计算的误差，有时会出现病态，即结果出现很大偏差. 因而，在求解线性方程组时，一定要检验其解的正确程度，看是否出现了病态.

第三节　矩阵特征值与特征向量

特征值与特征向量是线性代数中一个非常重要的概念，在工农业生产实际问题的求解中有着广泛的应用.

一、求矩阵的特征值与特征向量

特征值 λ 与特征向量 X 满足关系 $AX=\lambda X$，而对于矩阵 A 和 B，其广义特征值 λ 与广义特征向量 X 满足关系 $AX=\lambda BX$，这里 A 与 B 为同阶方阵. 在 MATLAB 中可用如下函数求矩阵的特征值与特征向量.

$d=\text{eig}(A)$　　返回由矩阵特征值组成的列向量.

$[v,d]=\text{eig}(A)$　　返回特征向量矩阵 v 和特征值矩阵 d，其中 d 为对角阵，且满足 $Av=vd$.

若将 eig(A)换成 eig(A,B)，则返回广义特征值与特征向量，且满足 $Av=Bvd$，各特征向量的范数为 1. 若 B 可逆，则广义特征值问题等价于求 inv(B)A 的常义特征值问题. 如

```
A=[ 1   4   2; 0   -3   4;   0   4   3 ];
[v,d]=eig(A)
v=[ 1.0000     0.4082     -0.6667
    0.0000    -0.8165     -0.3333
    0.0000     0.4082    -0.6667]
d=[  1     0    0
     0    -5    0
     0     0    5]
```

二、矩阵的对角化

下述函数用来判断矩阵是否可对角化，若可对角化返回 1，否则返回 0.

```
function y=trigle(A)
y=1;c=size(A);
if c(1)~=c(2)
   y=0;
   return;
end
```

```
e = eig(A);n = length(A);
while 1
   if isempty(e)
      return;
   end
   d = e(1);
   f = sum(abs(e - d) < 10 * eps);
   g = n - rank(A - d * eye(n));
   if f ~ = g
      y = 0;
      return;
   end
   e(find(abs(e - d) < 10 * eps)) = [];
end
```

当一个矩阵可对角化时,用该矩阵的特征向量 P 得到对角化后的矩阵,即特征值矩阵,这里 P 是可逆的,且 inv(P)AP 为特征矩阵.

实对称矩阵都是可对角化的,并且存在正交矩阵 Q,使得 inv(Q)AQ 为对角阵,这里 Q 可由特征向量阵正交规范化得到.事实上,对于实对称矩阵 A 来说 eig(A) 返回的特征向量就是正交矩阵.如

```
D = [1 2 2;2 1 2; 2 2 1];
trigle(D) =      1
[l , k] = eig(D)
l =       - 0.7850     0.2246     0.5774
           0.5870     0.5676     0.5774
           0.1980    - 0.7921     0.5774
k =     - 1.0000             0             0
             0          - 1.0000             0
             0                0          5.0000
inv(l) * D * l =    - 1.0000          0   0.0000
                     0.0000    - 1.0000   0.0000
                     0.0000     0.0000   5.0000
F = [0 1 1 - 1;1 0 - 1 1;1 - 1 0 1; - 1 1 1 0];
[Fd , Fv] = eig(F)
```

```
Fd =   0.7887    0.2113    0.5000   -0.2887
       0.2113    0.7887   -0.5000    0.2887
       0.5774   -0.5774   -0.5000    0.2887
            0         0    0.5000    0.8660
Fv =   1.0000         0         0         0
            0    1.0000         0         0
            0         0   -3.0000         0
            0         0         0    1.0000
Fd' * Fd =   1.0000    0.0000    0.0000    0.0000
             0.0000    1.0000    0.0000    0.0000
             0.0000    0.0000    1.0000    0.0000
             0.0000    0.0000    0.0000    1.0000
Fd' * F * Fd =   1.0000    0.0000    0.0000    0.0000
                 0.0000    1.0000    0.0000    0.0000
                 0         0        -3.0000    0.0000
                 0.0000    0.0000    0.0000    1.0000
```

三、矩阵相似与 Jordan 标准形

两个矩阵 A 与 B 相似，即存在可逆矩阵 P，使得 $A = P^{-1}BP$ 成立，其中 P 称为过渡矩阵. 由线性代数理论知，任何一个方阵都与一个 Jordan 标准形相似，并且这个 Jordan 标准形是唯一的. 因而，判断两个矩阵是否相似，就归结成看它们是否能相似于同一 Jordan 标准形. 在 MATLAB 中没有直接判断两矩阵是否相似的函数. 下面我们来构造这个函数.

首先判断矩阵 A 是否可以构成 Jordan 块. 由于只有方阵才有标准形，故等价于判断矩阵 A 是否为方阵. 然后分别用函数 diaglink( )、lk( )、lkk( )、cc( )、Jordan( )和 alikeb( ) 来判别矩阵 A 是否与矩阵 B 相似，并给出由 A 转化为 B 的过渡矩阵 P. 其中，判断是否为方阵用函数 besquare( ). 这些函数及其作用如下

```
function y = besquare(A)
% 判断 A 矩阵是否为方阵
% 返回 1 为方阵,返回 0 非方阵
c = size(A);
nx = c(1);ny = c(2);
y = 1;
if nx~ = ny
```

```
    y = 0;
    return;
end
function y = diaglink(A,B)
% 连接两个 Jordan 块 A 和 B
% 返回连接成功后的矩阵
if ~besquare(A)|~besquare(B)
    error('你输入的矩阵不是 Jordan 块');
    return;
end
if isempty(A)
    if length(B)>1
      y = diag(diag(B)) + diag(diag(B,1),1);
    else
      y = B;
    end
elseif isempty(B)
    if length(A)>1
      y = diag(diag(A)) + diag(diag(A,1),1);
    else
      y = A;
    end
else
    da = diag(A)';
    db = diag(B)';
    da1 = diag(A,1)';
    db1 = diag(B,1)';
    if length(A) = = 1
      da1 = [];
    end
    if length(B) = = 1
      db1 = [];
    end
    d = [da db];
    d1 = [da1 0 db1];
```

```
    c1 = diag(d);
    c2 = diag(d1,1);
    y = c1 + c2;
end
function y = lk(k,n)
% 生成特征值 k 的 n 阶 Jordan 块
% n 应为整数,且大于 0
n = fix(n);
if n<1
    error('n 必须大于 0.');
    return
end
if n = = 1
    y = k;
elseif n>1
    c = diag(ones(1,(n - 1)),1);
    d = k * eye(n);
    y = c + d;
end
function [cc,y] = lkk(A,k,v)
% 生成矩阵 A 的关于特征值 k 的所有 Jordan 块
% 并加以连接,同时返回特征值 k 的特征向量
% 和广义特征向量构成的矩阵 cc,用来构成最终
% 的可逆矩阵 P,使 A 化成 Jordan 标准形。
% 向量 v 是特征值 k 的 Jordan 块的说明向量,
% v(i)表示 i 重 Jordan 块的个数,i 表示 Jordan 块的重数,
n = length(v);
AA = A - k * eye(length(A));
D = null(AA);
kk = 1;cc = [];y = [];
for i = 1:n
  for j = 1:v(i)
    cc = [cc D(:,kk)];
    b = D(:,kk);
    for l = 2:i
```

```
        x = pinv(AA) * b;
        cc = [cc x];
        b = x;
      end
      kk = kk + 1;
      c = lk(k, i);
      y = diaglink(y, c);
    end
end
function [ks, kn, kr] = cc(A)
% 生成矩阵 A 的特征向量 ks,特征值的代数重数组成的向量 kn,
% 及几何重数 kr.若特征值为复数,该函数将虚部去除。
[v, e] = eig(A);
e = diag(e);
ne = length(e);
for i = 1:ne - 1
  for j = i + 1:ne
    if e(i)>e(j)
      temp = e(i);e(i) = e(j);e(j) = temp;
    end
  end
end
e = real(e);
f = e;
n = length(A);
ks = [];kn = [];kr = [];
while~isempty(e)
  c = e(1);
  ks = [ks c];
  s = sum(abs(e - c)<0.0001);
  F = v(:, find(abs(f - c)<0.0001));
  r = rank(F);
  kn = [kn s];
  kr = [kr r];
  e(find(abs(e - c)<0.0001)) = [];
```

```
end
function [p,y] = jordan(A)
% 生成Jordan矩阵标准形 y,及过渡矩阵 p。
if ~besquare(A)
  error('矩阵必须为方阵');
  return;
end
y = [];n = length(A);
p = [];
if n = = 0
  error('不能是空矩阵。');
   return;
end
if n = = 1
  p = 1;y = A;
  return;
end
y = [];n = length(A);
p = [];
[ks kn kr] = cc(A);
for i = 1:length(ks)
  v = [];
  if kn(i) = = 1
    v(1) = 1;
  elseif kr(i) = = 1
    v(kn(i)) = 1;
  elseif kn(i) = = kr(i)
    v(1) = kn(i);
  else
    v(1) = 0;
    RA = A - ks(i) * eye(n);
    j = 2;
    while 1
      v(j) = rank(RA^(j - 1)) + rank(RA^(j + 1)) - 2 * rank(RA^j);
      ss = 0;rr = 0;
```

```
        for ii = 2:j
          ss = ss + ii * v(ii);
          rr = rr + v(ii);
        end
        aa = kn(i) - ss;
        if aa = = (kr(i) - rr)
          v(1) = aa;
          break;
        end
        j = j + 1;
      end
    end
    [cc, AA] = lkk(A, ks(i), v);
    p = [p cc];
    y = diaglink(y, AA);
end
function [y, p] = alikeb(A,B)
% 判别矩阵 A 与 B 是否相似,相似返回 1,否则返回 0。
% 同时返回过渡矩阵 p.
y = 1;p = [];
if ~besquare(A) | ~besquare(B)
  y = 0;
  error('矩阵不是方阵。');
   return;
end
na = length(A);nb = length(B);
if na~ = nb
  y = 0;
  error('两矩阵不同阶。');
   return;
end
n = na;
ea = eig(A);ea = real(ea);
eb = eig(B);eb = real(eb);
for i = 1:n - 1
```

```
    for j = i + 1:n
      if ea(i) > ea(j)
        temp = ea(i);ea(i) = ea(j);ea(j) = temp;
      end
      if eb(i) > eb(j)
        temp = eb(i);eb(i) = eb(j);eb(j) = temp;
      end
    end
  end
  if ~all(abs(ea - eb) < 10 * eps)
    y = 0;
    return;
  end
  [ya ja] = jordan(A);
  [yb jb] = jordan(B);
  if ~all(abs(ja - jb) < 10 * eps)
    y = 0;
    return;
  end
  p = yb * inv(ya);
```

如定义并判断矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 是否相似，结果如下

```
A =   3   0    2   -1
      0   3   -2    2
      0   0    3    0
      0   0    0    3
B =   3   0    4   1
      0   3   -2   0
      0   0    3   0
      0   0    0   3
[is,P] = alikeb(A,B)
is =     1
P = 1.0000         0         0         0
         0   -1.0000    0.0000    0.0000
         0    0.0000   -1.0000    1.0000
         0    0.0000    6.0000   -5.0000
```

验证 inv(**P**) * **B** * **P** 是否为 **A**

```
inv(P) * B * P
ans = 3.0000  0.0000   2.0000  -1.0000
          0   3.0000  -2.0000   2.0000
          0   0.0000   3.0000   0.0000
          0   0.0000   0.0000   3.0000
```

四、化二次型为标准形

对于线性空间上的二次齐次函数

$$f(x_1,x_2,\cdots,x_n)=a_{11}x_1^2+a_{22}x_2^2+\cdots+a_{mn}x_n^2+2a_{12}x_1x_2+2a_{13}x_1x_3+\cdots+2a_{n-1n}x_{n-1}x_n,$$

若令 $a_{ij}=a_{ji}$,则由 a_{ij} 构成的对称矩阵 $\boldsymbol{A}=(a_{ij})$ 与该函数相对应,这个函数称为二次型.函数

$$f(x_1,x_2,\cdots,x_n)=k_1x_1^2+k_2x_2^2+\cdots+k_nx_n^2$$

称为标准二次型或标准型,对于任何二次型 $f=\boldsymbol{X}'\boldsymbol{AX}$,都可以通过一个正交阵 $\boldsymbol{P}$ 将其化为标准型.如

```
[p,e] = eig(A)
p =   0.5774   0.4082  -0.5000  -0.5000
      0.4082  -0.5774  -0.5000   0.5000
      0.5774   0.4082   0.5000   0.5000
      0.4082  -0.5774   0.5000  -0.5000
e =   1.0000        0        0        0
           0   1.0000        0        0
           0        0   3.0000        0
           0        0        0  -1.0000
inv(p) * A * p
ans = 1.0000   0.0000   0.0000   0.0000
      0.0000   1.0000   0.0000   0.0000
           0   0.0000   3.0000   0.0000
      0.0000   0.0000   0.0000  -1.0000
inv(p)
ans = 0.5774   0.4082   0.5774   0.4082
      0.4082  -0.5774   0.4082  -0.5774
     -0.5000  -0.5000   0.5000   0.5000
     -0.5000   0.5000   0.5000  -0.5000
```