NAG Library Function Document

nag_real_eigensystem (f02agc)

1 Purpose

nag_real_eigensystem (f02agc) calculates all the eigenvalues and eigenvectors of a real unsymmetric matrix.

2 Specification

```
#include <nag.h>
#include <nagf02.h>
void nag_real_eigensystem (Integer n, double a[], Integer tda, Complex r[],
     Complex v[], Integer tdv, Integer iter[], NagError *fail)
```

3 Description

The matrix A is first balanced and then reduced to upper Hessenberg form using real stabilised elementary similarity transformations. The eigenvalues and eigenvectors of the Hessenberg matrix are calculated using the QR algorithm. The eigenvectors of the Hessenberg matrix are back-transformed to give the eigenvectors of the original matrix A.

4 References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag

5 Arguments

1: **n** – Integer *Input*

On entry: n, the order of the matrix A.

Constraint: **n** ≥ 1.

2: **a**[**n** × **tda**] – double *Input/Output*

Note: the (i,j)th element of the matrix A is stored in **a**$[(i-1) \times$ **tda** $+ j - 1]$.

On entry: the n by n matrix A.

On exit: **a** is overwritten.

3: **tda** – Integer *Input*

On entry: the stride separating matrix column elements in the array **a**.

Constraint: **tda** $\geq$ **n**.

4: **r**[**n**] – Complex *Output*

On exit: the eigenvalues.

5: **v**[**n** × **tdv**] – Complex *Output*

Note: the (i,j)th element of the matrix V is stored in **v**$[(i-1) \times$ **tdv** $+ j - 1]$.

On exit: the eigenvectors, stored by columns. The ith column corresponds to the ith eigenvalue. The eigenvectors are normalized 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. This ensures that real eigenvalues have real eigenvectors.

6: **tdv** – Integer *Input*

On entry: the stride separating matrix column elements in the array **v**.

Constraint: **tdv** $\geq$ **n**.

7: **iter**[**n**] – Integer *Output*

On exit: **iter**$[i-1]$ contains the number of iterations used to find the ith eigenvalue. If **iter**$[i-1]$ is negative, the ith eigenvalue is the second of a pair found simultaneously.

Note: the eigenvalues are found in reverse order, starting with the nth.

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **tda** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **tda** $\geq$ **n**.

On entry, **tdv** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **tdv** $\geq$ **n**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

NE_TOO_MANY_ITERATIONS

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

7 Accuracy

The accuracy of the results depends on the original matrix and the multiplicity of the roots. For a detailed error analysis see pages 352 and 390 Wilkinson and Reinsch (1971).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_real_eigensystem (f02agc) is approximately proportional to n^3.

10 Example

To calculate all the eigenvalues and eigenvectors of the real matrix

$$\begin{pmatrix} 1.5 & 0.1 & 4.5 & -1.5 \\ -22.5 & 3.5 & 12.5 & -2.5 \\ -2.5 & 0.3 & 4.5 & -2.5 \\ -2.5 & 0.1 & 4.5 & 2.5 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_real_eigensystem (f02agc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 * Mark 8 revised, 2004.
 */

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

#define COMPLEX(A) A.re, A.im
#define A(I, J)    a[(I) *tda + J]
#define V(I, J)    v[(I) *tdv + J]

int main(void)
{

  Complex  *r = 0, *v = 0;
  Integer  exit_status = 0, i, *iter = 0, j, n, tda, tdv;
  NagError fail;
  double   *a = 0;

  INIT_FAIL(fail);

  printf("nag_real_eigensystem (f02agc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &n);
#else
  scanf("%"NAG_IFMT"", &n);
#endif

  if (n >= 1)
    {
      if (!(a = NAG_ALLOC(n*n, double)) ||
          !(iter = NAG_ALLOC(n, Integer)) ||
          !(r = NAG_ALLOC(n, Complex)) ||
          !(v = NAG_ALLOC(n*n, Complex)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
      tdv = n;
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
```

```
      return exit_status;
    }
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
#ifdef _WIN32
      scanf_s("%lf", &A(i, j));
#else
      scanf("%lf", &A(i, j));
#endif
  /* nag_real_eigensystem (f02agc).
   * All eigenvalues and eigenvectors of real matrix
   */
  nag_real_eigensystem(n, a, tda, r, v, tdv, iter, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_eigensystem (f02agc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
  printf("Eigenvalues\n");
  for (i = 0; i < n; i++)
    printf("(%7.3f, %7.3f) \n", COMPLEX(r[i]));
  printf("\nEigenvectors\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      printf("(%7.3f, %7.3f) %s", COMPLEX(V(i, j)),
              (j%4 == 3 || j == n-1)?"\n":" ");
 END:
  NAG_FREE(a);
  NAG_FREE(iter);
  NAG_FREE(r);
  NAG_FREE(v);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_eigensystem (f02agc) Example Program Data
  4
   1.5   0.1   4.5  -1.5
 -22.5   3.5  12.5  -2.5
  -2.5   0.3   4.5  -2.5
  -2.5   0.1   4.5   2.5
```

10.3 Program Results

```
nag_real_eigensystem (f02agc) Example Program Results
Eigenvalues
(  3.000,   4.000)
(  3.000,  -4.000)
(  4.000,   0.000)
(  2.000,   0.000)

Eigenvectors
(  0.113,  -0.151)  (  0.113,   0.151)  ( -0.033,  -0.000)  (  0.063,   0.000)
(  0.945,   0.000)  (  0.945,   0.000)  (  0.988,   0.000)  (  0.996,   0.000)
(  0.189,   0.000)  (  0.189,  -0.000)  (  0.011,   0.000)  (  0.006,   0.000)
(  0.113,  -0.151)  (  0.113,   0.151)  (  0.154,   0.000)  (  0.063,   0.000)
```