NAG Toolbox for MATLAB

f08nv

1 Purpose

f08nv balances a complex general matrix in order to improve the accuracy of computed eigenvalues and/or eigenvectors.

2 Syntax

```
[a, ilo, ihi, scale, info] = f08nv(job, a, 'n', n)
```

3 Description

f08nv balances a complex general matrix A. The term ‘balancing’ covers two steps, each of which involves a similarity transformation of A. The function can perform either or both of these steps.

1. The function first attempts to permute A to block upper triangular form by a similarity transformation:

$$PAP^{\mathrm{T}} = A' = \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A'_{33} \end{pmatrix}$$

   where P is a permutation matrix, and A'_{11} and A'_{33} are upper triangular. Then the diagonal elements of A'_{11} and A'_{33} are eigenvalues of A. The rest of the eigenvalues of A are the eigenvalues of the central diagonal block A'_{22}, in rows and columns i_{lo} to i_{hi}. Subsequent operations to compute the eigenvalues of A (or its Schur factorization) need only be applied to these rows and columns; this can save a significant amount of work if $i_{\mathrm{lo}} > 1$ and $i_{\mathrm{hi}} < n$. If no suitable permutation exists (as is often the case), the function sets $i_{\mathrm{lo}} = 1$ and $i_{\mathrm{hi}} = n$, and A'_{22} is the whole of A.

2. The function applies a diagonal similarity transformation to A', to make the rows and columns of A'_{22} as close in norm as possible:

$$A'' = DA'D^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A'_{33} \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22}^{-1} & 0 \\ 0 & 0 & I \end{pmatrix}.$$

   This scaling can reduce the norm of the matrix (that is, $\|A''_{22}\| < \|A'_{22}\|$) and hence reduce the effect of rounding errors on the accuracy of computed eigenvalues and eigenvectors.

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **job – string**

Indicates whether A is to be permuted and/or scaled (or neither).

job = 'N'

A is neither permuted nor scaled (but values are assigned to **ilo**, **ihi** and **scale**).

job = 'P'

A is permuted but not scaled.

job = 'S'

A is scaled but not permuted.

job = 'B'

A is both permuted and scaled.

Constraint: **job** = 'N', 'P', 'S' or 'B'.

2: **a(lda,∗) – complex array**

The first dimension of the array **a** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

The n by n matrix A.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The second dimension of the array **a**.

n, the order of the matrix A.

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

5.3 Input Parameters Omitted from the MATLAB Interface

lda

5.4 Output Parameters

1: **a(lda,∗) – complex array**

The first dimension of the array **a** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

a contains the balanced matrix. If **job** = 'N', **a** is not referenced.

2: **ilo – int32 scalar**
3: **ihi – int32 scalar**

The values i_{lo} and i_{hi} such that on exit $\mathbf{a}(i,j)$ is zero if $i > j$ and $1 \leq j < i_{\mathrm{lo}}$ or $i_{\mathrm{hi}} < i \leq n$.

If **job** = 'N' or 'S', $i_{\mathrm{lo}} = 1$ and $i_{\mathrm{hi}} = n$.

4: **scale(∗) – double array**

Note: the dimension of the array **scale** must be at least $\max(1, \mathbf{n})$.

Details of the permutations and scaling factors applied to A. More precisely, if p_j is the index of the row and column interchanged with row and column j and d_j is the scaling factor used to balance row and column j then

$$\textbf{scale}(j) = \begin{cases} p_j, & j = 1, 2, \ldots, i_{\mathrm{lo}} - 1 \\ d_j, & j = i_{\mathrm{lo}}, i_{\mathrm{lo}} + 1, \ldots, i_{\mathrm{hi}} \quad \text{and} \\ p_j, & j = i_{\mathrm{hi}} + 1, i_{\mathrm{hi}} + 2, \ldots, n. \end{cases}$$

The order in which the interchanges are made is n to $i_{\mathrm{hi}} + 1$ then 1 to $i_{\mathrm{lo}} - 1$.

5: **info – int32 scalar**

info $= 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

info $= -i$

If **info** $= -i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **job**, 2: **n**, 3: **a**, 4: **lda**, 5: **ilo**, 6: **ihi**, 7: **scale**, 8: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

7 Accuracy

The errors are negligible, compared with those in subsequent computations.

8 Further Comments

If the matrix A is balanced by f08nv, then any eigenvectors computed subsequently are eigenvectors of the matrix A'' (see Section 3) and hence f08nw **must** then be called to transform them back to eigenvectors of A.

If the Schur vectors of A are required, then this function must **not** be called with **job** = 'S' or 'B', because then the balancing transformation is not unitary. If this function is called with **job** = 'P', then any Schur vectors computed subsequently are Schur vectors of the matrix A'', and f08nw **must** be called (with **side** = 'R') to transform them back to Schur vectors of A.

The total number of real floating-point operations is approximately proportional to n^2.

The real analogue of this function is f08nh.

9 Example

```
job = 'Both';
a = [complex(1.5, -2.75), complex(0, +0), complex(0, +0), complex(0, +0);
          complex(-8.06,  -1.24),  complex(-2.5,  -0.5),  complex(0,  +0),
complex(-0.75, +0.5);
      complex(-2.09, +7.56), complex(1.39, +3.97), complex(-1.25, +0.75),
complex(-4.82, -5.67);
          complex(6.18,  +9.79),  complex(-0.92,  -0.62),  complex(0,  +0),
complex(-2.5, -0.5)];
[aOut, ilo, ihi, scale, info] = f08nv(job, a)
```

```
aOut =
   -1.2500 + 0.7500i    1.3900 + 3.9700i   -4.8200 - 5.6700i   -2.0900 +
7.5600i
         0             -2.5000 - 0.5000i  -0.7500 + 0.5000i   -8.0600 -
1.2400i
```

```
          0              -0.9200 - 0.6200i   -2.5000 - 0.5000i    6.1800 +
9.7900i
         0                    0                    0              1.5000 -
2.7500i
ilo =
           2
ihi =
           3
scale =
     3
     1
     1
     1
info =
           0
```