NAG Library Function Document
nag_dgbbrd (f08lec)

1 Purpose
nag_dgbbrd (f08lec) reduces a real m by n band matrix to upper bidiagonal form.

2 Specification
#include <nag.h>
#include <nagf08.h>
void nag_dgbbrd (Nag_OrderType order, Nag_VectType vect, Integer m,
    Integer n, Integer ncc, Integer kl, Integer ku, double ab[],
    Integer pdab, double d[], double e[], double q[], Integer pdq,
    double pt[], Integer pdpt, double c[], Integer pdc, NagError *fail)

3 Description
nag_dgbbrd (f08lec) reduces a real m by n band matrix to upper bidiagonal form B by an orthogonal transformation: \( A = QBP^T \). The orthogonal matrices \( Q \) and \( P^T \), 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 \( \tilde{C} = Q^T C \).

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 3.2.1.3 in the Essential Introduction 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 \( P^T \) are generated.
   vect = Nag_DoNotForm
   Neither \( Q \) nor \( P^T \) is generated.
   vect = Nag_FormQ
   \( Q \) is generated.
   vect = Nag_FormP
   \( P^T \) is generated.
   vect = Nag_FormBoth
   Both \( Q \) and \( P^T \) are generated.
   Constraint: vect = Nag_DoNotForm, Nag_FormQ, Nag_FormP or Nag_FormBoth.
3:  \( m \) – Integer  
    \textit{Input}
    \textit{On entry:} \( m \), the number of rows of the matrix \( A \).  
    \textit{Constraint:} \( m \geq 0 \).

4:  \( n \) – Integer  
    \textit{Input}
    \textit{On entry:} \( n \), the number of columns of the matrix \( A \).  
    \textit{Constraint:} \( n \geq 0 \).

5:  \( ncc \) – Integer  
    \textit{Input}
    \textit{On entry:} \( nC \), the number of columns of the matrix \( C \).  
    \textit{Constraint:} \( ncc \geq 0 \).

6:  \( kl \) – Integer  
    \textit{Input}
    \textit{On entry:} the number of subdiagonals, \( k_1 \), within the band of \( A \).  
    \textit{Constraint:} \( kl \geq 0 \).

7:  \( ku \) – Integer  
    \textit{Input}
    \textit{On entry:} the number of superdiagonals, \( k_u \), within the band of \( A \).  
    \textit{Constraint:} \( ku \geq 0 \).

8:  \( ab[\text{dim}] \) – double  
    \textit{Input/Output}
    \textit{Note:} the dimension, \( \text{dim} \), of the array \( ab \) must be at least  
    \[
    \max(1, \text{pdab} \times n) \text{ when } \text{order} = \text{Nag\_ColMajor};  
    \max(1, m \times \text{pdab}) \text{ when } \text{order} = \text{Nag\_RowMajor}.  
    \]
    \textit{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 \( i = 1, \ldots, m \) and column  
    \( j = \max(1, i - k_1), \ldots, \min(n, i + k_u) \), depends on the \textit{order} argument as follows:  
    
    if \( \text{order} = \text{Nag\_ColMajor}, A_{ij} \text{ is stored as } ab[(j - 1) \times \text{pdab} + ku + i - j];  
    
    if \( \text{order} = \text{Nag\_RowMajor}, A_{ij} \text{ is stored as } ab[(i - 1) \times \text{pdab} + kl + j - i].  
    
    \textit{On exit:} \( ab \) is overwritten by values generated during the reduction.

9:  \( \text{pdab} \) – Integer  
    \textit{Input}
    \textit{On entry:} the stride separating row or column elements (depending on the value of \textit{order}) of the  
    matrix \( A \) in the array \( ab \).  
    \textit{Constraint:} \( \text{pdab} \geq kl + ku + 1 \).

10:  \( d[\min(m, n)] \) – double  
    \textit{Output}
    \textit{On exit:} the diagonal elements of the bidiagonal matrix \( B \).

11:  \( e[\min(m, n) - 1] \) – double  
    \textit{Output}
    \textit{On exit:} the superdiagonal elements of the bidiagonal matrix \( B \).
Note: the dimension, \( dim \), of the array \( q \) must be at least 
\[ \max(1, \text{pdq} \times m) \] when \( \text{vect} = \text{Nag\_FormQ} \) or \( \text{Nag\_FormBoth} \); 
1 otherwise.

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

On exit: if \( \text{vect} = \text{Nag\_FormQ} \) or \( \text{Nag\_FormBoth} \), contains the \( m \) by \( m \) orthogonal matrix \( Q \).
If \( \text{vect} = \text{Nag\_DoNotForm} \) or \( \text{Nag\_FormP} \), \( q \) is not referenced.

13: \( \text{pdq} \) – Integer 
Input

On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( q \).

Constraints:
\[
\text{if } \text{vect} = \text{Nag\_FormQ} \text{ or } \text{Nag\_FormBoth}, \quad \text{pdq} \geq \max(1, m); 
\text{otherwise } \text{pdq} \geq 1.
\]

14: \( \text{pt}[\text{dim}] \) – double 
Output

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

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

On exit: the \( n \) by \( n \) orthogonal matrix \( P^T \), if \( \text{vect} = \text{Nag\_FormP} \) or \( \text{Nag\_FormBoth} \). If \( \text{vect} = \text{Nag\_DoNotForm} \) or \( \text{Nag\_FormQ} \), \( pt \) is not referenced.

15: \( \text{pdpt} \) – Integer 
Input

On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( pt \).

Constraints:
\[
\text{if } \text{vect} = \text{Nag\_FormP} \text{ or } \text{Nag\_FormBoth}, \quad \text{pdpt} \geq \max(1, n); 
\text{otherwise } \text{pdpt} \geq 1.
\]

16: \( \text{c}[\text{dim}] \) – double 
Input/Output

Note: the dimension, \( dim \), of the array \( c \) must be at least 
\[ \max(1, \text{pdc} \times \text{ncc}) \] when \( \text{order} = \text{Nag\_ColMajor}; 
\max(1, \text{m} \times \text{pdc}) \] when \( \text{order} = \text{Nag\_RowMajor}. 

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

On entry: an \( m \) by \( nC \) matrix \( C \). 
On exit: \( c \) is overwritten by \( Q^T C \). If \( \text{ncc} = 0 \), \( c \) is not referenced.
17:  \textbf{pdc} – Integer  

\textit{Input}

\textit{On entry}: the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textit{c}.

\textit{Constraints}:

\begin{align*}
\text{if } \textbf{order} = \text{Nag\_ColMajor,} & \\
\text{if } \text{ncc} > 0, \text{ } \textbf{pdc} \geq \max(1, m); & \\
\text{if } \text{ncc} = 0, \text{ } \textbf{pdc} \geq 1; & \\
\text{if } \textbf{order} = \text{Nag\_RowMajor, } \textbf{pdc} \geq \max(1, \text{ncc}).&
\end{align*}

18:  \textbf{fail} – NagError *  

\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6  \textbf{Error Indicators and Warnings}

\textbf{NE\_ALLOC\_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE\_BAD\_PARAM}

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

\textbf{NE\_ENUM\_INT\_2}

On entry, \textbf{vect} = \langle value \rangle, \textbf{pdpt} = \langle value \rangle and \textbf{n} = \langle value \rangle.
Constraint: if \textbf{vect} = \text{Nag\_FormP} or \text{Nag\_FormBoth, } \textbf{pdpt} \geq \max(1, \text{n});
otherwise \textbf{pdpt} \geq 1.

On entry, \textbf{vect} = \langle value \rangle, \textbf{pdq} = \langle value \rangle and \textbf{m} = \langle value \rangle.
Constraint: if \textbf{vect} = \text{Nag\_FormQ} or \text{Nag\_FormBoth, } \textbf{pdq} \geq \max(1, \text{m});
otherwise \textbf{pdq} \geq 1.

\textbf{NE\_INT}

On entry, \textbf{kl} = \langle value \rangle.
Constraint: \textbf{kl} \geq 0.

On entry, \textbf{ku} = \langle value \rangle.
Constraint: \textbf{ku} \geq 0.

On entry, \textbf{m} = \langle value \rangle.
Constraint: \textbf{m} \geq 0.

On entry, \textbf{n} = \langle value \rangle.
Constraint: \textbf{n} \geq 0.

On entry, \textbf{ncc} = \langle value \rangle.
Constraint: \textbf{ncc} \geq 0.

On entry, \textbf{pdab} = \langle value \rangle.
Constraint: \textbf{pdab} > 0.

On entry, \textbf{pdc} = \langle value \rangle.
Constraint: \textbf{pdc} > 0.

On entry, \textbf{pdpt} = \langle value \rangle.
Constraint: \textbf{pdpt} > 0.

On entry, \textbf{pdq} = \langle value \rangle.
Constraint: \textbf{pdq} > 0.
On entry, \( pdc = \langle \text{value} \rangle \) and \( ncc = \langle \text{value} \rangle \).
Constraint: \( pdc \geq \max(1, ncc) \).

On entry, \( ncc = \langle \text{value} \rangle \), \( pdc = \langle \text{value} \rangle \) and \( m = \langle \text{value} \rangle \).
Constraint: if \( ncc > 0 \), \( pdc \geq \max(1, m) \);
if \( ncc = 0 \), \( pdc \geq 1 \).

On entry, \( pdab = \langle \text{value} \rangle \), \( kl = \langle \text{value} \rangle \) and \( ku = \langle \text{value} \rangle \).
Constraint: \( pdab \geq kl + ku + 1 \).

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 3.6.6 in the Essential Introduction for further information.

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

The computed bidiagonal form \( B \) satisfies \( QBPT = A + E \), where
\[
\|E\|_2 \leq c(n)\epsilon\|A\|_2,
\]
c\( (n) \) is a modestly increasing function of \( n \), and \( \epsilon \) is the \textit{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 orthogonal matrix by a matrix \( F \) such that
\[
\|F\|_2 = O(\epsilon).
\]
A similar statement holds for the computed matrix \( P^T \).

The total number of real floating-point operations is approximately the sum of:
- \( 6n^2k \), if \( \text{vect} = \text{Nag_DoNotForm} \) and \( ncc = 0 \), and
- \( 3n^2n_C(k - 1)/k \), if \( C \) is updated, and
- \( 3n^3(k - 1)/k \), if either \( Q \) or \( P^T \) 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 complex analogue of this function is nag_zgbbrd (f08lsc).

10 Example

This example reduces the matrix \( A \) to upper bidiagonal form, where

\[
A = \begin{pmatrix}
-0.57 & -1.28 & 0.00 & 0.00 \\
-1.93 & 1.08 & -0.31 & 0.00 \\
2.30 & 0.24 & 0.40 & -0.35 \\
0.00 & 0.64 & -0.66 & 0.08 \\
0.00 & 0.00 & 0.15 & -2.13 \\
-0.00 & 0.00 & 0.00 & 0.50
\end{pmatrix}.
\]

10.1 Program Text

/* nag_dgbbrd (f08lec) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
*/

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagf16.h>
#include <nagx02.h>
#include <nagx04.h>

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

    INIT_FAIL(fail);

    printf("nag_dgbbrd (f08lec) Example Program Results\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[^
"]
    #else
    f08lec.6 Mark 25
```c
#include "nagcmex.h"
#include "nagcmex.h"

/* Allocate memory */
if (!ab = NAG_ALLOC((kl+ku+1) * m, double)) ||
!aw = NAG_ALLOC(m * n, double)) ||
!f = NAG_ALLOC(m * n, double)) ||
!c = NAG_ALLOC(m * MAX(1, ncc), double)) ||
!(d = NAG_ALLOC(d_len, double)) ||
!(e = NAG_ALLOC(e_len, double)) ||
!(pt = NAG_ALLOC(n * n, double)) ||
!(q = NAG_ALLOC(m * m, double)))
{
    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)
    {
        if(j >= MAX(1, i - kl) && j<= MIN(n, i + ku))
            AW(i, j) = AB(i, j);
        else
            AW(i, j) = 0;
    }
}

/* Copy AB into AW */
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= n; ++j)
    {
        if(j >= MAX(1, i - kl) & j<= MIN(n, i + ku))
            AW(i, j) = AB(i, j);
        else
            AW(i, j) = 0;
    }
}
```
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n,
aw, pdaw, "Matrix A", 0, &fail);
printf("\n");
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Reduce A to bidiagonal form */
/* nag_dgbbrd (f08lec). */
* Reduction of real rectangular band matrix to upper
* bidiagonal form */
nag_dgbbrd(order, Nag_FormBoth, 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_dgbbrd (f08lec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* F = Q*B */
for(i = 1; i <= m; i++)
{
    F(i, 1) = Q(i, 1) * d[0];
    for(j = 2; j <= n; j++)
    {
        F(i, j) = (Q(i, j) * d[j-1]) + (Q(i, j-1) * e[j-2]);
    }
}

/* nag_dgemm (f16yac): Compute A - Q*B*P^T from the factorization of A */
/* and store in matrix AW */
alpha = -1.0;
beta = 1.0;
nag_dgemm(order, Nag_NoTrans, Nag_NoTrans, m, n, n, alpha, f, pdf,
pt, pdpt, beta, aw, pdaw, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dgemm (f16yac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_dge_norm (f16rac): Find norm of matrix AW and print warning if */
/* it is too large*/
nag_dge_norm(order, Nag_OneNorm, m, n, aw, pdaw, &norm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dge_norm (f16rac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
if (norm > pow(x02ajc(),0.8))
{
    printf("%s\n","Norm of A-(Q*B*P^T) is much greater than 0."
    "Schur factorization has failed.");
}

END:
NAG_FREE(ab);
NAG_FREE(aw);
NAG_FREE(c);
NAG_FREE(d);
NAG_FREE(e);
NAG_FREE(f);
NAG_FREE(pt);
NAG_FREE(q);

return exit_status;
}

10.2 Program Data

nag_dgbbrd (f08lec) Example Program Data
6 4 2 1 0 :Values of M, N, KL, KU and NCC
-0.57 -1.28
-1.93 1.08 -0.31
2.30 0.24 0.40 -0.35
0.64 -0.66 0.08
0.15 -2.13
0.50 :End of matrix A

10.3 Program Results

nag_dgbbrd (f08lec) Example Program Results

Matrix A

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.57</td>
<td>-1.28</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>-1.93</td>
<td>1.080</td>
<td>-0.31</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>2.300</td>
<td>0.240</td>
<td>0.400</td>
<td>-0.35</td>
</tr>
<tr>
<td>4</td>
<td>0.000</td>
<td>0.640</td>
<td>-0.66</td>
<td>0.080</td>
</tr>
<tr>
<td>5</td>
<td>0.000</td>
<td>0.000</td>
<td>0.150</td>
<td>-2.13</td>
</tr>
<tr>
<td>6</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.500</td>
</tr>
</tbody>
</table>