nag_hermitian_eigensystem (f02axc)

1. Purpose

nag_hermitian_eigensystem (f02axc) calculates all the eigenvalues and eigenvectors of a complex Hermitian matrix.

2. Specification

```
#include <nag.h>
#include <nagf02.h>

void nag_hermitian_eigensystem(Integer n, Complex a[], Integer tda,
            double r[], Complex v[], Integer tdv, NagError *fail)
```

3. Description

The complex Hermitian matrix A is first reduced to a real tridiagonal matrix by $n-2$ unitary transformations and a subsequent diagonal transformation. The eigenvalues and eigenvectors are then derived using the QL algorithm, an adaptation of the QR algorithm.

4. Parameters

n
Input: n, the order of the matrix A.
Constraint: $\mathbf{n} \geq 1$.

a[n][tda]
Input: the elements of the lower triangle of the n by n complex Hermitian matrix A. Elements of the array above the diagonal need not be set. See also Section 6.

tda
Input: the last dimension of the array **a** as declared in the function from which nag_hermitian_eigensystem is called.
Constraint: $\mathbf{tda} \geq \mathbf{n}$.

r[n]
Output: the eigenvalues in ascending order.

v[n][tdv]
Output: the eigenvectors, stored by columns. The ith column corresponds to the ith eigenvector. The eigenvectors are normalised so that the sum of the squares of the moduli of the elements is equal to 1 and the element of largest modulus is real. See also Section 6.

tdv
Input: the last dimension of the array **v** as declared in the function from which nag_hermitian_eigensystem is called.
Constraint: $\mathbf{tdv} \geq \mathbf{n}$.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_INT_ARG_LT
On entry, **n** must not be less than 1: $\mathbf{n} = \langle value \rangle$.

NE_2_INT_ARG_LT
On entry $\mathbf{tda} = \langle value \rangle$ while $\mathbf{n} = \langle value \rangle$. These parameters must satisfy $\mathbf{tda} \geq \mathbf{n}$.
On entry $\mathbf{tdv} = \langle value \rangle$ while $\mathbf{n} = \langle value \rangle$. These parameters must satisfy $\mathbf{tdv} \geq \mathbf{n}$.

NE_ALLOC_FAIL
Memory allocation failed.

NE_DIAG_IMAG_NON_ZERO
Matrix diagonal element **a**[⟨*value*⟩][⟨*value*⟩] has non-zero imaginary part.

NE_TOO_MANY_ITERATIONS
More than ⟨*value*⟩ iterations are required to isolate all the eigenvalues.

6. Further Comments

The time taken by the function is approximately proportional to n^3.
The function may be called with the same actual array supplied for **a** and **v**, in which case the eigenvectors will overwrite the original matrix A.

6.1. Accuracy

The eigenvectors are always accurately orthogonal but the accuracy of the individual eigenvalues and eigenvectors is dependent on their inherent sensitivity to small changes in the original matrix. For a detailed error analysis see Wilkinson and Reinsch (1971) page 235.

6.2. References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation (Vol. II, Linear Algebra)* Springer-Verlag pp 227-240.

7. See Also

None

8. Example

To calculate the eigenvalues and eigenvectors of the complex Hermitian matrix:

$$\begin{pmatrix} 0.50 & 0.00 & 1.84+1.38i & 2.08-1.56i \\ 0.00 & 0.50 & 1.12+0.84i & -0.56+0.42i \\ 1.84-1.38i & 1.12-0.84i & 0.50 & 0.00 \\ 2.08+1.56i & -0.56-0.42i & 0.00 & 0.50 \end{pmatrix}.$$

8.1. Program Text

```
/* nag_hermitian_eigensystem(f02axc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf02.h>

#define NMAX 4
#define TDA  NMAX
#define TDV NMAX

main()
{
  Integer i, j, n;
  Complex a[NMAX][TDA], v[NMAX][TDV];
  double r[NMAX];

  Vprintf("f02axc Example Program Results\n");
  Vscanf("%*[^\n]"); /* Skip heading in data file */
  Vscanf("%ld", &n);
  if (n>=1 || n<=NMAX)
    {
      for (i=0; i<n; i++)
        for (j=0; j<n; j++)
          Vscanf(" ( %lf , %lf ) ", &a[i][j].re,&a[i][j].im);
```

```
      f02axc(n, (Complex *)a, (Integer)TDA, r, (Complex *)v,
             (Integer)TDV, NAGERR_DEFAULT);
      Vprintf("Eigenvalues\n");
      for (i=0; i<n; i++)
        Vprintf("%9.4f", r[i]);
      Vprintf("\nEigenvectors\n");
      for (i=0; i<n; i++)
        for (j=0; j<n; j++)
          Vprintf("(%7.3f %7.3f )%s",v[i][j].re,v[i][j].im,
                  (j%4==3 || j==n-1) ? "\n" : " ");
      exit(EXIT_SUCCESS);
    }
  else
    {
      Vfprintf(stderr,"n is out of range: n = %4ld,\n",n);
      exit(EXIT_FAILURE);
    }
}
```

8.2. Program Data

```
f02axc Example Program Data
  4
(0.50, 0.00) ( 0.00, 0.00) (1.84, 1.38 ) ( 2.08,-1.56 )
(0.00, 0.00) ( 0.50, 0.00) (1.12, 0.84 ) (-0.56, 0.42 )
(1.84,-1.38) ( 1.12,-0.84) (0.50, 0.00 ) ( 0.00, 0.00 )
(2.08, 1.56) (-0.56,-0.42) (0.00, 0.00 ) ( 0.50, 0.00 )
```

8.3. Program Results

```
f02axc Example Program Results
Eigenvalues
  -3.0000  -1.0000   2.0000   4.0000
Eigenvectors
(  0.700   0.000 ) ( -0.100   0.000 ) ( -0.100   0.000 ) (  0.700   0.000 )
(  0.100   0.000 ) (  0.700   0.000 ) (  0.700   0.000 ) (  0.100   0.000 )
( -0.400   0.300 ) ( -0.400   0.300 ) (  0.400  -0.300 ) (  0.400  -0.300 )
( -0.400  -0.300 ) (  0.400   0.300 ) ( -0.400  -0.300 ) (  0.400   0.300 )
```