NAG C Library Function Document

nag_zungqr (f08atc)

1 Purpose

nag_zungqr (f08atc) generates all or part of the complex unitary matrix Q from a QR factorization computed by nag_zgeqrf (f08asc) or nag_zgeqpf (f08bsc).

2 Specification

```
void nag_zungqr (Nag_OrderType order, Integer m, Integer n, Integer k, Complex a[],
     Integer pda, const Complex tau[], NagError *fail)
```

3 Description

nag_zungqr (f08atc) is intended to be used after a call to nag_zgeqrf (f08asc) or nag_zgeqpf (f08bsc), which perform a QR factorization of a complex matrix A. The unitary matrix Q is represented as a product of elementary reflectors.

This function may be used to generate Q explicitly as a square matrix, or to form only its leading columns.

Usually Q is determined from the QR factorization of an m by p matrix A with $m \geq p$. The whole of Q may be computed by:

```
nag_zungqr (order,m,m,p,&a,pda,tau,&fail)
```

(note that the array **a** must have at least m columns) or its leading p columns by:

```
nag_zungqr (order,m,p,p,&a,pda,tau,&fail)
```

The columns of Q returned by the last call form an orthonormal basis for the space spanned by the columns of A; thus nag_zgeqrf (f08asc) followed by nag_zungqr (f08atc) can be used to orthogonalise the columns of A.

The information returned by the QR factorization functions also yields the QR factorization of the leading k columns of A, where $k < p$. The unitary matrix arising from this factorization can be computed by:

```
nag_zungqr (order,m,m,k,&a,pda,tau,&fail)
```

or its leading k columns by:

```
nag_zungqr (order,m,k,k,&a,pda,tau,&fail)
```

4 References

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

5 Parameters

1: **order** – Nag_OrderType *Input*

On entry: the **order** parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = **Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.

2: **m** – Integer *Input*

On entry: m, the order of the unitary matrix Q.

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

3: **n** – Integer *Input*

On entry: n, the number of columns of matrix Q that are required.

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

4: **k** – Integer *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

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

5: **a**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **a** must be at least $\max(1, \mathbf{pda} \times \mathbf{n})$ when **order** = **Nag_ColMajor** and at least $\max(1, \mathbf{pda} \times \mathbf{m})$ when **order** = **Nag_RowMajor**.

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix A is stored in **a**$[(j-1) \times \mathbf{pda} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix A is stored in **a**$[(i-1) \times \mathbf{pda} + j - 1]$.

On entry: details of the vectors which define the elementary reflectors, as returned by nag_zgeqrf (f08asc) or nag_zgeqpf (f08bsc).

On exit: the m by n matrix Q.

6: **pda** – Integer *Input*

On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **a**.

Constraints:

if **order** = **Nag_ColMajor**, **pda** $\geq \max(1, \mathbf{m})$;
if **order** = **Nag_RowMajor**, **pda** $\geq \max(1, \mathbf{n})$.

7: **tau**[dim] – const Complex *Input*

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

On entry: further details of the elementary reflectors, as returned by nag_zgeqrf (f08asc) or nag_zgeqpf (f08bsc).

8: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **m** = ⟨*value*⟩.
Constraint: $\mathbf{m} \geq 0$.

On entry, **pda** = ⟨*value*⟩.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, **m** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: $\mathbf{m} \geq \mathbf{n} \geq 0$.

On entry, **n** = ⟨*value*⟩, **k** = ⟨*value*⟩.
Constraint: **n** ≥ **k** ≥ 0.

On entry, **pda** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **pda** ≥ max(1, **m**).

On entry, **pda** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pda** ≥ max(1, **n**).

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ had an illegal value.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

7 Accuracy

The computed matrix Q differs from an exactly unitary matrix by a matrix E such that

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

where ϵ is the ***machine precision***.

8 Further Comments

The total number of real floating-point operations is approximately $16mnk - 8(m+n)k^2 + \frac{16}{3}k^3$; when $n = k$, the number is approximately $\frac{8}{3}n^2(3m-n)$.

The real analogue of this function is nag_dorgqr (f08afc).

9 Example

To form the leading 4 columns of the unitary matrix Q from the QR factorization of the matrix A, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}.$$

The columns of Q form an orthonormal basis for the space spanned by the columns of A.

9.1 Program Text

```
/* nag_zungqr (f08atc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>
```

```
int main(void)
{

  /* Scalars */
  Integer  i, j, m, n, pda, tau_len;
  Integer  exit_status=0;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  char    *title=0;
  Complex *a=0, *tau=0;

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
  order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
  Vprintf("f08atc Example Program Results\n\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%ld%*[^\n] ", &m, &n);
#ifdef NAG_COLUMN_MAJOR
  pda = m;
#else
  pda = n;
#endif
  tau_len = MIN(m,n);

  /* Allocate memory */
  if ( !(title = NAG_ALLOC(31, char)) ||
       !(a = NAG_ALLOC(m * n, Complex)) ||
       !(tau = NAG_ALLOC(tau_len, Complex))  )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A from data file */
  for (i = 1; i <= m; ++i)
    {
      for (j = 1; j <= n; ++j)
        Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
  Vscanf("%*[^\n] ");

  /* Compute the QR factorization of A */
  f08asc(order, m, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08asc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Form the leading N columns of Q explicitly */
  f08atc(order, m, n, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08atc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print the leading N columns of Q only */
```

```
  Vsprintf(title, "The leading %2ld columns of Q\n", n);
  x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n, a, pda,
         Nag_BracketForm, "%7.4f", title, Nag_IntegerLabels, 0,
         Nag_IntegerLabels, 0, 80, 0, 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from x04dbc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  if (title) NAG_FREE(title);
  if (a) NAG_FREE(a);
  if (tau) NAG_FREE(tau);

  return exit_status;
}
```

9.2 Program Data

```
f08atc Example Program Data
  6  4                                                  :Values of M and N
 ( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
 (-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
 ( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
 (-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
 ( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
 ( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26)   :End of matrix A
```

9.3 Program Results

```
f08atc Example Program Results

 The leading  4 columns of Q

                    1                   2                   3                   4
 1  (-0.3110, 0.2624)  (-0.3175, 0.4835)  ( 0.4966,-0.2997)  (-0.0072,-0.3718)
 2  ( 0.3175,-0.6414)  (-0.2062, 0.1577)  (-0.0793,-0.3094)  (-0.0282,-0.1491)
 3  (-0.2008, 0.1490)  ( 0.4892,-0.0900)  ( 0.0357,-0.0219)  ( 0.5625,-0.0710)
 4  ( 0.1199,-0.1231)  ( 0.2566,-0.3055)  ( 0.4489,-0.2141)  (-0.1651, 0.1800)
 5  (-0.2689,-0.1652)  ( 0.1697,-0.2491)  (-0.0496, 0.1158)  (-0.4885,-0.4540)
 6  (-0.3499, 0.0907)  (-0.0491,-0.3133)  (-0.1256,-0.5300)  ( 0.1039, 0.0450)
```