NAG Toolbox

nag_lapack_zppequ (f07gt)

1 Purpose

nag_lapack_zppequ (f07gt) computes a diagonal scaling matrix S intended to equilibrate a complex n by n Hermitian positive definite matrix A, stored in packed format, and reduce its condition number.

2 Syntax

```
[s, scond, amax, info] = nag_lapack_zppequ(uplo, n, ap)
[s, scond, amax, info] = f07gt(uplo, n, ap)
```

3 Description

nag_lapack_zppequ (f07gt) computes a diagonal scaling matrix S chosen so that

$$s_j = 1/\sqrt{a_{jj}}.$$

This means that the matrix B given by

$$B = SAS,$$

has diagonal elements equal to unity. This in turn means that the condition number of B, $\kappa_2(B)$, is within a factor n of the matrix of smallest possible condition number over all possible choices of diagonal scalings (see Corollary 7.6 of Higham (2002)).

4 References

Higham N J (2002) *Accuracy and Stability of Numerical Algorithms* (2nd Edition) SIAM, Philadelphia

5 Parameters

5.1 Compulsory Input Parameters

1: **uplo** – CHARACTER(1)

Indicates whether the upper or lower triangular part of A is stored in the array **ap**, as follows:

uplo = 'U'
The upper triangle of A is stored.

uplo = 'L'
The lower triangle of A is stored.

Constraint: **uplo** = 'U' or 'L'.

2: **n** – INTEGER

n, the order of the matrix A.

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

3: **ap**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **ap** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$

The n by n Hermitian matrix A, packed by columns.

More precisely,

if **uplo** $=$ 'U', the upper triangle of A must be stored with element A_{ij} in **ap**$(i + j(j-1)/2)$ for $i \leq j$;

if **uplo** $=$ 'L', the lower triangle of A must be stored with element A_{ij} in **ap**$(i + (2n-j)(j-1)/2)$ for $i \geq j$.

Only the elements of **ap** corresponding to the diagonal elements A are referenced.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **s**(**n**) – REAL (KIND=nag_wp) array

If **info** $= 0$, **s** contains the diagonal elements of the scaling matrix S.

2: **scond** – REAL (KIND=nag_wp)

If **info** $= 0$, **scond** contains the ratio of the smallest value of **s** to the largest value of **s**. If **scond** ≥ 0.1 and **amax** is neither too large nor too small, it is not worth scaling by S.

3: **amax** – REAL (KIND=nag_wp)

$\max|a_{ij}|$. If **amax** is very close to overflow or underflow, the matrix A should be scaled.

4: **info** – INTEGER

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

6 Error Indicators and Warnings

info < 0

If **info** $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

info > 0

The $\langle$*value*$\rangle$th diagonal element of A is not positive (and hence A cannot be positive definite).

7 Accuracy

The computed scale factors will be close to the exact scale factors.

8 Further Comments

The real analogue of this function is nag_lapack_dppequ (f07gf).

9 Example

This example equilibrates the Hermitian positive definite matrix A given by

$$A = \begin{pmatrix} 3.23 & 1.51 - 1.92i & (1.90 + 0.84i) \times 10^5 & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 & (-0.23 + 1.11i) \times 10^5 & -1.18 + 1.37i \\ (1.90 - 0.84i) \times 10^5 & (-0.23 - 1.11i) \times 10^5 & 4.09 \times 10^{10} & (2.33 - 0.14i) \times 10^5 \\ 0.42 - 2.50i & -1.18 - 1.37i & (2.33 + 0.14i) \times 10^5 & 4.29 \end{pmatrix}.$$

Details of the scaling factors and the scaled matrix are output.

9.1 Program Text

```
function f07gt_example

fprintf('f07gt example results\n\n');

% Upper triangular part of Hermitian matrix A
uplo = 'Upper';
n = nag_int(4);
ap = [ 3.23   + 0i,       ...
       1.51   - 1.92i,    3.58   + 0i,       ...
       1.90e5 + 0.84e5i, -0.23e5 + 1.11e5i, 4.09e10 + 0i,       ...
       0.42   + 2.50i,   -1.18   + 1.37i,   2.33e5  - 0.14e5i,  4.29 + 0i];

% Scale A
[s, scond, amax, info] = f07gt( ...
                                uplo, n, ap);

fprintf('scond = %8.1e, amax = %8.1e\n\n', scond, amax);
disp('Diagonal scaling factors');
fprintf('%10.1e',s);
fprintf('\n\n');

% Apply scalings
k = 0;
for i = 1:n
  for j = 1:i
    k = k + 1;
    asp(k) = s(i)*ap(k)*s(j);
  end
end

[ifail] = x04dc( ...
                 'Upper', 'Non-unit', n, asp, 'Scaled matrix');
```

9.2 Program Results

```
f07gt example results

scond =  8.9e-06, amax =  4.1e+10

Diagonal scaling factors
   5.6e-01   5.3e-01   4.9e-06   4.8e-01

 Scaled matrix
          1       2       3       4
 1   1.0000  0.4441  0.5227  0.1128
     0.0000 -0.5646  0.2311  0.6716

 2           1.0000 -0.0601 -0.3011
             0.0000  0.2901  0.3496

 3                   1.0000  0.5562
                     0.0000 -0.0334

 4                           1.0000
                             0.0000
```