第 5 章　矩阵特征值与特征向量的计算

5.1　约化对称矩阵为对称三对角阵的豪斯荷尔德变换法

【功能】 用豪斯荷尔德(Householder)变换将 n 阶实对称矩阵约化为对称三对角阵。

【方法说明】 化 n 阶实对称矩阵 $\boldsymbol{A}$ 为对称三对角阵的豪斯荷尔德方法,就是使 $\boldsymbol{A}$ 经过 $n-2$ 次正交变换后变成三对角阵 $\boldsymbol{A}_{n-2}$。即

$$\boldsymbol{A}_{n-2} = P_{n-3}\cdots P_1 P_0 \boldsymbol{A} P_0 P_1 \cdots P_{n-3}$$

每一次的正交变换 $P_i(i=0,1,\cdots,n-3)$具有如下形式:

$$\boldsymbol{P}_i = I - U_i U_i^{\mathrm{T}} / H_i$$

其中 $\boldsymbol{P}_i$ 为对称正交矩阵,且

$$H_i = \frac{1}{2} U_i^T U_i$$

$$U_i = (a_{t0}^{(i)}, a_{t1}^{(i)}, \cdots, a_{t,t-2}^{(i)}, a_{t,t-1}^{(i)} \pm \sigma_i^{1/2}, 0, \cdots, 0)^{\mathrm{T}}$$

$$\sigma_i = (a_{t0}^{(i)})^2 + (a_{t1}^{(i)})^2 + \cdots + (a_{t,t-1}^{(i)})^2$$

式中 $t=n-i-1$。

对 $\boldsymbol{A}$ 的每一次变换为

$$A_{i+1} = P_i A_i P_i = (I - U_i U_i^{\mathrm{T}} / H_i) A_i (I - U_i U_i^{\mathrm{T}} / H_i)$$

若令

$$\begin{cases} s_i = A_i U_i / H_i \\ k_i = U_i^{\mathrm{T}} s_i / (2H_i) \\ q_i = s_i - k_i U_i \end{cases}$$

则有

$$A_{i+1} = A_i - U_i q_i^{\mathrm{T}} - q_i U_i^{\mathrm{T}}$$

其中 s_i 的形式为

$$s_i = (s_{i0}, s_{i1}, \cdots, s_{it}, 0, \cdots, 0)^{\mathrm{T}}$$

q_i 的形式为

$$q_i = (s_{i0} - k_i u_{i0}, s_{i1} - k_i u_{i1}, \cdots, s_{i,t-1} - k_i u_{i,t-1}, s_{it}, 0, \cdots, 0)^{\mathrm{T}}$$

本函数与 5.2 节的函数 sstq()联用,可以计算实对称矩阵 $\boldsymbol{A}$ 的全部特征值与相应的特征向量。

【函数语句与形参说明】

```
void strq(a,n,q,b,c)
```

形参与函数类型	参　数　意　义
double　a[n][n]	存放 n 阶实对称矩阵 $\boldsymbol{A}$
int　n	实对称矩阵 $\boldsymbol{A}$ 的阶数

续表

形参与函数类型	参 数 意 义
double q[n][n]	返回豪斯荷尔德变换的乘积矩阵 $\boldsymbol{Q}$。在与 5.2 节中的函数 sstq()联用时，若将 $\boldsymbol{Q}$ 矩阵作为函数 sstq()中的一个参数，则可以计算一般实对称矩阵的全部特征值及相应的特征向量
double b[n]	返回对称三角阵中的主对角线元素
double c[n]	前 $n-1$ 个元素返回对称三角阵中的次对角线元素
void strq()	过程

由上述参数可知，本函数返回的对称三角阵为

$$\boldsymbol{T}=\begin{bmatrix} b_0 & c_0 & & & & 0 \\ c_0 & b_1 & c_1 & & & \\ & c_1 & b_2 & c_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-3} & b_{n-2} & c_{n-2} \\ 0 & & & & c_{n-2} & b_{n-1} \end{bmatrix}$$

【函数程序】（文件名：5strq.c）

```
#include "math.h"
void strq(a,n,q,b,c)
int n;
double a[],q[],b[],c[];
{ int i,j,k,u;
  double h,f,g,h2;
  for (i=0; i<=n-1; i++)
  for (j=0; j<=n-1; j++)
    { u=i*n+j; q[u]=a[u];}
  for (i=n-1; i>=1; i--)
    { h=0.0;
      if (i>1)
        for (k=0; k<=i-1; k++)
          { u=i*n+k; h=h+q[u]*q[u];}
      if (h+1.0==1.0)
        { c[i]=0.0;
          if (i==1) c[i]=q[i*n+i-1];
          b[i]=0.0;
        }
      else
        { c[i]=sqrt(h);
          u=i*n+i-1;
          if (q[u]>0.0) c[i]=-c[i];
          h=h-q[u]*c[i];
          q[u]=q[u]-c[i];
```

```
        f=0.0;
        for (j=0; j<=i-1; j++)
          { q[j*n+i]=q[i*n+j]/h;
            g=0.0;
            for (k=0; k<=j; k++)
              g=g+q[j*n+k]*q[i*n+k];
            if (j+1<=i-1)
              for (k=j+1; k<=i-1; k++)
                g=g+q[k*n+j]*q[i*n+k];
            c[j]=g/h;
            f=f+g*q[j*n+i];
          }
        h2=f/(h+h);
        for (j=0; j<=i-1; j++)
          { f=q[i*n+j];
            g=c[j]-h2*f;
            c[j]=g;
            for (k=0; k<=j; k++)
              { u=j*n+k;
                q[u]=q[u]-f*c[k]-g*q[i*n+k];
              }
          }
        b[i]=h;
      }
  }
for (i=0; i<=n-2; i++) c[i]=c[i+1];
c[n-1]=0.0;
b[0]=0.0;
for (i=0; i<=n-1; i++)
  { if ((b[i]!=0.0)&&(i-1>=0))
      for (j=0; j<=i-1; j++)
        { g=0.0;
          for (k=0; k<=i-1; k++)
            g=g+q[i*n+k]*q[k*n+j];
          for (k=0; k<=i-1; k++)
            { u=k*n+j;
              q[u]=q[u]-g*q[k*n+i];
            }
        }
    u=i*n+i;
    b[i]=q[u]; q[u]=1.0;
    if (i-1>=0)
      for (j=0; j<=i-1; j++)
        { q[i*n+j]=0.0; q[j*n+i]=0.0;}
  }
```

```
  return;
}
```

【例】 用豪斯荷尔德变换将下列 5 阶实对称矩阵约化为对称三对角阵，并给出豪斯荷尔德变换的乘积矩阵 $\boldsymbol{Q}$：

$$\boldsymbol{A}=\begin{bmatrix} 10 & 1 & 2 & 3 & 4 \\ 1 & 9 & -1 & 2 & -3 \\ 2 & -1 & 7 & 3 & -5 \\ 3 & 2 & 3 & 12 & -1 \\ 4 & -3 & -5 & -1 & 15 \end{bmatrix}$$

主函数程序(文件名：5strq0.c)如下：

```
#include "stdio.h"
#include "5strq.c"
main()
{ int i,j;
  double b[5],c[5],q[5][5];
  double a[5][5]={ {10.0,1.0,2.0,3.0,4.0},
      {1.0,9.0,-1.0,2.0,-3.0},{2.0,-1.0,7.0,3.0,-5.0},
      {3.0,2.0,3.0,12.0,-1.0},{4.0,-3.0,-5.0,-1.0,15.0}};
  strq(a,5,q,b,c);
  printf("MAT A IS:\n");
  for (i=0; i<=4; i++)
    { for (j=0; j<=4; j++)
        printf("%13.6e ",a[i][j]);
      printf("\n");
    }
  printf("\n");
  printf("MAT Q IS:\n");
  for (i=0; i<=4; i++)
    { for (j=0; j<=4; j++)
        printf("%13.6e ",q[i][j]);
      printf("\n");
    }
  printf("\n");
  printf("MAT B IS:\n");
  for (i=0; i<=4; i++)
    printf("%13.6e ",b[i]);
  printf("\n\n");
  printf("MAT C IS:\n");
  for (i=0; i<=4; i++)
    printf("%13.6e ",c[i]);
  printf("\n\n");
}
```

运行结果为

```
MAT A IS:
1.000000e+001   1.000000e+000   2.000000e+000   3.000000e+000   4.000000e+000
1.000000e+000   9.000000e+000  -1.000000e+000   2.000000e+000  -3.000000e+000
2.000000e+000  -1.000000e+000   7.000000e+000   3.000000e+000  -5.000000e+000
3.000000e+000   2.000000e+000   3.000000e+000   1.200000e+001  -1.000000e+000
4.000000e+000  -3.000000e+000  -5.000000e+000  -1.000000e+000   1.500000e+001

MAT Q IS:
-3.845416e-002  -8.269598e-001   3.054904e-002   5.601120e-001  0.000000e+000
-8.685653e-001  -2.401904e-001   1.069216e-001  -4.200840e-001  0.000000e+000
 4.492444e-001  -5.037103e-001  -2.329364e-001  -7.001400e-001  0.000000e+000
 2.056576e-001  -6.871667e-002   9.661134e-001  -1.400280e-001  0.000000e+000
 0.000000e+000   0.000000e+000   0.000000e+000   0.000000e+000  1.000000e+000

MAT B IS:
9.295202e+000  1.162671e+001  1.096044e+001  6.117647e+000  1.500000e+001

MAT C IS:
-7.494847e-001  -4.496268e+000  -2.157041e+000  7.141428e+000  0.000000e+000
```

即返回的三对角阵为

$$\boldsymbol{T}=\begin{bmatrix} 9.295202 & -0.7494847 & & & \\ -0.7494847 & 11.62671 & -4.496268 & & \\ & -4.496268 & 10.96044 & -2.157041 & \\ & & -2.157041 & 6.117647 & 7.141428 \\ & & & 7.141428 & 15.00000 \end{bmatrix}$$

5.2 求对称三对角阵的全部特征值与特征向量

【功能】 用变形 QR 方法计算实对称三对角阵的全部特征值与相应的向量。

【方法说明】 本函数采用变形的 QR 方法。有关 QE 方法的说明参见 5.4 节。

本函数与 5.1 节中的函数 strq()联用,可以计算一般实对称矩阵的全部特征值与相应的向量。

【函数语句与形参说明】

```
int sstq(n,b,c,q,eps,l)
```

形参与函数类型	参 数 意 义
int n	实对称三角阵的阶数
double b[n]	存放 n 阶实对称三角阵的主对角线上的元素,返回时存放全部特征值
double c[n]	前 $n-1$ 个元素存放实对称三角阵的次对角线上的元素

续表

形参与函数类型	参 数 意 义
double q[n][n]	若存放 n 阶单位矩阵，则返回实对称三对角阵 $\boldsymbol{T}$ 的特征向量组；若存放由5.1节中的函数 strq()所返回的一般实对称矩阵 $\boldsymbol{A}$ 的豪斯荷尔德变换的乘积矩阵 $\boldsymbol{Q}$，则返回实对称矩阵 $\boldsymbol{A}$ 的特征向量组。其中 q 中的第 j 列为与数组 b 中第 j 个特征值对应的特征向量
double eps	控制精度要求
int l	允许的最大迭代次数
int sstq()	函数返回标志值。若返回的标志值小于0，则表示程序工作失败；若返回的标志值大于0，则说明程序正常返回

由上述参数可知，n 阶实对称三角阵为

$$\boldsymbol{T}=\begin{bmatrix} b_0 & c_0 & & & & 0 \\ c_0 & b_1 & c_1 & & & \\ & c_1 & b_2 & c_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-3} & b_{n-2} & c_{n-2} \\ 0 & & & & c_{n-2} & b_{n-1} \end{bmatrix}$$

【函数程序】（文件名：5sstq.c）

```
#include "stdio.h"
#include "math.h"
int sstq(n,b,c,q,eps,l)
int n,l;
double b[],c[],q[],eps;
{ int i,j,k,m,it,u,v;
  double d,f,h,g,p,r,e,s;
  c[n-1]=0.0; d=0.0; f=0.0;
  for (j=0; j<=n-1; j++)
    { it=0;
      h=eps*(fabs(b[j])+fabs(c[j]));
      if (h>d) d=h;
      m=j;
      while ((m<=n-1)&&(fabs(c[m])>d)) m=m+1;
      if (m!=j)
        { do
            { if (it==1)
                { printf("fail\n");
                  return (-1);
                }
              it=it+1;
              g=b[j];
              p=(b[j+1]-g)/(2.0*c[j]);
```

```
            r=sqrt(p*p+1.0);
            if (p>=0.0) b[j]=c[j]/(p+r);
            else b[j]=c[j]/(p-r);
            h=g-b[j];
            for (i=j+1; i<=n-1; i++)
              b[i]=b[i]-h;
            f=f+h; p=b[m]; e=1.0; s=0.0;
            for (i=m-1; i>=j; i--)
              { g=e*c[i]; h=e*p;
                if (fabs(p)>=fabs(c[i]))
                  { e=c[i]/p; r=sqrt(e*e+1.0);
                    c[i+1]=s*p*r; s=e/r; e=1.0/r;
                  }
                else
          { e=p/c[i]; r=sqrt(e*e+1.0);
                    c[i+1]=s*c[i]*r;
                    s=1.0/r; e=e/r;
                  }
                p=e*b[i]-s*g;
                b[i+1]=h+s*(e*g+s*b[i]);
                for (k=0; k<=n-1; k++)
                  { u=k*n+i+1; v=u-1;
                    h=q[u]; q[u]=s*q[v]+e*h;
                    q[v]=e*q[v]-s*h;
                  }
              }
            c[j]=s*p; b[j]=e*p;
          }
        while (fabs(c[j])>d);
      }
    b[j]=b[j]+f;
  }
for (i=0; i<=n-1; i++)
  { k=i; p=b[i];
    if (i+1<=n-1)
      { j=i+1;
        while ((j<=n-1)&&(b[j]<=p))
          { k=j; p=b[j]; j=j+1;}
      }
    if (k!=i)
      { b[k]=b[i]; b[i]=p;
        for (j=0; j<=n-1; j++)
          { u=j*n+i; v=j*n+k;
            p=q[u]; q[u]=q[v]; q[v]=p;
          }
```

```
        }
    }
  return (1);
}
```

【例】 与 5.1 节中的函数 strq()联用,计算下列 5 阶实对称矩阵的全部特征值与相应的特征向量:

$$\boldsymbol{A}=\begin{bmatrix}10 & 1 & 2 & 3 & 4\\ 1 & 9 & -1 & 2 & -3\\ 2 & -1 & 7 & 3 & -5\\ 3 & 2 & 3 & 12 & -1\\ 4 & -3 & -5 & -1 & 15\end{bmatrix}$$

取最大迭代次数为 60,控制精度要求 $\varepsilon=0.000001$。

主函数程序(文件名:5sstq0.c)如下:

```
#include "stdio.h"
#include "5sstq.c"
#include "5strq.c"
main()
{ int i,j,k,l=60;
  double q[5][5],b[5],c[5];
  double a[5][5]={ {10.0,1.0,2.0,3.0,4.0},
     {1.0,9.0,-1.0,2.0,-3.0},{2.0,-1.0,7.0,3.0,-5.0},
     {3.0,2.0,3.0,12.0,-1.0},{4.0,-3.0,-5.0,-1.0,15.0}};
  double eps=0.000001;
  strq(a,5,q,b,c);
  k=sstq(5,b,c,q,eps,l);
  printf("MAT A IS:\n");
  for (i=0; i<=4; i++)
    { for (j=0; j<=4; j++)
        printf("%13.6e ",a[i][j]);
      printf("\n");
    }
  printf("\n");
  if (k>0)
    { printf("MAT B IS:\n");
      for (i=0; i<=4; i++)
        printf("%13.6e ",b[i]);
      printf("\n\n");
      printf("MAT Q IS:\n");
      for (i=0; i<=4; i++)
        { for (j=0; j<=4; j++)
            printf("%13.6e ",q[i][j]);
          printf("\n");
```

```
        }
      printf("\n");
    }
}
```

运行结果为

```
MAT A IS:
1.000000e+001   1.000000e+000   2.000000e+000   3.000000e+000   4.000000e+000
1.000000e+000   9.000000e+000  -1.000000e+000   2.000000e+000  -3.000000e+000
2.000000e+000  -1.000000e+000   7.000000e+000   3.000000e+000  -5.000000e+000
3.000000e+000   2.000000e+000   3.000000e+000   1.200000e+001  -1.000000e+000
4.000000e+000  -3.000000e+000  -5.000000e+000  -1.000000e+000   1.500000e+001

MAT B IS:
6.994838e+000  9.365555e+000  1.655266e+000  1.580892e+001  1.917542e+001

MAT Q IS:
 6.540830e-001   5.215112e-002   3.872969e-001  -6.237025e-001   1.745051e-001
 1.996813e-001  -8.599639e-001  -3.662210e-001  -1.591011e-001  -2.473025e-001
 2.565105e-001   5.055751e-001  -7.043773e-001  -2.272975e-001  -3.616417e-001
-6.604027e-001   2.011666e-004   1.189262e-001  -6.926844e-001  -2.644109e-001
-1.742799e-001  -4.621920e-002  -4.534231e-001  -2.328223e-001   8.412441e-001
```

5.3 约化一般实矩阵为赫申伯格矩阵的初等相似变换法

【功能】 用初等相似变换将一般实矩阵约化为上 H 矩阵，即赫申伯格(Hessenberg)矩阵。

【方法说明】 为了要将矩阵 $\boldsymbol{A}$ 化为上 H 矩阵，只需要依次将矩阵 $\boldsymbol{A}$ 中的每一列都化为上 H 矩阵就行了。因此，对于 k 从 2 到 $n-1$(其中第 $n-1$ 列与第 n 列都已经是上 H 矩阵)做如下操作：

(1) 从第 $k-1$ 列的第 $k-1$ 个以下的元素中选出绝对值最大的元素 $a_{m,k-1}$。

(2) 交换第 m 行和第 k 行，交换第 m 列和第 k 列。

(3) 对于 $i=k+1,\cdots,n$ 进行如下变换：

将第 $k-1$ 列的第 i 个元素消成 0：$t_i=a_{i,k-1}/a_{k,k-1},0\Rightarrow a_{i,k-1}$

第 i 行中的其他元素做相应的消元变换：$a_{ij}-t_i a_{kj}\Rightarrow a_{ij},j=k,\cdots,n$

第 i 列中的其他元素也要做相应的变换：$a_{jk}+t_i a_{ji}\Rightarrow a_{jk},j=1,2,\cdots,n$

上述的相似变换过程可以表示成

$$\boldsymbol{A}_k = \boldsymbol{N}_k^{-1}\boldsymbol{I}_{km}\boldsymbol{A}_{k-1}\boldsymbol{I}_{km}\boldsymbol{N}_k, \quad k=2,\cdots,n-1$$

其中 $\boldsymbol{A}_1=\boldsymbol{A}$；$\boldsymbol{I}_{km}$ 是初等置换矩阵(表示第 k 和第 m 行或列的交换)；$\boldsymbol{N}_k$ 是一个初等矩阵，即

$$(N_k)_{ij}=\begin{cases} t_i, & i=k+1,\cdots,n; j=k \\ \delta_{ij}, & \text{其他} \end{cases}$$

【函数语句与形参说明】

void hhbg(a,n)

形参与函数类型	参 数 意 义
double a[n][n]	存放一般实矩阵 A,返回上 H 矩阵
int n	矩阵的阶数
void hhbg()	过程

【函数程序】（文件名：5hhbg.c）

```
#include "math.h"
void hhbg(a,n)
int n;
double a[];
{ int i,j,k,u,v;
  double d,t;
  for (k=1; k<=n-2; k++)
    { d=0.0;
      for (j=k; j<=n-1; j++)
        { u=j*n+k-1; t=a[u];
          if (fabs(t)>fabs(d))
            { d=t; i=j;}
        }
      if (fabs(d)+1.0!=1.0)
        { if (i!=k)
            { for (j=k-1; j<=n-1; j++)
                { u=i*n+j; v=k*n+j;
                  t=a[u]; a[u]=a[v]; a[v]=t;
                }
              for (j=0; j<=n-1; j++)
                { u=j*n+i; v=j*n+k;
                  t=a[u]; a[u]=a[v]; a[v]=t;
                }
            }
          for (i=k+1; i<=n-1; i++)
            { u=i*n+k-1; t=a[u]/d; a[u]=0.0;
              for (j=k; j<=n-1; j++)
                { v=i*n+j;
                  a[v]=a[v]-t*a[k*n+j];
                }
              for (j=0; j<=n-1; j++)
                { v=j*n+k;
                  a[v]=a[v]+t*a[j*n+i];
                }
```