2.4812    3.2288    2.5767
    1.4100    1.0458    2.5691    1.8036
    3.0394    1.9091    3.3480    3.3748
>> C = logm(A)
```

警告：没有为包含非正实数特征值的 A 定义主矩阵对数，返回非主矩阵对数。

```
> In logm (line 78)
C =
   -1.4731 + 1.9566i    0.6562 - 0.7525i    2.0518 - 1.3177i    1.3357 - 1.1300i
    0.2052 - 0.8626i   -0.5924 + 2.5938i    0.3846 - 0.9592i    0.9259 - 0.8225i
    1.7795 - 0.4950i    0.0202 - 0.3144i   -1.6024 + 2.5911i   -0.9987 - 0.4720i
    0.2522 - 0.9002i    0.2665 - 0.5716i    0.4827 - 1.0009i    0.0236 + 2.2832i
```

3.1.13 Jordan 标准型

即使一个矩阵无法相似于对角矩阵，它依旧可以相似于一个分块对角阵。如果某个 t 重特征值 λ_k 计算得出的线性无关特征向量数量小于 t，那么：

$$(A-\lambda_k)^j\eta = 0$$

其中，$j=1,2,\cdots,k-1$。由此计算的 η 成为第 j 阶广义特征向量。增加 j 的值，直到找出全部的 t 个线性无关广义特征向量。每个高阶特征向量都能在那些比它低一阶的特征向量中找到一个对应的特征向量，满足 $(A-\lambda)\eta_j=\eta_{j-1}$。然后再把所有特征向量组合成矩阵 P 时，需要将高阶广义特征向量与它对应的低一阶的特征向量放在一起，满足

$$J = P^{-1}AP$$

计算得到矩阵 A 的 Jordan 标准型 J。

J 是一个分块对角矩阵，而且对角矩阵块满足如下情形：

$$\begin{bmatrix} \lambda_k & 1 & 0 & 0 & 0 \\ 0 & \lambda_k & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \lambda_k & 1 \\ 0 & 0 & 0 & 0 & \lambda_k \end{bmatrix}$$

容易发现，Jordan 标准型有个不错的性质：计算 A 的 n 次方，当 n 趋于无穷时，只要将特征值 λ_k 的绝对值小于 1，其对应的矩阵块将趋于 0。

在 MATLAB 中，提供了 jordan 函数来求 Jordan 标准型。函数的调用格式为：

J=jordan(A)：J 为 A 的 Jordan 标准型。

[V,J]=jordan(A)：J 为 A 的 Jordan 标准型，同时返回标准型的相似变换矩阵 V。

【例 3-18】 求矩阵的 Jordan 标准型。

```
>> A = [1 -3 -2; -1 1 -1; 2 4 5]
[V, J] = jordan(A)
A =
     1    -3    -2
    -1     1    -1
     2     4     5
V =
    -1     1    -1
    -1     0     0
     2     0     1
J =
     2     1     0
     0     2     0
     0     0     3
>> V\A*V
ans =
     2     1     0
     0     2     0
     0     0     3
```

3.2 矩阵的数理分析

前面对矩阵的一些运算及操作进行了介绍，本节对矩阵的数理分析进行介绍。

3.2.1 最大值与最小值

在 MATLAB 中，提供了 max 与 min 函数用于求数据序列的最大值与最小值。两个函数的调用格式与操作过程类似，可以分别用来求向量或矩阵的最大值与最小值。max 函数的调用格式为：

C=max(A)：如果 A 为向量，则返回 A 的最大值存入 C，若 A 中包含复数元素，则按模取最大值。如果 A 为矩阵，即返回一个行向量，向量的第 i 个元素是矩阵 A 的第 i 列上的最大值。

C=max(A,B)：返回与 A、B 大小相同的最大元素，A、B 的大小必须相同或是 A、B

都为方阵。

C=max(A,[],dim)：dim 取 1 或 2。当 dim=1 时，该函数等价于 max(A)；当 dim=2 时，返回一个列向量，其第 i 个元素是 A 矩阵的第 i 行上的最大值。

[C,I]=max(…)：返回行向量 C 与 I，C 向量记录 A 的每列的最大值，I 向量记录 A 的每列的最大值行号。

min 函数与 max 函数的调用格式类似。

【例 3-19】 求向量与矩阵的最大(小)值。

```
>> clear all;
A = [1 8 9 0 5 7 6 12 8 -7 15];
y = max(A)                    % 求向量 A 的最大值
y =
     15
>> y2 = min(A)                % 求向量 A 的最小值
y2 =
     -7
>> B = [12 3 -0 49;5 39 75 4;5 3 8 121];
[C,I] = max(B)                % 求矩阵 B 的最大值
C =
     12    39    75    121
I =
      1     2     2      3
>> [C2,I2] = min(B)           % 求矩阵 B 的最小值
C2 =
     5    3    0    4
I2 =
     2    1    1    2
```

3.2.2 元素的查找

在 MATLAB 中，提供了 find 函数用于矩阵元素的查找，它通常与关系函数和逻辑运算相结合。函数的调用格式为：

ind=find(X)：查找矩阵 X 中的非零矩阵，函数返回这些元素的单下标。

[row,col]=find(X, …)：查找矩阵 X 中的非零元素，函数返回这些元素的双下标 row 和 col。

[row,col,v]=find(X, …)：查找矩阵 X 中的非零元素，函数返回这些元素的双下标 row 和 col，同时返回下标的索引 v。

【例 3-20】 实现矩阵元素的查找。

```
>> clear all;
A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
>> [r,c,v] = find(A > 10)
r =
     1
     2
     4
     4
     1
     3
c =
     1
     2
     2
     3
     4
     4
v =
     1
     1
     1
     1
     1
     1
```

3.2.3 元素的排序

在 MATLAB 中，矩阵元素的排序使用函数 sort，该函数默认按照升序排列，返回值为排序后的矩阵，和原矩阵的维数相同。函数的调用格式为：

B＝sort(A)：对矩阵 A 按照升序进行排列。当 A 为向量时，返回由小到大排序后的向量；当 A 为矩阵时，返回 A 中各列按照由小到大排序后的矩阵。

B＝sort(A,dim)：返回在给定的维数 dim 上按照由小到大顺序排序后的结果。当 dim＝1 时，按照列进行排序；当 dim＝2 时，按照行进行排列。

B＝sort(…,mode)：可以指定排序的方式。参数 mode 默认值为'ascend'，即按照升序进行排列；当 mode 为'descend'时，对矩阵进行降序排列。

[B,IX]＝sort(A,…)：输出参数 B 为排序后的结果，输出参数 IX 中元素表示 B 中对应元素在输入参数 A 中的位置。

【例 3-21】 矩阵元素的排序。

```
>> clear all;
>> A = [9 0 -7 5 3 8 -10 4 2];
B = sort(A)                    %向量排序
B =
    -10    -7     0     2     3     4     5     8     9
>> C=[3 6 5; 7 -2 4; 1 0 -9]
C =
     3     6     5
     7    -2     4
     1     0    -9
```

```
>> D1 = sort(C)                    % 矩阵中元素按照列进行升序排序
D1 =
      1    -2    -9
      3     0     4
      7     6     5
>> D2 = sort(C,2)                  % 矩阵中元素按照行进行升序排序
D2 =
      3     5     6
     -2     4     7
     -9     0     1
>> D3 = sort(C,'descend')          % 矩阵中元素按照列进行降序排序
D3 =
      7     6     5
      3     0     4
      1    -2    -9
>> D4 = sort(C,2,'descend')        % 矩阵中元素按照行进行降序排序
D4 =
      6     5     3
      7     4    -2
      1     0    -9
```

3.2.4 求和与求积运算

在 MATLAB 中，提供了 sum 函数用于求数据序列的和，prod 函数用于求数据序列的积。sum 函数的调用格式为：

B＝sum(A)：如果 A 为向量，返回各元素的和；如果 A 为矩阵，返回一个行向量，其第 i 个元素是矩阵 A 的第 i 列的各元素之和。

B＝sum(A,dim)：当 dim＝1 时，等价于 sum(A)；当 dim＝2 时，返回一个列向量，其第 i 个元素是矩阵 A 的第 i 行的各元素之和。

B＝sum(…, 'double')或 B＝sum(…, dim,'double')：不论 A 为单精度类型或是整型数据，都返回一个双精度类型。

prod 函数的调用格式与 sum 函数类似。

【例 3-22】 矩阵的求和与求积运算。

```
>> clear all;
>> A = [1 3 2; 4 2 5; 6 1 4]
A =
    1     3     2
    4     2     5
    6     1     4
>> sum(A)                    % 矩阵中元素按照列进行求和
ans =
    11     6    11
>> sum(A,2)                  % 矩阵中元素按照行进行求和
ans =
     6
    11
```

```
    11
>> prod(A)                          % 矩阵中元素按照列进行求积
ans =
     24      6      40
>> prod(A,2)                        % 矩阵中元素按照行进行求积
ans =
      6
     40
     24
```

提示：通过 sum(sum())可求出矩阵所有元素的和。

3.2.5 求累和与求累积运算

在 MATLAB 中，提供了 cumsum 函数用于求向量或矩阵的累和运算，cumprod 函数用于求向量或矩阵的累积运算。函数的调用格式为：

B＝cumsum(A)：如果 A 为向量，返回向量 A 的累加和；如果 A 为矩阵，返回一个矩阵，其第 i 列是矩阵 A 的第 i 列的累加和。

B＝cumsum(A,dim)：当 dim＝1 时，等价于 cumsum(A)；当 dim＝2 时，返回一个矩阵，其第 i 行是矩阵 A 的第 i 行的累加和。

B＝cumprod(A)：当 A 为向量时，返回向量 A 的累乘积；当 A 为矩阵时，返回一个矩阵，其第 i 列是矩阵 A 的第 i 列的累乘积。

B＝cumprod(A,dim)：当 dim＝1 时，等价于 comprod(A)；当 dim＝2 时，返回一个向量，其第 i 行是矩阵 A 的第 i 行的累乘积。

【例 3-23】 求矩阵的累和与累积运算。

```
>> clear all;
>> A = [1 4 7; 2 5 8; 3 6 9]
A =
    1      4      7
    2      5      8
    3      6      9
>> cumsum(A)                        % 矩阵各列元素的和
ans =
    1       4       7
    3       9      15
    6      15      24
>> cumsum(A,2)                      % 矩阵各行元素的和
ans =
    1      5      12
    2      7      15
    3      9      18
>> cumprod(A)                       % 矩阵各列元素的积
ans =
    1       4       7
    2      20      56
    6     120     504
```

```
>> cumprod(A,2)                          % 矩阵各行元素的积
ans =
     1     4    28
     2    10    80
     3    18   162
```

3.2.6 平均值与中值

在 MATLAB 中，提供了 mean 函数用于求数据序列的平均值，median 函数用于求数据序列的中值。函数的调用格式为：

M＝mean(A)：如果 A 为向量，返回 A 的算术平均值；如果 A 为矩阵，返回一个行向量，其第 i 个元素是矩阵 A 的第 i 列的算术平均值。

M＝mean(A,dim)：当 dim＝1 时，等价于 mean(A)；当 dim＝2 时，返回一个列向量，其第 i 个元素是矩阵 A 的第 i 行的算术平均值。

M＝median(A)：如果 A 为向量，返回 X 的中值；如果 A 为矩阵，返回一个行向量，其第 i 个元素是矩阵 A 的第 i 列的中值。

M＝median(A,dim)：当 dim＝1 时，等价于 median(A)；当 dim＝2 时，返回一个列向量，其第 i 个元素是矩阵 A 的第 i 行的中值。

【例 3-24】 求矩阵的平均值与中值运算。

```
>> clear all;
>> A = [0 1 1; 2 3 2; 1 3 2; 4 2 2]
A =
     0     1     1
     2     3     2
     1     3     2
     4     2     2
>> M1 = mean(A)
M1 =
    1.7500    2.2500    1.7500
>> M2 = mean(A,2)
M2 =
     0.6667
     2.3333
     2.0000
     2.6667
>> m1 = median(A)
m1 =
    1.5000    2.5000    2.0000
>> m2 = median(A,2)
m2 =
     1
     2
     2
     2
```

3.2.7 标准差

在MATLAB中，提供了std函数用于计算数据序列的标准差。函数的调用格式为：

s=std(X)：如果X为向量，返回向量X的一个标准差；如果X为矩阵，返回一个行向量，它的各个元素便是矩阵X各列或各行的标准差。

s=std(X,flag,dim)：dim可以取1或2。当dim=1时，求各列元素的标准差；当dim=2时，则求各行元素的标准差。flag可以取0或1，当flag=0时，置前因子为1/n−1；否则置前因子为1/n。默认flag=0且dim=1。

【例3-25】 求矩阵的标准差。

```
>> A = [4 -5 1; 2 3 5; -9 1 7];
S = std(A)
S =
     7.0000    4.1633    3.0551
>> S2 = std(A,0,1)
S2 =
     7.0000    4.1633    3.0551
>> S3 = std(A,0,2)
S3 =
     4.5826
     1.5275
     8.0829
>> S4 = std(A,1,1)
S4 =
     5.7155    3.3993    2.4944
>> S5 = std(A,1,2)
S5 =
     3.7417
     1.2472
     6.5997
>> w = [1 1 0.5];              %置前因子
S6 = std(A,w)
S6 =
     4.8826    3.6661    2.4000
```

3.2.8 相关系数

在MATLAB中，提供了corrcoef函数用于求数据的相关系数。函数的调用格式为：

R=corrcoef(X)：返回从矩阵X形成的一个相关系数矩阵，此相关系数矩阵的大小与矩阵X一样。它把矩阵X的每列作为一个变量，然后求它们的相关系数。

R=corrcoef(X,Y)：返回两个随机变量X与Y之间的相关系数。

[R,P]=corrcoef(…)：返回一个矩阵P，用于测试矩阵的p值。

[R,P,RLO,RUP]=corrcoef(…)：返回矩阵RLO与RUP，其大小与R相同，分别为置信区间在95%相关系数的上限与下限。

[…]=corrcoef(…,'param1',val1,'param2',val2,…)：param1,param2,…与 val1,var2,…为指定的额外参数名与参数值。

【例 3-26】 计算矩阵的相关系数。

```
>> clear all;
x = randn(30,4);                          % 不相关数据
x(:,4) = sum(x,2);                        % 相关性引进
[r,p] = corrcoef(x)                       % 计算样本相关及 p 值
r =
      1.0000    -0.0994    -0.3324     0.3448
     -0.0994     1.0000    -0.0978     0.3810
     -0.3324    -0.0978     1.0000     0.4787
      0.3448     0.3810     0.4787     1.0000
p =
    1.0000    0.6012    0.0727    0.0620
    0.6012    1.0000    0.6070    0.0378
    0.0727    0.6070    1.0000    0.0075
    0.0620    0.0378    0.0075    1.0000
>> [i,j] = find(p<0.05);                  % 寻找相关性
[i,j]
ans =
     4     2
     4     3
     2     4
     3     4
```

3.2.9 元素的差分

差分运算是累积和的逆运算，MATLAB 中对应的函数为 diff，该函数是数组支持函数。函数的调用格式为：

Y=diff(X)：计算矩阵各列的差分。

Y=diff(X,n)：计算矩阵各列的 n 阶差分。

Y=diff(X,n,dim)：计算矩阵在 dim 方向上的 n 维差分。当 dim=1 时，计算矩阵各列元素的差分；当 dim=2 时，计算矩阵各行元素的差分。

【例 3-27】 矩阵的差分运算。

```
>> clear all;
>> A = magic(3)
A =
    8    1    6
    3    5    7
    4    9    2
>> B1 = diff(A)                    % 矩阵各列元素的差分
B1 =
    -5    4     1
     1    4    -5
>> B2 = diff(A,2)                  % 矩阵各行元素的差分
```

```
B2 =
    6     0     -6
>> B3 = diff(A,1,1)                    %矩阵各列元素的差分
B3 =
     -5     4      1
      1     4     -5
>> B4 = diff(A,1,2)                    %矩阵各行元素的差分
B4 =
     -7      5
      2      2
      5     -7
```

提示：当参数 n≥size(x,dim)时，函数的值是空矩阵。

3.3 高维数组

随着数组的维数增加，数组的运算和处理就会变得越来越困难，在 MATLAB 中提供了一些函数可以进行这些高维数组的处理和运算。常见的高维数组处理和运算函数如表 3-3 所示。

表 3-3 常见的高维数组处理和运算函数

函　数	说　明
squeeze	用此函数来消除数组中的“弧维”，即大小等于 1 的维，从而起到降维的作用
sub2ind	将下标转换为单一索引数值
ind2sub	将数组的单一索引数值转换为数组的下标
flipdim	沿着数组的某个维轮换顺序，第二个参数为变换的对称面
shiftdim	维序号循环轮换移动
permute	对多维数组进行广义共轭转置操作
ipermute	取消转置操作
size	获取数组的维数大小数值

【例 3-28】 高维数组的处理和操作。

```
>> clear all;
>> A = [ -1:2;3:6;7:10]
A =
     -1     0     1      2
      3     4     5      6
      7     8     9     10
>> B = reshape(A,[2 2 3])
B(:,:,1) =
     -1     7
      3     0
B(:,:,2) =
      4     1
      8     5
```

```
B(:,:,3) =
      9     6
      2    10
>> C = cat(4,B(:,:,1),B(:,:,2),B(:,:,3))
C(:,:,1,1) =
     -1     7
      3     0
C(:,:,1,2) =
      4     1
      8     5
C(:,:,1,3) =
      9     6
      2    10
>> D = squeeze(C)                    % 降维操作
D(:,:,1) =
     -1     7
      3     0
D(:,:,2) =
      4     1
      8     5
D(:,:,3) =
      9     6
      2    10
>> sub2ind(size(D),1,2,3)            % 索引转换
ans =
     11
>> [i,j,k] = ind2sub(size(D),11)
i =
     1
j =
     2
k =
     3
>> flipdim(D,1)                      % 按行进行翻转
ans(:,:,1) =
      3     0
     -1     7
ans(:,:,2) =
      8     5
      4     1
ans(:,:,3) =
      2    10
      9     6
>> flipdim(D,2)                      % 按列进行翻转
ans(:,:,1) =
      7    -1
      0     3
ans(:,:,2) =
      1     4
      5     8
ans(:,:,3) =
      6     9
     10     2
```

```
>> flipdim(D,3)                    % 按页进行翻转
ans(:,:,1) =
      9     6
      2    10
ans(:,:,2) =
      4     1
      8     5
ans(:,:,3) =
     -1     7
      3     0
>> shiftdim(D,1)                   % 移动一维
ans(:,:,1) =
     -1     4     9
      7     1     6
ans(:,:,2) =
      3     8     2
      0     5    10
>> F = ipermute(ans,[3,2,1])
F(:,:,1) =
     -1     4     9
      3     8     2
F(:,:,2) =
      7     1     6
      0     5    10
```

3.4 稀疏矩阵

在许多问题中提到了含有大量零元素的矩阵，这样的矩阵称为稀疏矩阵。为了节省存储空间和计算时间，MATLAB 考虑到矩阵的稀疏性，在对它进行运算时有特殊的命令。

一个稀疏矩阵中有许多元素等于零，这便于矩阵的计算和保存。如果 MATLAB 把一个矩阵当作稀疏矩阵，那么只需在 m×3 的矩阵中存储 m 个非零项。第 1 列是行下标，第 2 列是列下标，第 3 列是非零元素值，不必保存零元素。如果存储每个浮点数需要 8 字节，存储每个下标需要 4 字节，那么整个矩阵在内存中存储需要 16×m 字节。

3.4.1 稀疏矩阵与全矩阵

MATLAB 利用二维数组存储全矩阵时，对零元素、非零元素不作区分，统一采用浮点数。MATLAB 在存储稀疏矩阵时只存储非零元素及其对应的索引值(整型)，显然，这种方式能够大大提高稀疏矩阵的存储效率。

【例 3-29】 稀疏矩阵的存储效率。

```
>> tic
A = sprand(2000,3000,0.1);
toc
Elapsed time is 0.731645 seconds.
>> tic
```

```
fullA = full(A);
toc
Elapsed time is 0.039148 seconds.
>> whos
   Name         Size                Bytes    Class      Attributes
   A         2000x3000             6863092    double     sparse
   fullA     2000x3000            48000000    double
```

从上面的结果可看出，稀疏矩阵占用的空间比全矩阵小得多。

注意：代码中的 tic 表示计时开始，toc 表示计时结束，时间差约为中间代码执行时间。

采用稀疏矩阵的另外一个好处就是执行效率的提高。对 m×n 矩阵 A，非零元素所占比例为 α，计算 2×A 需要执行 m×n 次浮点数乘法；将矩阵 A 转换为稀疏矩阵 sparse_A，利用 sparse_A 计算 2×A 时，仅对非零元素进行运算，因此仅需执行 m×n×α 次浮点数乘法，运行效率提高了(1/α−1)倍。

3.4.2 稀疏矩阵的存储方式

对于稀疏矩阵，MATLAB 仅存储矩阵所有的非零元素的值及其位置(行号和列号)。显然，这对于具有大量零元素的稀疏矩阵来说是十分有效的。

设矩阵 $A=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 5 & 0 & 0\\ 2 & 0 & 0 & 6\end{bmatrix}$ 是具有稀疏矩阵特征的矩阵，其完全存储方式是按列存储的全部 12 个元素：1,0,2,0,5,0,0,0,0,0,0,6；其稀疏存储方式为：(1,1),1,(3,1),2,(2,2),5,(3,4),6，其中，括号内为元素的行列位置，后面为元素值。当矩阵非常“稀疏”时，会有效地节省存储空间。

3.4.3 稀疏矩阵的生成

MATLAB 提供了多种创建稀疏矩阵的方法：

- 利用 sparse 函数从满矩阵转换得到稀疏矩阵。
- 利用一些特定函数创建包括单位稀疏矩阵在内的特殊稀疏矩阵。

1. 利用 sparse 创建稀疏矩阵

在 MATLAB 中，提供了 sparse 函数创建一般的稀疏矩阵，full 函数将稀疏矩阵 S 转换为一个满矩阵。函数的调用格式为：

S=sparse(A)：将矩阵 A 转化为稀疏矩阵形式，即由 A 的非零元素和下标构成稀疏矩阵 S。如果 A 本身为稀疏矩阵，则返回 A 本身。

S=sparse(m,n)：生成一个 m×n 的所有元素都是 0 的稀疏矩阵 S。

S=sparse(i,j,s)：生成一个由长度相同的向量 i、j 和 s 定义的稀疏矩阵 S，其中 i、j

是整数向量，定义稀疏矩阵的元素位置(i,j)，s是一个标量或与i、j长度相同的向量，表示在(i,j)位置上的元素。

S=sparse(i,j,s,m,n)：生成一个m×n的稀疏矩阵S，(i,j)对应位置元素为s，m=max(i)且n=max(j)。

S=sparse(i,j,s,m,n,nzmax)：生成一个m×n的含有nzmax个非零元素的稀疏矩阵S，nzmax的值必须大于或者等于向量i和j的长度。

【例3-30】 利用sparse函数创建稀疏矩阵。

```
>> S = sparse(1:10,1:10,1:10)              % 行下标分别为1～10
S =
     (1,1)       1
     (2,2)       2
     (3,3)       3
     (4,4)       4
     (5,5)       5
     (6,6)       6
     (7,7)       7
     (8,8)       8
     (9,9)       9
     (10,10)    10
>> S = sparse(1:10,1:10,5)                 % 行下标都设置为5
S =
     (1,1)       5
     (2,2)       5
     (3,3)       5
     (4,4)       5
     (5,5)       5
     (6,6)       5
     (7,7)       5
     (8,8)       5
     (9,9)       5
     (10,10)     5
```

此外，sparse函数还可以将一个满矩阵转换成一个稀疏矩阵，相应的调用格式为：S=sparse(X)：X为满矩阵。

【例3-31】 将满矩阵 $A=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 5 & 0 & 0\\ 2 & 0 & 0 & 6\end{bmatrix}$ 转换为稀疏矩阵。

```
>> clear all;
>> A = [1 0 0 0;0 5 0 0;2 0 0 6];
>> S = sparse(A)
S =
    (1,1)       1
    (3,1)       2
    (2,2)       5
    (3,4)       6
```

反之，MATLAB提供了full函数把稀疏矩阵转换为满矩阵。full函数的调用格

式为：

A＝full(S)：把矩阵存储方式从任何一个存储形式转换为满矩阵形式。

【例 3-32】 利用 sparse 函数创建稀疏矩阵，并将稀疏矩阵转换为满矩阵。

```
>> clear
>> s = sparse([1 2 3 4 5],[2 1 4 6 2],[10 3 -2 -5 1],10,12)   % 利用 sparse 函数创建一个稀
                                                               % 疏矩阵
s =
    (2,1)       3
    (1,2)      10
    (5,2)       1
    (3,4)      -2
    (4,6)      -5
>> F = full(s)                                                 % 将稀疏矩阵转换为满矩阵
F =
     0    10     0     0     0     0     0     0     0     0     0     0
     3     0     0     0     0     0     0     0     0     0     0     0
     0     0     0    -2     0     0     0     0     0     0     0     0
     0     0     0     0     0    -5     0     0     0     0     0     0
     0     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     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
>> whos s F
  Name      Size            Bytes  Class     Attributes
  s        10x12              112  double    sparse
  F        10x12              960  double
```

2. 利用特定函数创建稀疏矩阵

在 MATLAB 中，除了 sparse 函数外，还提供了一些函数来创建特殊的稀疏矩阵，如表 3-4所示。

表 3-4 创建特殊稀疏矩阵的函数

函　数	说　明
S＝speye(m,n)	创建单位稀疏矩阵
S＝spones(X)	创建非零元素为 1 的稀疏矩阵
S＝sprand(X)	创建非零元素为均匀分布的随机数的稀疏矩阵
S＝sprandn(X)	创建非零元素为高斯分布的随机数的稀疏矩阵
s＝sprandsym(X)	创建非零元素为高斯分布的随机数的对称稀疏矩阵
s＝spdiags(X)	创建对角稀疏矩阵
s＝spalloc(X)	为稀疏矩阵分配空间

【例 3-33】 利用 speye 函数创建单位稀疏矩阵。

```
>> clear all;
>> A = speye(5)                    % 创建 5 阶单位稀疏矩阵
```

```
A =
   (1,1)    1
   (2,2)    1
   (3,3)    1
   (4,4)    1
   (5,5)    1
>> C1 = full(A)
C1 =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
>> B = speye(5,6)                    % 创建 5×6 稀疏矩阵
B =
   (1,1)    1
   (2,2)    1
   (3,3)    1
   (4,4)    1
   (5,5)    1
>> C2 = full(B)
C2 =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
```

注意：通过 speye 函数建立的单位稀疏矩阵，只有对角线元素为 1，其余位置元素均为 0。

【例 3-34】 创建非零元素为随机数的对称稀疏矩阵。

```
>> clear all;
>> A = sprandsym(5,0.1)              % 建立非零元素为随机数的对称稀疏矩阵
A =
     (4,1)    0.5377
     (5,1)    1.8339
     (1,4)    0.5377
     (1,5)    1.8339
>> C1 = full(A)
C1 =
         0         0         0    0.5377    1.8339
         0         0         0         0         0
         0         0         0         0         0
    0.5377         0         0         0         0
    1.8339         0         0         0         0
>> B = spones(A)                     % 建立非零元素为 1 的与矩阵 A 维数相同的对称稀疏矩阵
B =
     (4,1)    1
     (5,1)    1
     (1,4)    1
     (1,5)    1
```

3.4.4 稀疏矩阵的操作

由于稀疏矩阵的维度一般比较大，直接查看系数矩阵不利于用户查看系统矩阵的信息，为此 MATLAB 提供了查看稀疏矩阵定量信息和图形化信息的函数，主要用来查看稀疏矩阵的非零元素信息和图形化稀疏矩阵信息。

1）nnz 函数

该函数用于返回稀疏矩阵中所有非零元素存储单元的个数。函数的调用格式为：

n＝nnz(X)：返回矩阵 X 中分配给稀疏矩阵中所有非零元素存储单元的个数。

2）nonzeros 函数

该函数用于返回一个包含所有非零元素的列向量。函数的调用格式为：

s＝nonzeros(A)：返回矩阵 A 中非零元素按列顺序构成的列向量。

3）nzmax 函数

该函数用于返回稀疏矩阵中所有非零元素存储单元的个数。函数的调用格式为：

n＝nzmax(S)：返回矩阵 S 中分配给稀疏矩阵中所有非零元素存储单元的个数。

4）spy 函数

MATLAB 提供了查看稀疏矩阵的图形化函数 spy。函数的调用格式为：

- spy(S)：绘制稀疏矩阵 S 中非零元素的分布图形，S 可以为全元素矩阵。
- spy(S，markersize)：绘制稀疏矩阵 S 中非零元素的分布图形，markersize 为整数，指定点阵大小。
- spy(S，'LineSpec')：绘制稀疏矩阵 S 中非零元素的分布图形，LineSpec 指定绘图标记和颜色。
- spy(S，'LineSpec'，markersize)：绘制稀疏矩阵 S 中非零元素的分布图形，LineSpec 指定绘图标记和颜色，markersize 指定点阵的大小。

【例 3-35】 查看稀疏矩阵的信息实例。

```
>> A = [1 5 0 0 0;0 0 2 1 0;0 0 0 0 3;0 7 8 5 0];
>> S = sparse(A);
>> n1 = nnz(S)                          % 非零元素的个数
n1 =
     8
>> n2 = nonzeros(S)                     % 非零元素的值
n2 =
     1
     5
     7
     2
     8
     1
     5
     3
>> n3 = nzmax(S)                        % 非零元素的存储空间
n3 =
     8
>> spy(S)                               % 以图形形式显示稀疏矩阵，效果如图 3-1 所示
```

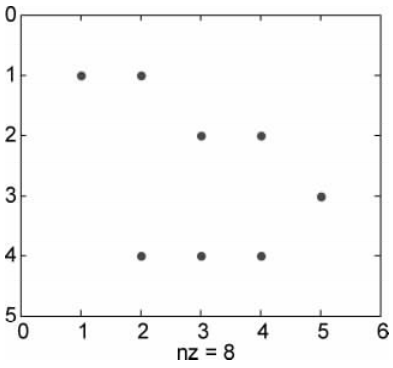

图 3-1　稀疏矩阵图形显示效果

3.4.5　稀疏矩阵的运算

满矩阵的四则运算对稀疏矩阵同样有效，但是返回结果有可能是稀疏矩阵或满矩阵。

- 对于单个稀疏矩阵的输入，大部分函数输出的结果都是稀疏矩阵，少部分函数输出的结果是满矩阵。
- 对于多个矩阵的输入，如果其中至少有一个矩阵是满矩阵，那么大部分函数的输出结果是满矩阵。
- 对于矩阵的加、减、乘、除运算，只要其中有一个是满矩阵，则输出的结果都是满矩阵。
- 稀疏矩阵的数乘为稀疏矩阵。
- 稀疏矩阵的幂为稀疏矩阵。

3.5　矩阵的分解

矩阵分解是一个非常重要的概念，如求矩阵的特征值和特征向量、矩阵的秩等重要参数时都要用到矩阵的分解。在实际工程中，在特定的场合要对矩阵进行特定形式的分解。

在 MATLAB 中，常见的矩阵分解有 Cholesky 分解、LU 分解、QR 分解、Schur 分解、Hessenberg 分解等。

3.5.1　Cholesky 分解

Cholesky 分解是专门针对对称正定矩阵的分解。设 $A=(a_{ij})\in R^{n\times n}$ 是对称正定矩阵，$A=R^{\mathrm{T}}R$ 称为矩阵 A 的 Cholesky 分解，其中 $R\in R^{n\times n}$ 是一个具有正值对角元素的上三角矩阵，即：

$$R=\begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ & r_{22} & r_{23} & r_{24} \\ & & r_{33} & r_{34} \\ & & & r_{44} \end{pmatrix}$$

这种分解是唯一存在的。

在 MATLAB 中，提供了 chol 函数用于实现 Cholesky 分解。函数的调用格式为：

R＝chol(A)：返回 Cholesky 分解因子 R。

L＝chol(A,'lower')：把矩阵 A 分解为一个下三角矩阵。

R＝chol(A,'upper')：把矩阵 A 分解为一个上三角矩阵。

[R,p]＝chol(A)：若 A 为正定矩阵，则 p＝0，R 为分解因子；若 A 为非正定矩阵，则 p 为正整数，R 为有序的上三角矩阵。

[L,p]＝chol(A,'lower')：若 A 为正定矩阵，则 p＝0，L 为下三角矩阵；若 A 为非正定矩阵，则 p 为正整数，L 为有序的下三角矩阵。

[R,p]＝chol(A,'upper')：若 A 为正定矩阵，则 p＝0，R 为上三角矩阵；若 A 为非正定矩阵，则 p 为正整数，L 为有序的上三角矩阵。

[R,p,S]＝chol(A)：若 A 为稀疏矩阵，S 为返回的转置矩阵。

[R,p,s]＝chol(A,'vector')：如果 A 为稀疏矩阵，则返回一个向量转换信息 s，使得 A(s,s)＝R' * R。

【例 3-36】 利用 chol 函数实现矩阵的 Cholesky 分解。

```
>> clear all;
>> n = 5;
X = pascal(n)
X =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
>> [R,p] = chol(X)
R =
     1     1     1     1     1
     0     1     2     3     4
     0     0     1     3     6
     0     0     0     1     4
     0     0     0     0     1
p =
     0
>> R' * R % 检验
ans =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
```

Cholesky 分解除了可应用于对正定矩阵进行分解外，还可对线性方程组进行求解。

【例 3-37】 使用 Cholesky 分解求解线性方程组 $\begin{cases} x_1+x_2+x_3+x_4=1 \\ x_1+2x_2+3x_3+4x_4=4 \\ x_1+3x_2+6x_3+10x_4=6 \\ x_1+4x_2+10x_3+20x_4=13 \end{cases}$。

其实现的 MATLAB 代码为：

```
>> clear all;
A = [1 1 1 1;1 2 3 4;1 3 6 10;1 4 10 20];
b = [1 4 6 13]';
x = A\b                          % 用左除法求解方程的解
x =
     -9
     23
    -19
      6
>> R = chol(A)                   % Cholesky 分解
R =
    1    1    1    1
    0    1    2    3
    0    0    1    3
    0    0    0    1
>> Rt = transpose(R);            % 进行转置
>> x1 = R\(Rt\b)                 % 利用 Cholesky 分解求解线性方程
x1 =
     -9
     23
    -19
      6
```

由 x 及 x1 的结果可看出，使用 Cholesky 分解求解得到的线性方程组的数值解和使用左除法求解线性方程组的数值解相一致。其数学原理为：

对于线性方程组 $Ax=b$，其中 A 为对称正定矩阵，有 $A=R^{\mathrm{T}}R$，则根据定义，线性方程组可以转换为 $R^{\mathrm{T}}Rx=b$，该方程组的数值为 $x=R\backslash(R^{\mathrm{T}}\backslash b)$。

3.5.2 LU 分解

高斯消去法分解又称 LU 分解，它可以将任意一个方阵 A 分解为一个下三角矩阵 L 和一个上三角矩阵 U 的乘积，即 $A=LU$。LU 分解在 MATLAB 中用函数 lu 来实现。函数的调用格式为：

[L,U]=lu(A)：对矩阵 A 进行 LU 分解，其中 L 为单位下三角矩阵或其变换形式，U 为上三角矩阵。

[L,U,P]=lu(A)：L 为单位下三角矩阵，U 为上三角矩阵，P 为置换矩阵，满足 LU=PA。

[L,U,P,Q]=lu(A)：L 为单位下三角矩阵，U 为上三角矩阵，P 为行重新排列矩

阵，Q 为列重新排列矩阵，满足 P＊A＊Q＝L＊U。

[L,U,P,Q,R]＝lu(A)：L 为单位下三角矩阵，U 为上三角矩阵，P、Q 为置换矩阵，R 为对角缩放矩阵，满足 P＊(R\A)＊Q＝L＊U。

Y＝lu(A)：A 为一个方阵，把上三角矩阵和下三角矩阵合并在矩阵 Y 中给出，矩阵 Y 的对角元素为上三角矩阵的对角元素，即 Y＝L＋U－I。置换矩阵 P 的信息丢失。

考虑线性方程组 $Ax=b$，其中，矩阵 A 可以被 LU 分解，使得 $A=LU$，这样线性方程组就可以改写成 $LUx=b$，由于左除运算符“\”可以快速处理三角矩阵，因此可以快速解出：

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

利用 LU 分解来计算行列式的值和矩阵的逆，其形式为：

- det(A)＝det(L)＊det(U)。
- inv(A)＝inv(U)＊inv(L)。

【例 3-38】 对矩阵进行 LU 分解。

```
>> clear all;
>> A = [1 4 9 11;2 5 8 13;3 6 9 17;0 1 8 10]
A =
     1     4     9    11
     2     5     8    13
     3     6     9    17
     0     1     8    10
>> [L1,U1] = lu(A)
L1 =
    0.3333    1.0000         0         0
    0.6667    0.5000   -0.2000    1.0000
    1.0000         0         0         0
         0    0.5000    1.0000         0
U1 =
    3.0000    6.0000    9.0000   17.0000
         0    2.0000    6.0000    5.3333
         0         0    5.0000    7.3333
         0         0         0    0.4667
>> [L2,U2,P] = lu(A)
L2 =
    1.0000         0         0         0
    0.3333    1.0000         0         0
         0    0.5000    1.0000         0
    0.6667    0.5000   -0.2000    1.0000
U2 =
    3.0000    6.0000    9.0000   17.0000
         0    2.0000    6.0000    5.3333
         0         0    5.0000    7.3333
         0         0         0    0.4667
P =
     0     0     1     0
     1     0     0     0
     0     0     0     1
     0     1     0     0
```

```
>> Y1 = lu(A)
Y1 =
    3.0000    6.0000    9.0000   17.0000
    0.3333    2.0000    6.0000    5.3333
         0    0.5000    5.0000    7.3333
    0.6667    0.5000   -0.2000    0.4667
>> L1 * U1                          % 验证
ans =
     1     4     9    11
     2     5     8    13
     3     6     9    17
     0     1     8    10
>> Y2 = L2 + U2 - eye(size(A))      % 验证
Y2 =
    3.0000    6.0000    9.0000   17.0000
    0.3333    2.0000    6.0000    5.3333
         0    0.5000    5.0000    7.3333
    0.6667    0.5000   -0.2000    0.4667
```

注意：通过 lu 函数对矩阵进行分解时，L1 为下三角形矩阵的置换矩阵，L2 为下三角形矩阵，并进行原矩阵 A=L1 * U1、Y2=L2+U2-eye(size(A))的验证。

此外，矩阵的 LU 分解常用于求解线性方程组 $Ax=b$。首先对系数矩阵 A 作 LU 分解，使 $PA=LU$，此时线性方程组 $Ax=b$ 转换为 $LUx=Pb$，求解过程分两步进行：

（1）求解线性方程组 $Ly=Pb$，得 $y=L\backslash(Pb)$。

（2）求原方程组的解 $Ux=y$，得 $x=U\backslash y$。

【例 3-39】 利用 LU 分解求线性方程组 $\begin{bmatrix}1&1&1&1\\1&2&3&4\\1&3&6&10\\1&4&10&20\end{bmatrix}x=\begin{bmatrix}1\\4\\6\\13\end{bmatrix}$ 的解。

```
>> clear all;
A = [1 1 1 1;1 2 3 4;1 3 6 10;1 4 10 20];
b = [1 4 6 13]';
[L,U,P] = lu(A)                    % LU 分解
L =
    1.0000         0         0         0
    1.0000    1.0000         0         0
    1.0000    0.6667    1.0000         0
    1.0000    0.3333    1.0000    1.0000
U =
    1.0000    1.0000    1.0000    1.0000
         0    3.0000    9.0000   19.0000
         0         0   -1.0000   -3.6667
         0         0         0    0.3333
P =
     1     0     0     0
     0     0     0     1
     0     0     1     0
     0     1     0     0
>> % 令 Ux = y,则 Ly = Py,所以 y = L\(Pb)
```

```
y = L\(P * b)
y =
      1
     12
     -3
      2
>> x = U\y                    % 原方程组 x = U\y 的解
x =
    -9.0000
    23.0000
   -19.0000
     6.0000
```

3.5.3 QR 分解

在数值分析中，为了求解矩阵的特征值，引入了正交(QR)分解方法。对于非奇异矩阵 $A(n\times n)$，则存在正交矩阵 Q 和上三角矩阵 R，使得 $A=Q*R$，QR 分解是唯一的。

在 MATLAB 中，提供了 qr 函数实现 QR 分解。函数的调用格式为：

[Q,R]=qr(A)：返回正交矩阵 Q 和上三角矩阵 R，Q 和 R 满足 A=QR。若 A 为 m×n 矩阵，则 Q 为 m×m 矩阵，R 为 m×n 矩阵。

[Q,R]=qr(A,0)：产生矩阵 A 的“经济型”分解，即若 A 为 m×n 矩阵，且 m>n，则返回 Q 的前 n 列，R 为 n×n 矩阵；否则该命令等价于[Q,R]=qr(A)。

[Q,R,E]=qr(A)：求得正交矩阵 Q 和上三角矩阵 R，E 为置换矩阵，使得 R 的对角线元素按绝对值大小降序排列，满足 AE=QR。

[Q,R,E]=qr(A,0)：产生矩阵 A 的“经济型”分解，E 为置换矩阵，使得 R 的对角线元素按绝对值大小降序排列，且 A(:,E)=Q*R。

R=qr(A)：对稀疏矩阵 A 进行分解，产生一个上三角矩阵 R，R 为 A'A 的 Cholesky 分解因子，即满足 R'R=A'A。

R=qr(A,0)：对稀疏矩阵 A 的“经济型”分解。

[C,R]=qr(A,B)：此命令用来计算方程组 Ax=b 的最小二乘解。

【例 3-40】 对矩阵进行 QR 分解。

```
>> clear all;
>> A = [-3 4 5;8 2 6;1 9 5];
>> [Q,R] = qr(A)
Q =
    -0.3487         -0.4556         -0.8190
     0.9300         -0.0598         -0.3627
     0.1162         -0.8882          0.4446
R =
    8.6023          1.5112          4.4174
         0         -9.9356         -7.0780
         0               0         -4.0482
>> [Q,R,E] = qr(A)
Q =
```

```
    -0.3980          0.4133         -0.8190
    -0.1990         -0.9104         -0.3627
    -0.8955          0.0186          0.4446
R =
   -10.0499         -1.2935         -7.6618
         0          -8.5045         -3.3028
         0                0         -4.0482
E =
     0     1     0
     1     0     0
     0     0     1
>> [Q,R] = qr(A,0)
Q =
    -0.3487         -0.4556         -0.8190
     0.9300         -0.0598         -0.3627
     0.1162         -0.8882          0.4446
R =
    8.6023          1.5112          4.4174
         0          -9.9356         -7.0780
         0                0         -4.0482
>> R = qr(A)
R =
     8.6023          1.5112          4.4174
    -0.6895          -9.9356         -7.0780
    -0.0862          0.6750          -4.0482
```

在程序中，对矩阵进行 QR 分解。矩阵的 QR 分解不一定是方阵，对于长方形矩阵也适用。对分解的结果进行验证，即 A＝QR。

【例 3-41】 利用 QR 分解求线性方程组 $\begin{bmatrix}1&1&1&1\\1&2&3&4\\1&3&6&10\\1&4&10&20\end{bmatrix}x=\begin{bmatrix}1\\4\\6\\13\end{bmatrix}$ 的解。

```
>> clear all;
A = [1 1 1 1;1 2 3 4;1 3 6 10;1 4 10 20];
b = [1 4 6 13]';
if issparse(A),
    R = qr(A);
else R = triu(qr(A));
end
x = R\(R'\(A' * b))              % 求解
r = b - A * x                    % 检验
err = R\(R'\(A' * r))            % 误差
```

运行程序，输出如下：

```
x =
   -9.0000
    23.0000
   -19.0000
```

```
    6.0000
r =
    1.0e-14 *
   -0.5329
         0
   -0.7105
         0
err =
     1.0e-12 *
   -0.0497
    0.1101
   -0.0924
    0.0266
```

在 MATLAB 中，还可以对 QR 分解得到的矩阵的行和列进行删除和添加操作，其中，qrdelete 函数为删除行或列，而 qrinsert 函数则为插值某些行或列。这两个函数格式的形式几乎相同，此处以 qrdelete 函数为例进行说明。qrdelete 函数的调用格式为：

[Q1，R1]＝qrdelete(Q，R，j)：返回矩阵 A1 的 QR 分解结果，其中 A1 是矩阵 A 删除第 j 列得到的结果，而矩阵 A＝QR。

[Q1，R1]＝qrdelete(Q，R，j，'col')：计算结果和[Q1，R1]＝qrdelete(Q，R，j)相同。

[Q1，R1]＝qrdelete(Q，R，j，'row')：返回矩阵 A1 的 QR 分解结果，其中 A1 是矩阵 A 删除第 j 行得到的结果，而矩阵 A＝QR。

【例 3-42】 QR 分解的删除与插入操作。

```
>> clear all;
A = magic(4);
[Q,R] = qr(A);
j = 3;
[Q1,R1] = qrdelete(Q,R,j,'row')                    %QR 分解的删除操作
Q1 =
    0.9284         -0.3592         -0.0950
    0.2901          0.5411          0.7893
    0.2321          0.7604         -0.6066
R1 =
    17.2337          8.2977          9.1681         14.6225
          0         15.8792         15.7392          0.4198
          0               0         -1.4909          4.4728
x = 1:4;
[Q1,R1] = qrinsert(Q,R,j,x,'row')                  %QR 分解的插入操作
Q1 =
    0.8219         -0.4207          0.2754         -0.1474          0.2236
    0.2568          0.5122         -0.4422          0.1617          0.6708
    0.0514          0.0900          0.4571          0.8833          0.0000
    0.4623          0.1278         -0.5177          0.2280         -0.6708
    0.2055          0.7323          0.5016         -0.3462         -0.2236
R1 =
    19.4679         10.6842         11.0438         18.6974
          0         16.1198         15.8814          1.2551
          0               0          2.1941         -3.8394
```

```
     0         0         0         5.3000
     0         0         0              0
```

3.5.4 Schur 分解

Schur 分解是 Schur 于 1909 年提出的矩阵分解，它是一种典型的酉相似变换，这种变换的最大好处是能够保持数值稳定，因此在工程计算中也是重要的工具之一。

Schur(舒尔)分解定义式为：

$$A = USU'$$

其中，A 必须是一个方阵，U 是一个酉矩阵，S 是一个块对角化矩阵，由对角线上的 1×1 和 2×2 块组成。特征值可以由矩阵 S 的对角块给出，而矩阵 U 给出比特征向量更多的数值特征。此外，对缺陷矩阵也可以进行 Schur 分解。

在 MATLAB 中，提供了 schur 函数用于实现 Schur 分解。函数的调用格式为：

T＝schur(A)：返回 Schur 矩阵 T，若 A 有复特征值，则相应的对角元以 2×2 的块矩阵形式给出。

T＝schur(A,flag)：若 A 有复特征值，则 flag＝complex，否则 flag＝real。

[U,T]＝schur(A,…)：返回酉矩阵 U 和 Schur 矩阵 T。

另外，函数 rsf2csf 可以把实数形式的舒尔矩阵转换成复数形式的舒尔矩阵。

【例 3-43】 对矩阵进行 Schur 分解。

```
>> clear all;
>> A = pascal(5)
  A =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
>> [U,S] = schur(A)
U =
    0.1680   -0.5706   -0.7660    0.2429    0.0175
   -0.5517    0.5587   -0.3830    0.4808    0.0749
    0.7025    0.2529    0.1642    0.6110    0.2055
   -0.4071   -0.5179    0.4377    0.4130    0.4515
    0.0900    0.1734   -0.2189   -0.4074    0.8649
S =
    0.0108         0         0         0         0
         0    0.1812         0         0         0
         0         0    1.0000         0         0
         0         0         0    5.5175         0
         0         0         0         0   92.2904
>> U * S * U' - A                    % 检验
ans =
    1.0e-13 *
    0    0.0044    0.0111    0.0133    0.0089
```

```
    0.0044    0.0089    0.0266    0.0178    0.0178
    0.0133    0.0266    0.0533    0.0711    0.0711
    0.0111    0.0178    0.0711    0.1421    0.1421
    0.0044    0.0178    0.0711    0.1421    0.2842
>> [t,t] = rsf2csf(U,S)
t =
    0.0108         0         0         0         0
         0    0.1812         0         0         0
         0         0    1.0000         0         0
         0         0         0    5.5175         0
         0         0         0         0   92.2904
t =
    0.0108         0         0         0         0
         0    0.1812         0         0         0
         0         0    1.0000         0         0
         0         0         0    5.5175         0
         0         0         0         0   92.2904
```

3.5.5 Hessenberg 分解

如果矩阵 H 的第一子对角线下元素都是 0，则 H（或其转置形式）称为上（下）Hessenberg 矩阵。这种矩阵在零元素所占比例及分布上都接近三角矩阵，虽然其在特征值等性质方面不如三角矩阵那样简单，但在实际应用中，应用相似变换将一个矩阵化为 Hessenberg 矩阵是可行的，而化为三角矩阵则是不易实现；而且通过化为 Hessenberg 矩阵来处理矩阵计算问题能够大大节省计算量，因此在工程计算中，Hessenberg 分解也是常用的工具之一。

在 MATLAB 中，提供了 hess 函数实现 Hessenberg 分解。函数的调用格式为：

H＝hess(A)：返回矩阵 A 上的 Hessenberg 分解矩阵 H。

[P,H]＝hess(A)：返回一个上 Hessenberg 矩阵 H 和一个酉矩阵 P，满足 A＝PHP 且 P'P＝I。

[AA,BB,Q,Z]＝hess(A,B)：对于方阵 A、B，返回上 Hessenberg 矩阵 H、上三角矩阵 T 及酉矩阵 Q、U，使得 QAU＝H 且 QBU＝T。

【例 3-44】 对矩阵进行 Hessenberg 分解。

```
>> clear all;
>> A = [ -149 -50 -154;537 180 546; -27 -9 -25];
>> H1 = hess(A)                    % 矩阵的 Hessenberg 分解
H1 =
  -149.0000   42.2037 -156.3165
  -537.6783  152.5511 -554.9272
          0    0.0728    2.4489
>> [P2,H2] = hess(A)               % 矩阵的 Hessenberg 分解
P2 =
    1.0000         0         0
         0   -0.9987    0.0502
```

```
     0         0.0502         0.9987
H2 =
 -149.0000          42.2037         -156.3165
 -537.6783         152.5511         -554.9272
         0           0.0728            2.4489
>> B = P2 * H2 * P2'                  % 验证
B =
 -149.0000         -50.0000         -154.0000
  537.0000         180.0000          546.0000
  -27.0000          -9.0000          -25.0000
>> C = P2' * P2
C =
    1.0000               0               0
         0          1.0000               0
         0               0          1.0000
```

程序中，对于分解的结果，采用公式 A=PHP'和 P'P=eye(size(P))进行验证，满足这两个公式。

3.5.6 SVD 分解

设 A 为 $M\times N$ 矩阵，$A^{\mathrm{H}}A$ 是特征值为 $\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_r\geqslant\lambda_{r+1}=\cdots=\lambda_n=0$，则称 $\sigma_i=\sqrt{\lambda_i}(i=1,2,\cdots,r)$ 为矩阵 A 的奇异值，r 为 A 的秩。存在 M 阶酉矩阵 U 和 N 阶酉矩阵 V，使得 $A=U\begin{bmatrix}\sum & 0_{r\times(N-r)}\\ 0_{(M-r)\times r} & 0_{(M-r)\times(N-r)}\end{bmatrix}V$，其中 $\sum=\begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r\end{bmatrix}$，称为 A 的 SVD(奇异值)分解。

在 MATLAB 中，提供了 svd 函数用于矩阵 SVD 分解。函数的调用格式为：

s=svd(X)：返回矩阵 X 的奇异值向量 s。

[U,S,V]=svd(X)：得到一个与 X 的维数相同的正交矩阵 S 和两个正定矩阵 U 与 V。

[U,S,V]=svd(X,0)：返回 m×n 矩阵 X 的“经济型”奇异值分解。若 m>n，则只计算出矩阵 U 的前 n 列，矩阵 S 为 n×n 矩阵，否则同[U,S,V]=svd(X)。

[U,S,V]=svd(X,'econ')：产生一个“经济型”分解，如果 X 为 m×n 矩阵，且 m>n，则等价于[U,S,V]=svd(X,0)；如果 m<n，则只计算矩阵 V 的前 m 列，S 为 m×n 矩阵。

矩阵的奇异值大小通常决定矩阵的形态，如果矩阵的奇异值变化特别大，则矩阵中某个元素有一个很小的变化会严重影响到原矩阵的参数，又称其为病态矩阵。

【例 3-45】 对矩阵进行 SVD 分解。

```
>> clear all;
>> X = [1 2;3 4;5 6;7 8];
>> [U,S,V] = svd(X)
```

```
U =
   -0.1525         -0.8226         -0.3945         -0.3800
   -0.3499         -0.4214          0.2428          0.8007
   -0.5474         -0.0201          0.6979         -0.4614
   -0.7448          0.3812         -0.5462          0.0407
S =
    14.2691              0
          0         0.6268
          0              0
          0              0
V =
    -0.6414          0.7672
    -0.7672         -0.6414
>> [U,S,V] = svd(X,0)
U =
    -0.1525         -0.8226
    -0.3499         -0.4214
    -0.5474         -0.0201
    -0.7448          0.3812
S =
    14.2691              0
          0         0.6268
V =
    -0.6414          0.7672
    -0.7672         -0.6414
```

3.5.7 特征分解

对 N 阶方阵 A，其特征值 $\lambda_1,\lambda_2,\cdots,\lambda_N$，对应的特征向量为 $v_1,v_2,\cdots,v_N$。令 $A=\begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N\end{bmatrix}$，$V=[v_1,v_2,\cdots,v_N]$，则有 $AV=VA$。如果$\{v_1,v_2,\cdots,v_N\}$线性无关，则 V 可逆，因此有 $A=V\Lambda V^{-1}$。$A=V\Lambda V^{-1}$称为矩阵 A 的特征解(EVD)，对角阵 Λ 称为 A 的标准型，并且称 A 相似于对角阵 Λ，相似于对角阵的矩阵称为对角化矩阵。

MATLAB 提供了 eig 函数用于矩阵的特征分解，在此通过实例来演示 eig 函数实现矩阵的分解。

【例 3-46】 矩阵的特征值分解。

```
>> X = magic(4)
X =
    16          2          3         13
     5         11         10          8
     9          7          6         12
     4         14         15          1
>> A = [1 1 0 1;0 2 3 6;4 7 3 1;0 0 8 9]
A =
```

```
     1         1         0         1
     0         2         3         6
     4         7         3         1
     0         0         8         9
>> E = eig(X)
E =
    34.0000
     8.9443
    -8.9443
     0.0000
>> [V,D] = eig(X)
V =
    -0.5000   -0.8236    0.3764   -0.2236
    -0.5000    0.4236    0.0236   -0.6708
    -0.5000    0.0236    0.4236    0.6708
    -0.5000    0.3764   -0.8236    0.2236
D =
    34.0000         0         0         0
         0    8.9443         0         0
         0         0   -8.9443         0
         0         0         0    0.0000
>> Z = X*V-V*D                          %验证
Z =
    1.0e-13 *
    -0.0355    0.0089   -0.0755    0.0135
    -0.1421    0.0400    0.0092    0.0005
    -0.0355    0.0144    0.0311   -0.0404
    -0.0711   -0.0133   -0.0444    0.0492
>> [V,D] = eig(X,A)
V =
     1.0000    0.3333   -0.0725   -0.4899
    -0.3815    1.0000    0.7584    1.0000
    -0.4186   -1.0000   -1.0000   -0.4306
     0.3300   -0.3333    0.2600    0.7381
D =
    19.2627         0         0         0
         0   -0.0000         0         0
         0         0    0.7796         0
         0         0         0    1.9758
>> Z = X*V-A*V*D
Z =
    1.0e-13 *
    -0.0355    0.0626    0.1110    0.0844
     0.0278   -0.0457    0.0035   -0.0533
    -0.1155   -0.0156   -0.0089   -0.0533
    -0.0355   -0.0568   -0.0178   -0.0888
```