题目

试用带双步位移的 QR 分解法求矩阵 A 的全部特征值，并对其中的每一个实特征值求相应的特征向量。已知

$$a_{ij} = \begin{cases} sin(0.5i + 0.2j) & i \neq j \\ 1.5cos(i + 1.2j) & i = j \end{cases} \quad (i, j = 1,2, \cdots ,10)$$

一、算法设计方案：

1. 根据题设条件求出矩阵 A；
2. 对矩阵 A 进行拟上三角化(具体算法如下)：
   记 $A^{(1)}$=A，并记 $A^{(r)}$的第 r 列至第 n 列的元素为 $a^{(r)}_{ij}(i = 1, 2, \cdots, n;\ j = r, r+1, \cdots, n)$。
   对于 r=1,2,⋯,n-2 执行
   (1)若 $a^{(r)}_{ir}(i = r+2, r+3, \cdots, n)$全为 0，则令 $A^{(r+1)}$=$A^{(r)}$，转(5)；否则转(2)。
   (2)计算

$$d_r = \sqrt{\sum_{i=r+1}^{n} \left(a_{ir}^{(r)}\right)^2}$$

$$c_r = -sgn\left(a_{r+1,r}^{(r)}\right)d_r \qquad (\text{若}a_{r+1,r}^{(r)}=0\text{，则取}c_r=d_r)$$

$$h_r = c_r^2 - c_r a_{r+1,r}^{(r)}$$

   (3)令$\boldsymbol{u}_r = \left(0, \ldots ,0, a_{r+1,r}^{(r)} - c_r, a_{r+2,r}^{(r)}, \ldots a_{n,r}^{(r)}\right)^T \in \mathbf{R}^n$。
   (4)计算

$$\boldsymbol{p}_r = \boldsymbol{A}^{(r)T}\boldsymbol{u}_r/h_r$$
$$\boldsymbol{q}_r = \boldsymbol{A}^{(r)}\boldsymbol{u}_r/h_r$$
$$t_r = \boldsymbol{p}_r^T\boldsymbol{u}_r/h_r$$
$$\boldsymbol{\omega}_r = \boldsymbol{q}_r - t_r\boldsymbol{u}_r$$
$$\boldsymbol{A}^{(r+1)} = \boldsymbol{A}^{(r)} - \boldsymbol{\omega}_r\boldsymbol{u}_r^T - \boldsymbol{u}_r\boldsymbol{p}_r^T$$

   (5)继续。
3. 对拟上三角化后的矩阵$\boldsymbol{A}^{(r+1)}$进行带双步位移的 QR 分解，求出所有的特征值(简要步骤如下)：
   (1)令 $A_1=A^{(n-1)}$；
   (2)计算$M_k = A_k^2 - tr(D_k)A_k + det(D_k)I$；其中$D_k$为$A_k$的最后二阶子式。
   (3)对M_k进行 QR 分解；
   (4)计算$A_{k+1} = Q_k^T A_k Q_k$；
   则随着迭代次数 k 的增加，$\{A_k\} \to B$。
   B为分块上三角矩阵，其对角块均为一阶和二阶子块，并且对角块中的每一个一阶子块给出 A 的实特征值，每一个二阶子块给出 A 的一对复共轭特征值。
4. 求出矩阵 A 的所有特征值后，根据相应特征值的重数 k 来设置相应特征向量的最后 k 个元素为常数(本程序中此常数为 1)，以消除相应的系数矩阵$(A-\lambda_i I)$行列式为 0 的影响。最后利用选主元的 GAUSS 消去法来求解相应的特征向量。

二、全部源程序

//1.主程序

```
#include "iostream.h"
#include "selfdef_c.h"
#include "math.h"
#include "fstream.h"
#include "iomanip.h"

void main()
{
	int r, i;//r 代表最外层循环次数,i 代表内层循环次数
	const int n = 10;//n 代表矩阵 A 的维数
	const double e = 1e-12;
	fstream fin;
	fin.open("record.txt", ios::trunc );
	//为矩阵 A(循环中使用)分配内存空间
	double **a;
	a = new double *[n];
	for ( i = 0; i < n; ++i )
	{
		a[i] = new double[n];
	}
//生成矩阵 A,即为矩阵 A 赋值
	ger_a( a, n );
	//输出矩阵 A
	cout<<"A="<<endl;
	cout<<setiosflags( ios::scientific )<<setprecision(5);
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{

	cout<<setw(15)<<a[r][i]<<setw(15);
			}
		cout<<endl;
	}
	//记录矩阵 A
	fin<<"A="<<endl;
	fin<<setiosflags( ios::scientific )<<setprecision(5);
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{
			fin<<setw(15)<<a[r][i]<<setw(15);
		}
		fin<<endl;
	}
//开始拟上三角化过程
	up_tri( a, n );
	//输出拟上三角化后的矩阵 A 到 record 文件中
	cout<<"拟上三角化后的 A 为:"<<endl;
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{

	cout<<setw(12)<<a[r][i]<<setw(12);
		}
		cout<<endl;
	}
	//记录拟上三角化后的矩阵 A 到 record 文件中
	fin<<"拟上三角化后的 A 为:"<<endl;
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{
			fin<<setw(15)<<a[r][i]<<setw(15);
		}
		fin<<endl;
	}
//开始带双步位移的 QR 分解
	Ccomplex *x;
	x = new Ccomplex[n];//x 用来存储 A 的特征值
	const int L = 100;
//	ger_a( a, n );
	QR_decomp( a, x, n, L, e );
	//显示双步位移 QR 分解后的矩阵 A 到 record 文件中
	cout<<"双步位移 QR 分解后的矩阵 A
```

```
为:"<<endl;
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{

	cout<<setw(15)<<a[r][i]<<setw(15);
		}
		cout<<endl;
	}
	//记录双步位移 QR 分解后的矩阵 A 到 record 文件中
	fin<<"双步位移 QR 分解后的矩阵 A 为:"<<endl;
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{
			fin<<setw(15)<<a[r][i]<<setw(15);
		}
		fin<<endl;
	}

	//输出矩阵的各个特征值 x
	cout<<"特征值为:"<<endl;
	for ( i = 0; i < n; ++i )
	{
		double real, imag;
		x[i].transp( real, imag );

	cout<<"λ"<<i+1<<"="<<setw(14)<<real;
		if( imag > 0 ) cout<<" + "<<imag<<endl;
		else if( imag < 0 ) cout<<" - "<<fabs(imag)<<"i"<<endl;
		else cout<<endl;
	}

	//cout<<endl;

	//记录矩阵的各个特征值 x
	fin<<"特征值为:"<<endl;
	for ( i = 0; i < n; ++i )
	{
		double real, imag;
		x[i].transp( real, imag );

	fin<<"λ"<<i+1<<"="<<setw(14)<<real;
		if( imag > 0 ) fin<<" + "<<imag<<endl;
		if( imag < 0 ) fin<<" - "<<fabs(imag)<<"i"<<endl;
		else fin<<endl;
	}

//开始求实特征值对应的解特征向量
	//求解 x 中实特征值的个数
	int m = 0;
	for ( i = 0; i < n; ++i )
	{
		double real, imag;
		x[i].transp( real, imag );
		if ( imag == 0 )
		{
			++m;
		}
	}
	//为特征向量 y 动态分配内存空间
	double **y;
	y = new double *[m];
	for ( i = 0; i < m; ++i)
	{
		y[i] = new double [n];
	}
	//输出矩阵的各个特征向量
	cout<<endl<<"各个实特征值对应的特征向量如下:"<<endl;
	eig( a, x, y, n );

	//记录矩阵的各个特征向量
	double *xx;//xx 用来存储解 x 中实特征值
	xx = new double [m];
	m = 0;
	//求解 x 中实特征值,并将其赋值给 xx
	for ( i = 0; i < n; ++i )
	{
		double real, imag;
		x[i].transp( real, imag );
```

```
		if ( imag == 0 )
		{
			xx[m] = real;
			++m;
		}
	}//此时 xx 中存储 x 中的实特征值
	fin<<endl<<"各个实特征值对应的特征向量如下:";
	for ( r = 0; r < m; ++r )
	{
		fin<<endl<<"                λ"<<r+1<<"="<<xx[r]<<" 对应的特征向量为:"<<endl;//记录特征值
		for ( i = 0; i < n; ++i )
		{
			fin<<setw(13)<<y[r][i]<<endl;// 记录特征向量
		}
	}

	fin.close();

	delete y;
	delete x;
	delete a;
}
```

//2.该函数用于求解矩阵 A 的特征向量

```
//其中 a 为待求矩阵,x 代表矩阵 a 的特征值,y 代表矩阵 a 的特征向量,n 代表矩阵 a 的维数
#include "selfdef_c.h"
#include "iostream.h"
void eig( double **a, Ccomplex *x, double **y, int n )
{
	int m = 0;//m 代表 x 中实特征值的个数
	//求解 x 中实特征值的个数
	for ( int i = 0; i < n; ++i )
	{
		double real, imag;
		x[i].transp( real, imag );
		if ( imag == 0 )
		{
			++m;//此时 m 为实特征值的个数
		}
	}

	double *xx;//xx 用来存储解 x 中实特征值
	xx = new double [m];
	m = 0;
	//求解 x 中实特征值,并将其赋值给 xx
	for ( i = 0; i < n; ++i )
	{
		double real, imag;
		x[i].transp( real, imag );
		if ( imag == 0 )
		{
			xx[m] = real;
			++m;
		}
	}
	//求解方程组(A-xx[m]*I)*y[m]=0
	for ( int r = 0; r < m; ++r )//r 为总方程组个数
	{
		ger_a( a, n );//生成矩阵 A
		for ( int i = 0; i < n; ++i )
		{
			a[i][i]  -=  xx[r];// 生 成 矩 阵 A-xx[r]*I
		}
		gauss( a, y[r], n );
		cout<<"λ "<<r+1<<"="<<xx[r]<<" 对应的特征向量为:"<<endl;//输出特征值
		for ( i = 0; i < n; ++i )
		{
			cout<<y[r][i]<<endl;//输出特征向量
		}
		cout<<endl;
	}
	delete xx;
```

//3. 选主元的 Gauss 消去法

```
#include "math.h"
#include "selfdef_c.h"
//交换的程序
void swap( double & a,double & b)
```

```
{
	double c=0.0;
	c=a;
	a=b;
	b=c;
};
//a 为系数矩阵 A,y 为解向量,n 为矩阵 A 的维数
void gauss( double **a, double *y, int n )
{
//创建向量 b
	double *b;
	b = new double [n];
	for ( int i = 0; i < n; ++i)   b[i] = 0.0;

//下面进行消元过程
	int index;
	double m;
	for(int k = 0; k != n-1; ++k)
	{
		//以下为选主元过程,即取最大值程序,得到的为最大值的下标
		index = k;		//k 用来存储主元的下标
		for(int l = k; l != n-1; ++l)
		{
			( fabs( a[l][k] ) < fabs( a[l+1][k] ) ) ? (index = l+1): (1);
		}
		//以下为交换的程序
		if ( index != k )
		{
			for( int n0 = k; n0 != n; ++n0 )
				swap( a[k][n0], a[index][n0] );
			swap( b[k], b[index] );
		}
		for(int i = k+1; i != n; ++i)	//i 代表行
		{
			m = a[i][k] / a[k][k];
			b[i] = b[i] - m * b[k];
			for(int j=k; j != n; ++j)
			{//以下注释掉的不是必须的：下界设置不好有可能会造成计算结果的错误!!!
				//	if(a[i][j] - m * a[k][j] < 1e-5)
				//		a[i][j] = 0;
				//	else
				a[i][j] = a[i][j] - m * a[k][j];
			}
		}
	}

//以下为回代过程
	y[n-1] = 1.0;
	for( int ko = n-2; ko >= 0; --ko)
	{
		double sum = 0.0;
		for( int j = ko + 1; j != n; ++j)
			sum += a[ko][j] * y[j];
		y[ko] = ( b[ko] - sum ) / a[ko][ko];
	};
	double norm = norm2( y, n );
	for ( i = 0; i < n; ++i )
	{
		y[i] /= norm;
	}
	//释放动态分配的内存空间
	delete b;
}
```

//4.生成矩阵 A

```
#include "math.h"
//其中,a 为矩阵 A,n 为矩阵 A 的维数
void ger_a( double **a, int n )
{
	int i, r;
	for ( r = 0; r < n; ++r )
	{
		for ( i = 0; i < n; ++i )
		{
//			a[r][i] = (r+1)*(i+1);
			if ( i != r )
				a[r][i] = sin(0.5*(r+1)+0.2*(i+1));
			else
				a[r][i] = 1.5*cos( r + 1 +
```

```
1.2*(i + 1) );
            }
      }
}
}
```

//5.最大值最小值

```
int max( int a, int b )
{
      return ((a > b) ? a: b);
}

int min( int a, int b )
{
      return ((a < b) ? a: b);
}
```

//6.此函数为求向量的二范数,y 为向量名,n 为向量维数

```
#include "math.h"
double norm2( double *y, int n )
{
      double sum = 0.0;
      for( int i = 0; i < n; ++i )
            sum += y[i] * y[i];
      return sqrt(sum);
}
```

//7.QR 分解子程序

a 为即将 QR 分解的矩阵 A,Q 为 QR 分解结束后的矩阵 Q,R 为 QR 分解结束后的矩阵 R

```
#include "math.h"
#include "iostream.h"
#include "iomanip.h"
void QR ( double **a, double **Q,/* double **R,*/ int n )
{
      int r, i, j;
      //为矩阵 Q 赋初始值
      for ( i = 0; i < n; ++i )
      {
            for ( j = 0; j < n; ++j )
            {
                  Q[i][j] = 0.0;
            }
            Q[i][i] = 1.0;
      }
      //开始对矩阵 A 进行 QR 分解
      for ( r = 0; r < n-1; ++r )
      {
            int flag = 0;//标记 A[i][j]是否全为 0
            for ( i = r+1; i < n; ++i )
            {
                  (a[i][r] == 0) ? flag += 0 : ++flag;
            }
            if ( flag == 0 )     continue;
            else
            {
                  //计算 d
                  double d, c, h, sum = 0;
                  for ( i = r; i < n; ++i )
                  {
                        sum += a[i][r]*a[i][r];
                  }
                  d = sqrt(sum);
                  //计算 c
                  (a[r][r] > 0) ? (c = -d) : (c = d);
                  //计算 h
                  h = c*c - c*a[r][r];

                  //计算矩阵或向量 w,p;
                  double *u, *w, *p;
                  u = new double [n];
                  p = new double [n];
                  w = new double [n];
                  //为向量 u 赋值
                  for ( i = 0; i < r; ++i )
                  {
                        u[i] = 0;
                  }
                  u[r] = a[r][r] - c;
                  for ( i = r+1; i < n; ++i )
                  {
                        u[i] = a[i][r];
                  }
                  //为向量 w 赋值
                  for ( i = 0; i < n; ++i )
                  {
```

```
			w[i] = 0;
		}
		for ( i = 0; i < n; ++i )
		{
			for ( j = r; j < n; ++j )
			{
				w[i] += Q[i][j]*u[j];
			}
		}
		//计算向量 Q
		for ( i = 0; i < n; ++i )
		{
			for ( j = r; j < n; ++j )
			{
				Q[i][j] -= w[i]*u[j]/h;
			}
		}
		//为向量 p 赋值
		for ( i = 0; i < n; ++i )
		{
			p[i] = 0;
		}
		for ( i = 0; i < n; ++i )
		{
			sum = 0;
			for ( j = r; j < n; ++j )
			{
				sum += a[j][i]*u[j];
			}
			p[i] = sum/h;
		}
		//计算矩阵 A
		for ( i = r; i < n; ++i )
		{
			for ( j = 0; j < n; ++j )
			{
				a[i][j]  -=  u[i]*p[j];//此时的 A 为 QR 分解后的上三角阵 R
				if ( fabs(a[i][j]) < 1e-12 )
				{
					a[i][j] = 0;
				}
			}
		}
			delete w;
			delete p;
			delete u;
		}
	}
}
```

//8.对拟上三角化后的矩阵 A 进行 QR 分解

```
//其中,a 代表拟上三角化后的矩阵 a,x 用来存储 A 的特征值
//n 代表矩阵 A 的维数,L 代表能够容忍的迭代次数,e 代表精度水平
#include "iostream.h"
#include "math.h"
#include "iomanip.h"
#include "selfdef_c.h"
void QR_decomp( double **a, Ccomplex *x, int n, int L, double e )
{
	int k = 0, m = n;//k 为迭代次数,m 为 A 的阶数
	double deta;//deta 为 d(A 的最后二阶子式)求根时的判别式
	int i, r;

	while ( k <= L )
	{
/////////////////////////////////第(3)步
step3:
		if ( fabs(a[m-1][m-2]) < e )//若最后一行的倒数第二个元素为 0
		{
	x[m-1].Ccomplex::Ccomplex(a[m-1][m-1], 0);
			--m;//只要 m 改变
			goto step4;//进入第(4)步
		}
		else
			goto step5;

/////////////////////////////////第(4)步
step4:
		if (    m == 2 )//若矩阵 A 只剩下两行
		{		deta		=
```

```
(a[0][0]+a[1][1])*(a[0][0]+a[1][1]) - 4*(a[0][0]*a[1][1] - a[0][1]*a[1][0]);
            if ( deta >= 0 )//若判别式>=0
            {
    x[m-1].Ccomplex::Ccomplex(( a[0][0] + a[1][1] + sqrt(deta) )/2, 0 );

    x[m-2].Ccomplex::Ccomplex(( a[0][0] + a[1][1] - sqrt(deta) )/2, 0 );
            }
            else
            {
    x[m-1].Ccomplex::Ccomplex(( a[0][0] + a[1][1] )/2, sqrt(-deta)/2 );

    x[m-2].Ccomplex::Ccomplex(( a[0][0] + a[1][1] )/2, -sqrt(-deta)/2 );
                cout<<"m = "<<m<<"时,deta小于0!!"<<endl;
            }
            goto step9;//转第(9)步
        }
        else if ( m == 1)//若矩阵A只剩下一行
        {

    x[m-1].Ccomplex::Ccomplex(a[m-1][m-1], 0);
            goto step9;//转第(9)步
        }
        else
            goto step3;//转第(3)步

/////////////////////////////////第(5)步
step5:
        if ( fabs(a[m-2][m-3]) < e )//若倒数第二行的倒数第三个元素为0
        {
            deta = (a[m-2][m-2]+a[m-1][m-1])*(a[m-2][m-2]+a[m-1][m-1]) - 4*(a[m-1][m-1]*a[m-2][m-2] - a[m-2][m-1]*a[m-1][m-2]);
            if ( deta >= 0 )//若判别式>0
            {
    x[m-1].Ccomplex::Ccomplex( ( a[m-1][m-1] + a[m-2][m-2] + sqrt(deta) )/2, 0 );

    x[m-2].Ccomplex::Ccomplex( ( a[m-1][m-1] + a[m-2][m-2] - sqrt(deta) )/2, 0 );
            }
            else
            {
    x[m-1].Ccomplex::Ccomplex( ( a[m-1][m-1] + a[m-2][m-2] )/2, sqrt(-deta)/2 );

    x[m-2].Ccomplex::Ccomplex( ( a[m-1][m-1] + a[m-2][m-2] )/2, -sqrt(-deta)/2 );
                cout<<"m = "<<m<<"时,deta小于0!!"<<endl;
            }
            m -= 2;//只要m改变,d的值就要改变
            goto step4;//转第(4)步
        }
        else
            goto step6;
/////////////////////////////////第(6)步
step6:
        if ( k != L )
            goto step7;//转第(7)步
        else
            goto step9;//转第(9)步
/////////////////////////////////第(7)步
step7://第(7)步比较麻烦
        double s, t, **M;//M为中间矩阵
        s = a[m-1][m-1] + a[m-2][m-2];
        t = a[m-2][m-2]*a[m-1][m-1] - a[m-1][m-2]*a[m-2][m-1];
        M = new double *[m];
        for ( i = 0; i < m; ++i )
        {
            M[i] = new double [m];
        }
        //计算矩阵a*a的值
        double **aa;
        aa = new double *[m];
        for ( i = 0; i < m; ++i )
        {
```

```
        aa[i] = new double [m];
    }
    //计算矩阵 a*a 即 aa 的值
    for ( i = 0; i < m; ++i)
    {
        for ( int j = 0; j < m; ++j )
        {
            aa[i][j] = 0.0;
            for ( int k0 = 0; k0 < m;
++k0 )
            {
                aa[i][j] +=
a[i][k0]*a[k0][j];
            }
            if ( fabs(aa[i][j]) < 1e-12 )
            {
                aa[i][j] = 0;
            }
        }
    }
    //为 M 矩阵赋值
    for ( i = 0; i < m; ++i)
    {
        for ( int j = 0; j < m; ++j )
        {
            (i == j) ? (M[i][j] = aa[i][j] -
s*a[i][j] + t) : (M[i][j] = aa[i][j] - s*a[i][j]);
            if ( fabs(M[i][j]) < 1e-12 )
            {
                M[i][j] = 0;
            }
        }
    }

    double **Q;
    Q = new double *[m];
    for ( r = 0; r < m; ++r )   Q[r] = new
double [m];
    QR( M, Q, m );//此时的 M 为上三角
阵就是 R,而我们只用到了 Q
    double **B;
    B = new double *[m];
    for ( r = 0; r < m; ++r )   B[r] = new
double [m];
    for ( i = 0; i < m; ++i)
    {
        for ( int j = 0; j < m; ++j )
        {
            double sum = 0.0;
            for ( int k0 = 0; k0 < m; ++k0)
            {
                sum += a[i][k0]*Q[k0][j];
            }
            B[i][j] = sum;
            if ( fabs(B[i][j]) < 1e-12 )
            {
                B[i][j] = 0;
            }
        }
    }
    for ( i = 0; i < m; ++i)
    {
        for ( int j = 0; j < m; ++j )
        {
            double sum = 0.0;
            for ( int k0 = 0; k0 < m; ++k0)
            {
                sum +=
Q[k0][i]*B[k0][j];
            }
            a[i][j] = sum;
            if ( fabs(a[i][j]) < 1e-12 )
            {
                a[i][j] = 0;
            }
        }
    }
//////////////////////////////////第(8)步
    ++k;
    delete Q;
    delete aa;
    delete M;
    goto step3;
//////////////////////////////////第(9)步
step9:
    cout<<"迭代次数 k="<<k<<endl;
    k = L + 1;
    cout<<"所有特征值均求出,计算结
```

```
束."<<endl;
	}
}
```

//9.拟上三角化过程

```
//其中,a 代表矩阵 A,n 代表矩阵 A 的维数
#include "iostream.h"
#include "math.h"
#include "selfdef_c.h"
void up_tri( double **a, int n )
{
	//定义中间使用到的矩阵,并为之分配内存单元
	double *u, *v, *p, *q, *w;
	u = new double[n];
	v = new double[n];
	p = new double[n];
	q = new double[n];
	w = new double[n];

	int r, i, flag;
	for ( r = 0; r < n-2; ++r )
	{
		//possible error
		flag = 0;//若下三角的元素均为0,flag=1;否则,flag=0,
		//判断下三角的元素是否均为 0
		i = r + 2;
		while ( (i < n) && ( fabs(a[i][r]) < 1e-12))//注意这个条件的前后顺序不能改变，变了就出错
		{
			++i;
			++flag;
		}
		if ( flag < n - r -1 )//若下三角的元素不均为 0
		{
			double d = 0, c = 0, h = 0, sum = 0;//d,c,h 分别对应笔记本上的 d,c,h
			int j, k;
			for ( j = r+1; j < n; ++j )
			{
				sum += a[j][r]*a[j][r];
			}
			d = sqrt( sum );
			( a[r+1][r] > 0) ? c = -d: c = d;
			h = c*( c - a[r+1][r] );
			//给 u 阵,v 阵赋值
			for ( j = 0; j <= r; ++j )	u[j] = 0.0, v[j] = 0.0;
			u[r+1] = a[r+1][r] - c, v[r+1] = u[r+1]/h;
			for ( j = r+2; j < n; ++j )	u[j] = a[j][r], v[j] = u[j]/h;
				//求 p 阵；p=A*u
			for ( j = 0; j < n; ++j )
			{
				sum = 0.0;
				for ( k = r+1; k < n; ++k )
				{
					sum += a[j][k]*u[k];
				}
				p[j] = sum;
			}
			//求 q 阵;q=A'*u
			for ( j = 0; j < n; ++j )
			{
				sum = 0.0;
				for ( k = r+1; k < n; ++k )
				{
					sum += a[k][j]*u[k];
				}
				q[j] = sum;
			}
			//求 t
			double t = 0.0;//t 与笔记本上意义相同
			for ( j = r+1; j < n; ++j )
			{
				t += u[j]*p[j];
			}
			//求 w
			for ( j = 0; j < n; ++j )
			{
				w[j] = q[j] - t*v[j];
			}
			//求拟上三角化后的 A 矩阵
```

```
            for ( j = 0; j < n; ++j )
            {
                for ( k = 0; k < n; ++k )
                {
                    a[j][k] = a[j][k] - p[j]*v[k] - v[j]*w[k];
                    //若 a[j][k]太小,则使之为 0
                    if ( (j > k) && ( fabs(a[j][k]) <= 1e-12 ) )    a[j][k] = 0.0;
                }
            }
        }
    }
    delete w;
    delete q;
    delete p;
    delete v;
    delete u;
}
```

//10.头文件

```
extern int max( int a,int b);
extern int min( int a,int b);
//拟上三角化过程
extern void up_tri( double **a, int n );
//生成矩阵 A
extern void ger_a( double **a, int n );
//a 为即将 QR 分解的矩阵 A,Q 为 QR 分解结束后的矩阵 Q,R 为 QR 分解结束后的矩阵 R
extern void QR ( double **a, double **Q,/* double **R,*/ int n );
//对拟上三角化后的矩阵 A 进行 QR 分解
extern void QR_decomp( double **a, Ccomplex *x, int n, int L, double e );
```

//11.自定义复数类

```
#include "iostream.h"
#include "math.h"
class Ccomplex
{
public:
    Ccomplex( double m,double n )//构造函数
    {
        Re = m;
        Im = n;
    }
    Ccomplex()//构造函数
    {
        Re = 0.0;
        Im = 0.0;
    }
    void print()//显示函数
    {
        cout<<Re;
        if( Im > 0 ) cout<<"+";
        if( Im !=0 ) cout<<Im<<"i"<<endl;
        else cout<<endl;
    }
    void transp( double &a, double &b )
    {
        a = Re;
        b = Im;
    }
private:
    double Re, Im;
};
extern int max( int a,int b);
extern int min( int a,int b);
//拟上三角化过程
extern void up_tri( double **a, int n );
//生成矩阵 A
extern void ger_a( double **a, int n );
//a 为即将 QR 分解的矩阵 A,Q 为 QR 分解结束后的矩阵 Q,R 为 QR 分解结束后的矩阵 R
extern void QR ( double **a, double **Q, int n );
//对拟上三角化后的矩阵 A 进行 QR 分解
extern void QR_decomp( double **a, Ccomplex *x, int n, int L, double e );
//交换的程序
extern void swap( double & a,double & b);
//以下为选主元的 Gauss 消去法
void gauss( double **a, double *xx, int n );
//该函数用于求解矩阵 A 的特征向量
void eig( double **a, Ccomplex *x, double **y, int n );
//此函数用于求解响亮的二范数
extern double norm2( double *y, int n );
```

三、各种求出的值

拟上三角化后的 A(5 位有效数字)：

-8.82752e-001 -9.93314e-002 -1.10335e+000 -7.60044e-001 1.54910e-001
-1.94659e+000 -8.78244e-002 -9.25589e-001 6.03260e-001 1.51886e-001
-2.34788e+000 2.37237e+000 1.81929e+000 3.23780e-001 2.20580e-001
2.10269e+000 1.81614e-001 1.27884e+000 -6.38058e-001 -4.15408e-001
0.00000e+000 1.72827e+000 -1.17147e+000 -1.24384e+000 -6.39976e-001
-2.00283e+000 2.92495e-001 -6.41283e-001 9.78400e-002 2.55776e-001
0.00000e+000 0.00000e+000 -1.29167e+000 -1.11160e+000 1.17135e+000
-1.30736e+000 1.80370e-001 -4.24639e-001 7.98896e-002 1.60882e-001
0.00000e+000 0.00000e+000 0.00000e+000 1.56013e+000 8.12505e-001
4.42176e-001 -3.58862e-002 4.69174e-001 -2.73660e-001 -7.35933e-002
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 -7.70777e-001
-1.58305e+000 -3.04284e-001 2.52871e-001 -6.70993e-001 2.54462e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
-7.46345e-001 -2.70837e-002 -9.48652e-001 1.19587e-001 1.92927e-002
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
0.00000e+000 -7.70180e-001 -4.69762e-001 4.98826e-001 1.13769e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
0.00000e+000 0.00000e+000 7.01317e-001 1.58218e-001 3.86259e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
0.00000e+000 0.00000e+000 0.00000e+000 4.84381e-001 3.99278e-001

对 $\boldsymbol{A}^{(n-1)}$进行带双步位移的 QR 分解后的 $\boldsymbol{A}$(5 位有效数字)：

3.38304e+000 8.94878e-001 -8.95676e-001 -8.46513e-002 2.61268e-001
1.61040e+000 -1.02261e+000 9.37189e-002 -1.00258e+000 -4.08626e-001
0.00000e+000 -2.11848e+000 -2.36153e+000 3.45561e-002 -4.73664e-002
1.81640e+000 -2.31898e-001 -1.43552e-001 -6.53708e-001 3.22715e-002
0.00000e+000 3.55513e-001 -2.52851e+000 6.37526e-001 2.02382e-002
1.83863e+000 1.86876e-001 -2.93258e-001 1.98707e+000 1.00463e+000
0.00000e+000 0.00000e+000 0.00000e+000 1.57755e+000 1.39596e-002
-6.97194e-001 1.55569e-001 8.40505e-003 -8.15439e-002 -1.08612e-001
0.00000e+000 0.00000e+000 0.00000e+000 -2.50506e-006 -1.48404e+000
-1.00529e-001 4.24989e-002 2.62360e-002 1.04008e-001 -1.18072e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
-6.94872e-001 -5.28921e-001 2.67916e-001 -5.96237e-001 -4.91159e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
1.78827e-001 -1.26619e+000 4.71500e-002 2.90478e-001 -3.57039e-002
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
0.00000e+000 0.00000e+000 9.35589e-001 1.87741e-001 1.36026e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
0.00000e+000 0.00000e+000 0.00000e+000 6.36051e-001 2.73703e-001
0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000 0.00000e+000
0.00000e+000 0.00000e+000 0.00000e+000 2.45761e-005 5.65165e-002

全部特征值为：

λ 1= 3.383039617436e+000
λ 2= -2.323496210212e+000 - 8.930405177200e-001i
λ 3= -2.323496210212e+000 + 8.930405177200e-001i
λ 4= -1.484039822259e+000 λ 5= 1.577548557113e+000
λ 6=-9.805309562902e-001 - 1.139489127430e-001i
λ 7= -9.805309562902e-001 + 1.139489127430e-001i
λ 8= 9.355889078188e-001 λ 9= 5.650488993501e-002
λ 10= 6.360627875745e-001

A 的各个实特征值对应的特征向量如下：

λ 1=3.383039617436e+000 的特征向量为:
-1.052215734361e-001
-2.183641973272e-001
-4.730178777759e-001
-2.608874788703e-001
-3.058056251403e-001
-2.583797832214e-001
8.733793639505e-002
4.054540338526e-001
5.090131137201e-001
2.409250445621e-001

λ 2=-1.484039822259e+000 的特征向量为:
-5.601181168003e-001
7.793415087695e-001
1.337801148572e-002
-2.774092878907e-001
3.005575883016e-003
-2.534834089601e-003
-2.062848490883e-002
-1.101348291072e-002
-1.224861665252e-002
3.236209304091e-002

λ 3=1.577548557113e+000 的特征向量为:
6.249625599173e-002
-1.120900451496e-002
-2.496667051788e-001
-1.313644861499e-001
-3.835412561202e-001
8.159443103526e-001
-1.245603520675e-001
-6.835610547208e-002
2.705874017772e-001
1.005191081460e-001

λ 4=9.355889078188e-001 的特征向量为:
8.056515482686e-002
4.611134688033e-002
-1.502437919765e-002
-4.812083461841e-002
-3.536338920690e-001
2.089190778112e-001
-1.557459663954e-001
8.196704272714e-001
-3.509825128187e-001
2.885138527827e-002

λ 5=5.650488993501e-00 的特征向量为:
-2.090175185815e-001
-2.000637966849e-001
3.891760618170e-001
-2.779391187071e-002
-3.932365479065e-001
-1.247038109520e-001
6.448109395581e-001
-3.027798531081e-001
-2.910952633140e-001
4.094365558118e-002

λ 6=6.360627875745e-00 的特征向量为:
1.070374683972e-001
7.123455832557e-002
3.902379171385e-001
-4.466555137564e-002
-7.190347442747e-001
1.758156884395e-001
-2.265379615437e-001
3.768861505021e-001
2.956254085048e-001
2.255779725592e-002