NAG Toolbox

nag_lapack_zgeqpf (f08bs)

1 Purpose

nag_lapack_zgeqpf (f08bs) computes the QR factorization, with column pivoting, of a complex m by n matrix.

2 Syntax

```
[a, jpvt, tau, info] = nag_lapack_zgeqpf(a, jpvt, 'm', m, 'n', n)
[a, jpvt, tau, info] = f08bs(a, jpvt, 'm', m, 'n', n)
```

3 Description

nag_lapack_zgeqpf (f08bs) forms the QR factorization, with column pivoting, of an arbitrary rectangular complex m by n matrix.

If $m \geq n$, the factorization is given by:

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where R is an n by n upper triangular matrix (with real diagonal elements), Q is an m by m unitary matrix and P is an n by n permutation matrix. It is sometimes more convenient to write the factorization as

$$AP = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}\begin{pmatrix} R \\ 0 \end{pmatrix},$$

which reduces to

$$AP = Q_1R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m - n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$AP = Q\begin{pmatrix} R_1 & R_2 \end{pmatrix},$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the F08 Chapter Introduction for details). Functions are provided to work with Q in this representation (see Section 9).

Note also that for any $k < n$, the information returned in the first k columns of the array **a** represents a QR factorization of the first k columns of the permuted matrix AP.

The function allows specified columns of A to be moved to the leading columns of AP at the start of the factorization and fixed there. The remaining columns are free to be interchanged so that at the ith stage the pivot column is chosen to be the column which maximizes the 2-norm of elements i to m over columns i to n.

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: **a**(lda, :) – COMPLEX (KIND=nag_wp) array

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

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

The m by n matrix A.

2: **jpvt**(:) – INTEGER array

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

If $\mathbf{jpvt}(i) \neq 0$, then the i th column of A is moved to the beginning of AP before the decomposition is computed and is fixed in place during the computation. Otherwise, the i th column of A is a free column (i.e., one which may be interchanged during the computation with any other free column).

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the first dimension of the array **a**.

m, the number of rows of the matrix A.

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

2: **n** – INTEGER

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

n, the number of columns of the matrix A.

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

5.3 Output Parameters

1: **a**(lda, :) – COMPLEX (KIND=nag_wp) array

The first dimension of the array **a** will be $\max(1, \mathbf{m})$.

The second dimension of the array **a** will be $\max(1, \mathbf{n})$.

If $m \geq n$, the elements below the diagonal store details of the unitary matrix Q and the upper triangle stores the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part stores details of the unitary matrix Q and the remaining elements store the corresponding elements of the m by n upper trapezoidal matrix R.

The diagonal elements of R are real.

2: **jpvt**(:) – INTEGER array

The dimension of the array **jpvt** will be $\max(1, \mathbf{n})$

Details of the permutation matrix P. More precisely, if $\mathbf{jpvt}(i) = k$, then the kth column of A is moved to become the i th column of AP; in other words, the columns of AP are the columns of A in the order $\mathbf{jpvt}(1), \mathbf{jpvt}(2), \ldots, \mathbf{jpvt}(n)$.

3: **tau**(**min**(**m**, **n**)) – COMPLEX (KIND=nag_wp) array

Further details of the unitary matrix Q.

4: **info** – INTEGER

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

6 Error Indicators and Warnings

info $= -i$

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

1: **m**, 2: **n**, 3: **a**, 4: **lda**, 5: **jpvt**, 6: **tau**, 7: **work**, 8: **rwork**, 9: **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 computed factorization is the exact factorization of a nearby matrix $(A+E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{8}{3}m^2(3n-m)$ if $m < n$.

To form the unitary matrix Q nag_lapack_zgeqpf (f08bs) may be followed by a call to nag_lapack_zungqr (f08at):

```
[a, info] = f08at(a(:,1:m), tau);
```

but note that the second dimension of the array **a** must be at least **m**, which may be larger than was required by nag_lapack_zgeqpf (f08bs).

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
[a, info] = f08at(a, tau);
```

To apply Q to an arbitrary complex rectangular matrix C, nag_lapack_zgeqpf (f08bs) may be followed by a call to nag_lapack_zunmqr (f08au). For example,

```
[c, info] = f08au('Left','Conjugate Transpose', a(:,min(m,n)), tau, c);
```

forms $C = Q^{\mathrm{H}}C$, where C is m by p.

To compute a QR factorization without column pivoting, use nag_lapack_zgeqrf (f08as).

The real analogue of this function is nag_lapack_dgeqpf (f08be).

9 Example

This example solves the linear least squares problems

$$\text{minimize } \|Ax_i - b_i\|_2, \quad i = 1, 2$$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} 0.47-0.34i & -0.40+0.54i & 0.60+0.01i & 0.80-1.02i \\ -0.32-0.23i & -0.05+0.20i & -0.26-0.44i & -0.43+0.17i \\ 0.35-0.60i & -0.52-0.34i & 0.87-0.11i & -0.34-0.09i \\ 0.89+0.71i & -0.45-0.45i & -0.02-0.57i & 1.14-0.78i \\ -0.19+0.06i & 0.11-0.85i & 1.44+0.80i & 0.07+1.14i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -0.85 - 1.63i & 2.49 + 4.01i \\ -2.16 + 3.52i & -0.14 + 7.98i \\ 4.57 - 5.71i & 8.36 - 0.28i \\ 6.38 - 7.40i & -3.55 + 1.29i \\ 8.41 + 9.39i & -6.72 + 5.03i \end{pmatrix}.$$

Here A is approximately rank-deficient, and hence it is preferable to use nag_lapack_zgeqpf (f08bs) rather than nag_lapack_zgeqrf (f08as).

9.1 Program Text

```
     function f08bs_example

fprintf('f08bs example results\n\n');

a = [ 0.47 - 0.34i, -0.40 + 0.54i,  0.60 + 0.01i,  0.80 - 1.02i;
     -0.32 - 0.23i, -0.05 + 0.20i, -0.26 - 0.44i, -0.43 + 0.17i;
      0.35 - 0.60i, -0.52 - 0.34i,  0.87 - 0.11i, -0.34 - 0.09i;
      0.89 + 0.71i, -0.45 - 0.45i, -0.02 - 0.57i,  1.14 - 0.78i;
     -0.19 + 0.06i,  0.11 - 0.85i,  1.44 + 0.80i,  0.07 + 1.14i];
b = [-0.85 - 1.63i,  2.49 + 4.01i;
     -2.16 + 3.52i, -0.14 + 7.98i;
      4.57 - 5.71i,  8.36 - 0.28i;
      6.38 - 7.40i, -3.55 + 1.29i;
      8.41 + 9.39i, -6.72 + 5.03i];
[m,n] = size(a);
jpvt = zeros(n,1,nag_int_name);

% Compute the QR factorization of a
[a, jpvt, tau, info] = f08bs( ...
       a, jpvt);

% Choose tol to reflect the relative accuracy of the input data
tol = 0.01;

% Determine which columns of R to use
k = find(abs(diag(a)) <= tol*abs(a(1,1)));
if numel(k) == 0
  k = numel(diag(a));
else
  k = k(1)-1;
end

% Compute c = (q^H)*b,
[c, info] = f08au( ...
    'Left', 'Conjugate Transpose', a, tau, b);

% Compute least-squares solution by backsubstitution in r*b = c
c(1:k, :) = inv(triu(a(1:k,1:k)))*c(1:k,:);
c(k+1:4, :) = 0;

% Unscramble the least-squares solution stored in c
x = zeros(4, 2);
for i=1:4
  x(jpvt(i), :) = c(i, :);
end

fprintf('\nLeast-squares solution\n');
disp(x);
```

9.2 Program Results

```
     f08bs example results

Least-squares solution
   0.0000 + 0.0000i   0.0000 + 0.0000i
   2.6925 + 8.0446i  -2.0563 - 2.9759i
   2.7602 + 2.5455i   1.0588 + 1.4635i
   2.7383 + 0.5123i  -1.4150 + 0.2982i
```