

# NAG Library Function Document

## nag\_zgbbrd (f08lsc)

### 1 Purpose

nag\_zgbbrd (f08lsc) reduces a complex  $m$  by  $n$  band matrix to real upper bidiagonal form.

### 2 Specification

```
#include <nag.h>
#include <nagf08.h>

void nag_zgbbrd (Nag_OrderType order, Nag_VectType vect, Integer m,
                Integer n, Integer ncc, Integer kl, Integer ku, Complex ab[],
                Integer pdab, double d[], double e[], Complex q[], Integer pdq,
                Complex pt[], Integer pdpt, Complex c[], Integer pdc, NagError *fail)
```

### 3 Description

nag\_zgbbrd (f08lsc) reduces a complex  $m$  by  $n$  band matrix to real upper bidiagonal form  $B$  by a unitary transformation:  $A = QBP^H$ . The unitary matrices  $Q$  and  $P^H$ , of order  $m$  and  $n$  respectively, are determined as a product of Givens rotation matrices, and may be formed explicitly by the function if required. A matrix  $C$  may also be updated to give  $\tilde{C} = Q^H C$ .

The function uses a vectorizable form of the reduction.

### 4 References

None.

### 5 Arguments

- 1: **order** – Nag\_OrderType *Input*  
*On entry:* the **order** argument 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 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.  
*Constraint:* **order** = Nag\_RowMajor or Nag\_ColMajor.
- 2: **vect** – Nag\_VectType *Input*  
*On entry:* indicates whether the matrices  $Q$  and/or  $P^H$  are generated.  
**vect** = Nag\_DoNotForm  
Neither  $Q$  nor  $P^H$  is generated.  
**vect** = Nag\_FormQ  
 $Q$  is generated.  
**vect** = Nag\_FormP  
 $P^H$  is generated.  
**vect** = Nag\_FormBoth  
Both  $Q$  and  $P^H$  are generated.  
*Constraint:* **vect** = Nag\_DoNotForm, Nag\_FormQ, Nag\_FormP or Nag\_FormBoth.

- 3: **m** – Integer *Input*  
*On entry:*  $m$ , the number of rows of the matrix  $A$ .  
*Constraint:*  $\mathbf{m} \geq 0$ .
- 4: **n** – Integer *Input*  
*On entry:*  $n$ , the number of columns of the matrix  $A$ .  
*Constraint:*  $\mathbf{n} \geq 0$ .
- 5: **ncc** – Integer *Input*  
*On entry:*  $n_C$ , the number of columns of the matrix  $C$ .  
*Constraint:*  $\mathbf{ncc} \geq 0$ .
- 6: **kl** – Integer *Input*  
*On entry:* the number of subdiagonals,  $k_l$ , within the band of  $A$ .  
*Constraint:*  $\mathbf{kl} \geq 0$ .
- 7: **ku** – Integer *Input*  
*On entry:* the number of superdiagonals,  $k_u$ , within the band of  $A$ .  
*Constraint:*  $\mathbf{ku} \geq 0$ .
- 8: **ab**[*dim*] – Complex *Input/Output*  
**Note:** the dimension, *dim*, of the array **ab** must be at least  
 $\max(1, \mathbf{pdab} \times \mathbf{n})$  when **order** = Nag\_ColMajor;  
 $\max(1, \mathbf{m} \times \mathbf{pdab})$  when **order** = Nag\_RowMajor.  
*On entry:* the original  $m$  by  $n$  band matrix  $A$ .  
This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements  $A_{ij}$ , for row  $i = 1, \dots, m$  and column  $j = \max(1, i - k_l), \dots, \min(n, i + k_u)$ , depends on the **order** argument as follows:  
if **order** = Nag\_ColMajor,  $A_{ij}$  is stored as  $\mathbf{ab}[(j - 1) \times \mathbf{pdab} + \mathbf{ku} + i - j]$ ;  
if **order** = Nag\_RowMajor,  $A_{ij}$  is stored as  $\mathbf{ab}[(i - 1) \times \mathbf{pdab} + \mathbf{kl} + j - i]$ .  
*On exit:* **ab** is overwritten by values generated during the reduction.
- 9: **pdab** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) of the matrix  $A$  in the array **ab**.  
*Constraint:*  $\mathbf{pdab} \geq \mathbf{kl} + \mathbf{ku} + 1$ .
- 10: **d**[**min(m, n)**] – double *Output*  
*On exit:* the diagonal elements of the bidiagonal matrix  $B$ .
- 11: **e**[**min(m, n) - 1**] – double *Output*  
*On exit:* the superdiagonal elements of the bidiagonal matrix  $B$ .

12: **q**[*dim*] – Complex

Output

**Note:** the dimension, *dim*, of the array **q** must be at least
$$\max(1, \mathbf{pdq} \times \mathbf{m}) \text{ when } \mathbf{vect} = \text{Nag\_FormQ or Nag\_FormBoth};$$

$$1 \text{ otherwise.}$$
The (*i*, *j*)th element of the matrix *Q* is stored in
$$\mathbf{q}[(j-1) \times \mathbf{pdq} + i - 1] \text{ when } \mathbf{order} = \text{Nag\_ColMajor};$$

$$\mathbf{q}[(i-1) \times \mathbf{pdq} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
*On exit:* if **vect** = Nag\_FormQ or Nag\_FormBoth, contains the *m* by *m* unitary matrix *Q*.If **vect** = Nag\_DoNotForm or Nag\_FormP, **q** is not referenced.13: **pdq** – Integer

Input

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

$$\text{if } \mathbf{vect} = \text{Nag\_FormQ or Nag\_FormBoth, } \mathbf{pdq} \geq \max(1, \mathbf{m});$$

$$\text{otherwise } \mathbf{pdq} \geq 1.$$
14: **pt**[*dim*] – Complex

Output

**Note:** the dimension, *dim*, of the array **pt** must be at least
$$\max(1, \mathbf{pdpt} \times \mathbf{n}) \text{ when } \mathbf{vect} = \text{Nag\_FormP or Nag\_FormBoth};$$

$$1 \text{ otherwise.}$$
The (*i*, *j*)th element of the matrix is stored in
$$\mathbf{pt}[(j-1) \times \mathbf{pdpt} + i - 1] \text{ when } \mathbf{order} = \text{Nag\_ColMajor};$$

$$\mathbf{pt}[(i-1) \times \mathbf{pdpt} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
*On exit:* the *n* by *n* unitary matrix  $P^H$ , if **vect** = Nag\_FormP or Nag\_FormBoth. If **vect** = Nag\_DoNotForm or Nag\_FormQ, **pt** is not referenced.15: **pdpt** – Integer

Input

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

$$\text{if } \mathbf{vect} = \text{Nag\_FormP or Nag\_FormBoth, } \mathbf{pdpt} \geq \max(1, \mathbf{n});$$

$$\text{otherwise } \mathbf{pdpt} \geq 1.$$
16: **c**[*dim*] – Complex

Input/Output

**Note:** the dimension, *dim*, of the array **c** must be at least
$$\max(1, \mathbf{pdc} \times \mathbf{ncc}) \text{ when } \mathbf{order} = \text{Nag\_ColMajor};$$

$$\max(1, \mathbf{m} \times \mathbf{pdc}) \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
The (*i*, *j*)th element of the matrix *C* is stored in
$$\mathbf{c}[(j-1) \times \mathbf{pdc} + i - 1] \text{ when } \mathbf{order} = \text{Nag\_ColMajor};$$

$$\mathbf{c}[(i-1) \times \mathbf{pdc} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
*On entry:* an *m* by *n<sub>C</sub>* matrix *C*.*On exit:* **c** is overwritten by  $Q^H C$ . If **ncc** = 0, **c** is not referenced.

17: **pdc** – Integer *Input*

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

*Constraints:*

if **order** = Nag\_ColMajor,  
     if **ncc** > 0, **pdc** ≥ max(1, **m**);  
     if **ncc** = 0, **pdc** ≥ 1.;  
 if **order** = Nag\_RowMajor, **pdc** ≥ max(1, **ncc**).

18: **fail** – NagError \* *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_ENUM\_INT\_2

On entry, **vect** =  $\langle value \rangle$ , **pdpt** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: if **vect** = Nag\_FormP or Nag\_FormBoth, **pdpt** ≥ max(1, **n**);  
 otherwise **pdpt** ≥ 1.

On entry, **vect** =  $\langle value \rangle$ , **pdq** =  $\langle value \rangle$  and **m** =  $\langle value \rangle$ .

Constraint: if **vect** = Nag\_FormQ or Nag\_FormBoth, **pdq** ≥ max(1, **m**);  
 otherwise **pdq** ≥ 1.

### NE\_INT

On entry, **kl** =  $\langle value \rangle$ .

Constraint: **kl** ≥ 0.

On entry, **ku** =  $\langle value \rangle$ .

Constraint: **ku** ≥ 0.

On entry, **m** =  $\langle value \rangle$ .

Constraint: **m** ≥ 0.

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n** ≥ 0.

On entry, **ncc** =  $\langle value \rangle$ .

Constraint: **ncc** ≥ 0.

On entry, **pdab** =  $\langle value \rangle$ .

Constraint: **pdab** > 0.

On entry, **pdc** =  $\langle value \rangle$ .

Constraint: **pdc** > 0.

On entry, **pdpt** =  $\langle value \rangle$ .

Constraint: **pdpt** > 0.

On entry, **pdq** =  $\langle value \rangle$ .  
 Constraint: **pdq** > 0.

#### NE\_INT\_2

On entry, **pdc** =  $\langle value \rangle$  and **ncc** =  $\langle value \rangle$ .  
 Constraint: **pdc** ≥ max(1, **ncc**).

#### NE\_INT\_3

On entry, **ncc** =  $\langle value \rangle$ , **pdc** =  $\langle value \rangle$  and **m** =  $\langle value \rangle$ .  
 Constraint: if **ncc** > 0, **pdc** ≥ max(1, **m**);  
 if **ncc** = 0, **pdc** ≥ 1.

On entry, **pdab** =  $\langle value \rangle$ , **kl** =  $\langle value \rangle$  and **ku** =  $\langle value \rangle$ .  
 Constraint: **pdab** ≥ **kl** + **ku** + 1.

#### 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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.  
 See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

#### NE\_NO\_LICENCE

Your licence key may have expired or may not have been installed correctly.  
 See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

The computed bidiagonal form  $B$  satisfies  $QBP^H = A + E$ , where

$$\|E\|_2 \leq c(n)\epsilon\|A\|_2,$$

$c(n)$  is a modestly increasing function of  $n$ , and  $\epsilon$  is the *machine precision*.

The elements of  $B$  themselves may be sensitive to small perturbations in  $A$  or to rounding errors in the computation, but this does not affect the stability of the singular values and vectors.

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

$$\|F\|_2 = O(\epsilon).$$

A similar statement holds for the computed matrix  $P^H$ .

## 8 Parallelism and Performance

nag\_zgbbbrd (f08lsc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

The total number of real floating-point operations is approximately the sum of:

$20n^2k$ , if **vect** = Nag\_DoNotForm and **ncc** = 0, and

$10n^2n_C(k-1)/k$ , if  $C$  is updated, and

$10n^3(k-1)/k$ , if either  $Q$  or  $P^H$  is generated (double this if both), where  $k = k_l + k_u$ , assuming  $n \gg k$ . For this section we assume that  $m = n$ . The real analogue of this function is nag\_dgbbrd (f08lec).

## 10 Example

This example reduces the matrix  $A$  to upper bidiagonal form, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & 0.00 + 0.00i & 0.00 + 0.00i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & 0.00 + 0.00i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ 0.00 + 0.00i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.17 - 0.46i & 1.47 + 1.59i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 0.26 + 0.26i \end{pmatrix}.$$

### 10.1 Program Text

```
/* nag_zgbbrd (f08lsc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
    /* Scalars */
    Integer i, j, kl, ku, m, n, ncc, pdab, pdc, pdq, pdpt;
    Integer d_len, e_len;
    Integer exit_status = 0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Complex *ab = 0, *c = 0, *pt = 0, *q = 0;
    double *d = 0, *e = 0;

#ifdef NAG_COLUMN_MAJOR
#define AB(I, J) ab[(J - 1) * pdab + ku + I - J]
    order = Nag_ColMajor;
#else
#define AB(I, J) ab[(I - 1) * pdab + kl + J - I]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);

    printf("nag_zgbbrd (f08lsc) Example Program Results\n");

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT
            "%*[\n] ", &m, &n, &kl, &ku, &ncc);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT
            "%*[\n] ", &m, &n, &kl, &ku, &ncc);
#endif
}
```

```

        "%*[^\\n] ", &m, &n, &kl, &ku, &ncc);
#endif
#ifdef NAG_COLUMN_MAJOR
    pdab = kl + ku + 1;
    pdq = m;
    pdpt = n;
    pdc = m;
#else
    pdab = kl + ku + 1;
    pdq = m;
    pdpt = n;
    pdc = MAX(1, ncc);
#endif
    d_len = MIN(m, n);
    e_len = MIN(m, n) - 1;

    /* Allocate memory */
    if (!(ab = NAG_ALLOC((kl + ku + 1) * m, Complex)) ||
        !(c = NAG_ALLOC(m * MAX(1, ncc), Complex)) ||
        !(d = NAG_ALLOC(d_len, double)) ||
        !(e = NAG_ALLOC(e_len, double)) ||
        !(pt = NAG_ALLOC(n * n, Complex)) || !(q = NAG_ALLOC(m * m, Complex)))
    {
        printf("Allocation failure\\n");
        exit_status = -1;
        goto END;
    }

    /* Read A from data file */
    for (i = 1; i <= m; ++i) {
        for (j = MAX(1, i - kl); j <= MIN(n, i + ku); ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &AB(i, j).re, &AB(i, j).im);
#else
            scanf(" ( %lf , %lf )", &AB(i, j).re, &AB(i, j).im);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif

    /* Reduce A to bidiagonal form */
    /* nag_zgbbbrd (f08lsc).
    * Reduction of complex rectangular band matrix to upper
    * bidiagonal form
    */
    nag_zgbbbrd(order, Nag_DoNotForm, m, n, ncc, kl, ku, ab,
                pdab, d, e, q, pdq, pt, pdpt, c, pdc, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zgbbbrd (f08lsc).\\n%s\\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print bidiagonal form */
    printf("\\nDiagonal\\n");
    for (i = 1; i <= MIN(m, n); ++i)
        printf("%9.4f%s", d[i - 1], i % 8 == 0 ? "\\n" : " ");
    if (m >= n)
        printf("\\nSuperdiagonal\\n");
    else
        printf("\\nSubdiagonal\\n");
    for (i = 1; i <= MIN(m, n) - 1; ++i)
        printf("%9.4f%s", e[i - 1], i % 8 == 0 ? "\\n" : " ");
    printf("\\n");

END:
    NAG_FREE(ab);
    NAG_FREE(c);
    NAG_FREE(d);

```

```

    NAG_FREE(e);
    NAG_FREE(pt);
    NAG_FREE(q);

    return exit_status;
}

```

## 10.2 Program Data

```

nag_zgbbrd (f08lsc) Example Program Data
  6  4  2  1  0                               :Values of M, N, KL, KU and NCC
( 0.96,-0.81) (-0.03, 0.96)
(-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42)
( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
               ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
               (-0.17,-0.46) ( 1.47, 1.59)
               ( 0.26, 0.26) :End of matrix A

```

## 10.3 Program Results

```

nag_zgbbrd (f08lsc) Example Program Results

```

```

Diagonal
  2.6560    1.7501    2.0607    0.8658
Superdiagonal
  1.7033    1.2800    0.1467

```

---