NAG Library Function Document

nag_complex_eigensystem_sel (f02gcc)

1 Purpose

nag_complex_eigensystem_sel (f02gcc) computes selected eigenvalues and eigenvectors of a complex general matrix.

2 Specification

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

```
void nag_complex_eigensystem_sel (Nag_Select_Eigenvalues crit, Integer n,
     Complex a[], Integer tda, double wl, double wu, Integer mest,
     Integer *m, Complex w[], Complex v[], Integer tdv, NagError *fail)
```

3 Description

nag_complex_eigensystem_sel (f02gcc) computes selected eigenvalues and the corresponding right eigenvectors of a complex general matrix A:

$$Ax_i = \lambda_i x_i.$$

Eigenvalues λ_i may be selected either by *modulus*, satisfying:

$$w_l \leq |\lambda_i| \leq w_u,$$

or by *real part*, satisfying:

$$w_l \leq \mathrm{Re}(\lambda_i) \leq w_u.$$

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Arguments

1: **crit** – Nag_Select_Eigenvalues *Input*

On entry: indicates the criterion for selecting eigenvalues:

if **crit** = Nag_Select_Modulus, then eigenvalues are selected according to their moduli: $w_l \leq |\lambda_i| \leq w_u$.

if **crit** = Nag_Select_RealPart, then eigenvalues are selected according to their real parts: $w_l \leq \mathrm{Re}(\lambda_i) \leq w_u$.

Constraint: **crit** = Nag_Select_Modulus or Nag_Select_RealPart.

2: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 0$.

3: $\mathbf{a}[\mathbf{n} \times \mathbf{tda}]$ – Complex *Input/Output*

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

On entry: the n by n general matrix A.

On exit: **a** contains the Hessenberg form of the balanced input matrix A' (see Section 9).

4: **tda** – Integer *Input*

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

Constraint: $\mathbf{tda} \geq \max(1, \mathbf{n})$.

5: **wl** – double *Input*
6: **wu** – double *Input*

On entry: w_l and w_u, the lower and upper bounds on the criterion for the selected eigenvalues.

Constraint: $\mathbf{wu} > \mathbf{wl}$.

7: **mest** – Integer *Input*

On entry: **mest** must be an upper bound on m, the number of eigenvalues and eigenvectors selected. No eigenvectors are computed if $\mathbf{mest} < m$.

Constraint: $\mathbf{mest} \geq \max(1, m)$.

8: **m** – Integer * *Output*

On exit: m, the number of eigenvalues actually selected.

9: $\mathbf{w}[\mathbf{max}(\mathbf{1}, \mathbf{n})]$ – Complex *Output*

On exit: the first **m** elements of **w** hold the selected eigenvalues; elements from the index **m** to $\mathbf{n} - 1$ contain the other eigenvalues.

10: $\mathbf{v}[\mathbf{n} \times \mathbf{tdv}]$ – Complex *Output*

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

On exit: **v** contains the selected eigenvectors, with the ith column holding the eigenvector associated with the eigenvalue λ_i (stored in $\mathbf{w}[i - 1]$).

11: **tdv** – Integer *Input*

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

Constraint: $\mathbf{tdv} \geq \mathbf{mest}$.

12: **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, $\mathbf{tdv} = \langle value \rangle$ while $\mathbf{mest} = \langle value \rangle$. These arguments must satisfy $\mathbf{tdv} \geq \mathbf{mest}$.

NE_2_REAL_ARG_LE

On entry, $\mathbf{wu} = \langle value \rangle$ while $\mathbf{wl} = \langle value \rangle$. These arguments must satisfy $\mathbf{wu} > \mathbf{wl}$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument **crit** had an illegal value.

NE_EIGVEC

Inverse iteration failed to compute all the specified eigenvectors. If an eigenvector failed to converge, the corresponding column of **v** is set to zero.

NE_INT_2

On entry, **tda** $= \langle value \rangle$ while **n** $= \langle value \rangle$.
Constraint: **tda** $\geq \max(1, \mathbf{n})$.

NE_INT_ARG_LT

On entry, **mest** $= \langle value \rangle$.
Constraint: **mest** ≥ 1.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

NE_QR_FAIL

The QR algorithm failed to compute all the eigenvalues. No eigenvectors have been computed.

NE_REQD_EIGVAL

There are more than **mest** eigenvalues in the specified range. The actual number of eigenvalues in the range is returned in **m**. No eigenvectors have been computed.

Rerun with the second dimension of **v** $=$ **mest** $\geq$ **m**.

7 Accuracy

If λ_i is an exact eigenvalue, and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$\left|\tilde{\lambda}_i - \lambda_i\right| \leq \frac{c(n)\epsilon\|A'\|_2}{s_i},$$

where $c(n)$ is a modestly increasing function of n, ϵ is the ***machine precision***, and s_i is the reciprocal condition number of λ_i; A' is the balanced form of the original matrix A, and $\|A'\| \leq \|A\|$.

If x_i is the corresponding exact eigenvector, and $\tilde{x}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{x}_i, x_i)$ between them is bounded as follows:

$$\theta(\tilde{x}_i, x_i) \leq \frac{c(n)\epsilon\|A'\|_2}{sep_i}$$

where sep_i is the reciprocal condition number of x_i.

8 Parallelism and Performance

Not applicable.

9 Further Comments

nag_complex_eigensystem_sel (f02gcc) first balances the matrix, using a diagonal similarity transformation to reduce its norm; and then reduces the balanced matrix A' to upper Hessenberg form H, using a unitary similarity transformation: $A' = QHQ^H$. The function uses the Hessenberg QR algorithm to compute all the eigenvalues of H, which are the same as the eigenvalues of A. It computes the eigenvectors of H which correspond to the selected eigenvalues, using inverse iteration. It premultiplies the eigenvectors by Q to form the eigenvectors of A'; and finally transforms the eigenvectors to those of the original matrix A.

Each eigenvector x is normalized so that $\|x\|_2 = 1$, and the element of largest absolute value is real and positive.

The inverse iteration function may make a small perturbation to the real parts of close eigenvalues, and this may shift their moduli just outside the specified bounds. If you are relying on eigenvalues being within the bounds, you should test them on return from nag_complex_eigensystem_sel (f02gcc).

The time taken by the function is approximately proportional to n^3.

The function can be used to compute *all* eigenvalues and eigenvectors, by setting **wl** large and negative, and **wu** large and positive.

10 Example

To compute those eigenvalues of the matrix A whose moduli lie in the range $[-5.5, +5.5]$, and their corresponding eigenvectors, where

$$A = \begin{pmatrix} -3.97 - 5.04i & -4.11 + 3.70i & -0.34 + 1.01i & 1.29 - 0.86i \\ 0.34 - 1.50i & 1.52 - 0.43i & 1.88 - 5.38i & 3.36 + 0.65i \\ 3.31 - 3.85i & 2.50 + 3.45i & 0.88 - 1.08i & 0.64 - 1.48i \\ -1.10 + 0.82i & 1.81 - 1.59i & 3.25 + 1.33i & 1.57 - 3.44i \end{pmatrix}$$

10.1 Program Text

```
/* nag_complex_eigensystem_sel (f02gcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 * Mark 8 revised, 2004.
 *
 */

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

#define MMAX 3

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

int main(void)
{

  Complex  *a = 0, *v = 0, *w = 0;
  Integer  exit_status = 0, i, j, m, mest = MMAX, n, tda, tdv;
  NagError fail;
  double   wl, wu;

  INIT_FAIL(fail);

  printf(
          "nag_complex_eigensystem_sel (f02gcc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%lf%lf%*[^\n] ", &n, &wl, &wu);
```

```
#else
  scanf("%"NAG_IFMT"%lf%lf%*[^\n] ", &n, &wl, &wu);
#endif
  if (n >= 0)
    {
      if (!(a = NAG_ALLOC(n*n, Complex)) ||
          !(v = NAG_ALLOC(n*mest, Complex)) ||
          !(w = NAG_ALLOC(n, Complex)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
      tdv = mest;
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }

  /* Read a from data file */
  for (i = 0; i < n; ++i)
    for (j = 0; j < n; ++j)
#ifdef _WIN32
      scanf_s(" ( %lf, %lf ) ", &A(i, j).re, &A(i, j).im);
#else
      scanf(" ( %lf, %lf ) ", &A(i, j).re, &A(i, j).im);
#endif

  /* Compute selected eigenvalues and eigenvectors of A */

  /* nag_complex_eigensystem_sel (f02gcc).
   * Computes selected eigenvalues and eigenvectors of a
   * complex general matrix
   */
  nag_complex_eigensystem_sel(Nag_Select_Modulus, n, a, tda, wl, wu, mest, &m,
                              w, v, tdv, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_complex_eigensystem_sel (f02gcc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n\nEigenvalues\n\n");
  for (i = 0; i < m; ++i)
    printf("(%7.4f, %7.4f)%s", w[i].re, w[i].im, (i+1)%2 == 0?"\n":" ");

  printf("\nEigenvectors\n");
  for (i = 1; i <= m; i++)
    printf("%15"NAG_IFMT"%s", i, i%m == 0?"\n":"");

  for (i = 0; i < n; i++)
    {
      printf("%"NAG_IFMT"  ", i + 1);
      for (j = 0; j < m; j++)
        printf("(%8.4f, %8.4f)%s", V(i, j).re,
                V(i, j).im, (j + 1)%m == 0?"\n":"  ");
    }
 END:
  NAG_FREE(a);
  NAG_FREE(v);
  NAG_FREE(w);
  return exit_status;
}
```

10.2 Program Data

```
nag_complex_eigensystem_sel (f02gcc) Example Program Data
  4  -5.5  5.5                                                  :Values of n, wl, wu
 (-3.97,-5.04) (-4.11, 3.70) (-0.34, 1.01) ( 1.29,-0.86)
 ( 0.34,-1.50) ( 1.52,-0.43) ( 1.88,-5.38) ( 3.36, 0.65)
 ( 3.31,-3.85) ( 2.50, 3.45) ( 0.88,-1.08) ( 0.64,-1.48)
 (-1.10, 0.82) ( 1.81,-1.59) ( 3.25, 1.33) ( 1.57,-3.44)   :End of matrix a
```

10.3 Program Results

```
nag_complex_eigensystem_sel (f02gcc) Example Program Results


Eigenvalues

(-5.0000,  2.0060) ( 3.0023, -3.9998)

Eigenvectors
              1              2
1  ( -0.3865,   0.1732)  ( -0.0356,  -0.1782)
2  ( -0.3539,   0.4529)  (  0.1264,   0.2666)
3  (  0.6124,   0.0000)  (  0.0129,  -0.2966)
4  ( -0.0859,  -0.3284)  (  0.8898,   0.0000)
```