

# NAG Library Function Document

## nag\_real\_eigensystem\_sel (f02ecc)

### 1 Purpose

nag\_real\_eigensystem\_sel (f02ecc) computes selected eigenvalues and eigenvectors of a real general matrix.

### 2 Specification

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

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

### 3 Description

nag\_real\_eigensystem\_sel (f02ecc) computes selected eigenvalues and the corresponding right eigenvectors of a real general matrix  $A$ :

$$Ax_i = \lambda_i x_i.$$

Eigenvalues  $\lambda_i$  may be selected either by *modulus*, satisfying:

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

or by *real part*, satisfying:

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

Note that even though  $A$  is real,  $\lambda_i$  and  $x_i$  may be complex. If  $x_i$  is an eigenvector corresponding to a complex eigenvalue  $\lambda_i$ , then the complex conjugate vector  $\bar{x}_i$  is the eigenvector corresponding to the complex conjugate eigenvalue  $\bar{\lambda}_i$ . The eigenvalues in a complex conjugate pair  $\lambda_i$  and  $\bar{\lambda}_i$  are either both selected or both not selected.

### 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:

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

*Constraint:* **crit** = Nag\_Select\_Modulus or Nag\_Select\_RealPart.

- 2: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $n \geq 0$ .
- 3: **a**[ $n \times tda$ ] – double *Input/Output*  
**Note:** the  $(i, j)$ th element of the matrix  $A$  is stored in  $\mathbf{a}[(i - 1) \times tda + j - 1]$ .  
*On entry:* the  $n$  by  $n$  general matrix  $A$ .  
*On exit:*  $\mathbf{a}$  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  $\mathbf{a}$ .  
*Constraint:*  $tda \geq \max(1, n)$ .
- 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:*  $wu > wl$ .
- 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 **mest**  $< m$ .  
*Constraint:* **mest**  $\geq \max(1, m)$ .
- 8: **m** – Integer \* *Output*  
*On exit:*  $m$ , the number of eigenvalues actually selected.
- 9: **w**[ $\max(1, n)$ ] – Complex *Output*  
*On exit:* the first **m** elements of  $\mathbf{w}$  hold the values of the selected eigenvalues; elements from the index **m** to  $n - 1$  contain the other eigenvalues. Complex conjugate pairs of eigenvalues are stored in consecutive elements of the array, with the eigenvalue having the positive imaginary part first.
- 10: **v**[ $n \times tdv$ ] – Complex *Output*  
**Note:** the  $(i, j)$ th element of the matrix  $V$  is stored in  $\mathbf{v}[(i - 1) \times tdv + j - 1]$ .  
*On exit:*  $\mathbf{v}$  contains the selected eigenvectors, with the  $i$ th column holding the eigenvector associated with the eigenvalue  $\lambda_i$  (stored in  $\mathbf{w}[i - 1]$ ).
- 11: **tdv** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array  $\mathbf{v}$ .  
*Constraint:* **tdv**  $\geq mest$ .
- 12: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_2\_REAL\_ARG\_LE

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

**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**  $\geq$   $\max(1, \mathbf{n})$ .

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

**NE\_INT\_ARG\_LT**

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

On entry, **n** =  $\langle value \rangle$ .  
Constraint: **n**  $\geq$  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  $\lambda_i$  is an exact eigenvalue, and  $\tilde{\lambda}_i$  is the corresponding computed value, then

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

where  $c(n)$  is a modestly increasing function of  $n$ ,  $\epsilon$  is the *machine precision*, and  $s_i$  is the reciprocal condition number of  $\lambda_i$ ;  $A'$  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\_real\_eigensystem\_sel (f02ecc) 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 an orthogonal similarity transformation:  $A' = QHQ^T$ . 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  $x$  (real or complex) 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\_real\_eigensystem\_sel (f02ecc).

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  $[0.2, 0.5]$ , and their corresponding eigenvectors, where

$$A = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix}$$

### 10.1 Program Text

```
/* nag_real_eigensystem_sel (f02ecc) 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 *v = 0, *w = 0;
    Integer exit_status = 0, i, j, m, mest, n, tda, tdv;
    double *a = 0, wl, wu;

    NagError fail;

    INIT_FAIL(fail);

    printf(
        "nag_real_eigensystem_sel (f02ecc) Example Program Results\n\n");

    /* Skip heading in data file */
#ifdef _WIN32
```

```

scanf_s("%*[\n]");
#else
scanf("%*[\n]");
#endif
mest = MMAX;
#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, double)) ||
!(w = NAG_ALLOC(n, Complex)) ||
!(v = NAG_ALLOC(n*mest, 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", &A(i, j));
#else
scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
scanf_s("%*[\n]");
#else
scanf("%*[\n]");
#endif

/* Compute selected eigenvalues and eigenvectors of A */

/* nag_real_eigensystem_sel (f02ecc).
* Computes selected eigenvalues and eigenvectors of a real
* general matrix
*/
nag_real_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_real_eigensystem_sel (f02ecc).\n%s\n",
fail.message);
exit_status = 1;
goto END;
}

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

printf(" Eigenvectors\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(w);
NAG_FREE(v);
return exit_status;
}

```

## 10.2 Program Data

```

nag_real_eigensystem_sel (f02ecc) Example Program Data
 4  0.2  0.5           :Values of n, wl, wu
 0.35  0.45 -0.14 -0.17
 0.09  0.07 -0.54  0.35
-0.44 -0.33 -0.03  0.17
 0.25 -0.32 -0.13  0.11 :End of matrix a

```

## 10.3 Program Results

```

nag_real_eigensystem_sel (f02ecc) Example Program Results

```

```

Eigenvalues
(-0.0994, 0.4008) (-0.0994, -0.4008)
Eigenvectors
      1          2
1 ( -0.1933,  0.2546) ( -0.1933, -0.2546)
2 (  0.2519, -0.5224) (  0.2519,  0.5224)
3 (  0.0972, -0.3084) (  0.0972,  0.3084)
4 (  0.6760,  0.0000) (  0.6760, -0.0000)

```

---