NAG Library Function Document

nag_real_eigenvalues (f02afc)

1 Purpose

nag_real_eigenvalues (f02afc) calculates all the eigenvalues of a real unsymmetric matrix.

2 Specification

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

3 Description

The matrix A is first balanced and then reduced to upper Hessenberg form using stabilised elementary similarity transformations. The eigenvalues are then found using the QR algorithm for real Hessenberg matrices.

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** $\times$ **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: **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.

6: **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** ≥ **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 367 Wilkinson and Reinsch (1971).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_real_eigenvalues (f02afc) is approximately proportional to n^3.

10 Example

To calculate all the eigenvalues 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_eigenvalues (f02afc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 2 revised, 1992.
 * 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]
```

```
int main(void)
{
  Complex  *r = 0;
  Integer  exit_status = 0, i, *iter = 0, j, n, tda;
  NagError fail;
  double   *a = 0;

  INIT_FAIL(fail);

  printf("nag_real_eigenvalues (f02afc) 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)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = 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_eigenvalues (f02afc).
   * All eigenvalues of real matrix
   */
  nag_real_eigenvalues(n, a, tda, r, iter, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_eigenvalues (f02afc).\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]));
 END:
  NAG_FREE(a);
  NAG_FREE(iter);
  NAG_FREE(r);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_eigenvalues (f02afc) 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_eigenvalues (f02afc) Example Program Results
Eigenvalues
(   3.000 ,    4.000 )
(   3.000 ,   -4.000 )
(   4.000 ,    0.000 )
(   2.000 ,    0.000 )
```