NAG C Library Function Document

nag_dgebrd (f08kec)

1 Purpose

nag_dgebrd (f08kec) reduces a real m by n matrix to bidiagonal form.

2 Specification

```
void nag_dgebrd (Nag_OrderType order, Integer m, Integer n, double a[],
      Integer pda, double d[], double e[], double tauq[], double taup[],
      NagError *fail)
```

3 Description

nag_dgebrd (f08kec) reduces a real m by n matrix A to bidiagonal form B by an orthogonal transformation: $A = QBP^T$, where Q and P^T are orthogonal matrices of order m and n respectively.

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

$$A = Q\begin{pmatrix} B_1 \\ 0 \end{pmatrix}P^T = Q_1 B_1 P^T,$$

where B_1 is an n by n upper bidiagonal matrix and Q_1 consists of the first n columns of Q.

If $m < n$, the reduction is given by

$$A = Q(\, B_1 \quad 0 \,)P^T = QB_1 P_1^T,$$

where B_1 is an m by m lower bidiagonal matrix and P_1^T consists of the first m rows of P^T.

The orthogonal matrices Q and P are not formed explicitly but are represented as products of elementary reflectors (see the f08 Chapter Introduction for details). Functions are provided to work with Q and P in this representation (see Section 8).

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 number of rows of the matrix A.

Constraint: **m** ≥ 0.

3: **n** – Integer *Input*

On entry: n, the number of columns of the matrix A.

Constraint: **n** ≥ 0.

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

On entry: the m by n matrix A.

On exit: if $m \geq n$, the diagonal and first super-diagonal are overwritten by the upper bidiagonal matrix B, elements below the diagonal are overwritten by details of the orthogonal matrix Q and elements above the first super-diagonal are overwritten by details of the orthogonal matrix P.

If $m < n$, the diagonal and first sub-diagonal are overwritten by the lower bidiagonal matrix B, elements below the first sub-diagonal are overwritten by details of the orthogonal matrix Q and elements above the diagonal are overwritten by details of the orthogonal matrix P.

5: **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** $\geq \max(1, \mathbf{n})$.

6: **d**[dim] – double *Output*

Note: the dimension, dim, of the array **d** must be at least $\max(1, \min(\mathbf{m}, \mathbf{n}))$.

On exit: the diagonal elements of the bidiagonal matrix B.

7: **e**[dim] – double *Output*

Note: the dimension, dim, of the array **e** must be at least $\max(1, \min(\mathbf{m}, \mathbf{n}) - 1)$.

On exit: the off-diagonal elements of the bidiagonal matrix B.

8: **tauq**[dim] – double *Output*

Note: the dimension, dim, of the array **tauq** must be at least $\max(1, \min(\mathbf{m}, \mathbf{n}))$.

On exit: further details of the orthogonal matrix Q.

9: **taup**[dim] – double *Output*

Note: the dimension, dim, of the array **taup** must be at least $\max(1, \min(\mathbf{m}, \mathbf{n}))$.

On exit: further details of the orthogonal matrix P.

10: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **m** = $\langle$*value*$\rangle$.
Constraint: **m** ≥ 0.

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

On entry, **pda** = ⟨*value*⟩.
Constraint: **pda** > 0.

NE_INT_2

On entry, **pda** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **pda** $\geq \max(1, \mathbf{m})$.

On entry, **pda** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pda** $\geq \max(1, \mathbf{n})$.

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 bidiagonal form B satisfies $QBP^T = A + E$, where

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

$c(n)$ is a modestly increasing function of n, and ϵ 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.

8 Further Comments

The total number of floating-point operations is approximately $\frac{4}{3}n^2(3m - n)$ if $m \geq n$ or $\frac{4}{3}m^2(3n - m)$ if $m < n$.

If $m \gg n$, it can be more efficient to first call nag_dgeqrf (f08aec) to perform a QR factorization of A, and then to call this function to reduce the factor R to bidiagonal form. This requires approximately $2n^2(m + n)$ floating-point operations.

If $m \ll n$, it can be more efficient to first call nag_dgelqf (f08ahc) to perform an LQ factorization of A, and then to call this function to reduce the factor L to bidiagonal form. This requires approximately $2m^2(m + n)$ operations.

To form the orthogonal matrices P^T and/or Q, this function may be followed by calls to nag_dorgbr (f08kfc):

to form the m by m orthogonal matrix Q

```
nag_dorgbr (order,Nag_FormQ,m,m,n,&a,pda,tauq,&fail)
```

but note that the second dimension of the array **a** must be at least **m**, which may be larger than was required by nag_dgebrd (f08kec);

to form the n by n orthogonal matrix P^T

```
nag_dorgbr (order,Nag_FormP,n,n,m,&a,pda,taup,&fail)
```

but note that the first dimension of the array **a**, specified by the parameter **pda**, must be at least **n**, which may be larger than was required by nag_dgebrd (f08kec).

To apply Q or P to a real rectangular matrix C, this function may be followed by a call to nag_dormbr (f08kgc).

The complex analogue of this function is nag_zgebrd (f08ksc).

9 Example

To reduce the matrix A to bidiagonal form, where

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}.$$

9.1 Program Text

```
/* nag_dgebrd (f08kec) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

int main(void)
{
  /* Scalars */
  Integer  i, j, m, n, pda, d_len, e_len, tauq_len, taup_len;
  Integer  exit_status=0;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  double *a=0, *d=0, *e=0, *taup=0, *tauq=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("f08kec Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%ld%*[^\n] ", &m, &n);
#ifdef NAG_COLUMN_MAJOR
  pda = m;
#else
  pda = n;
#endif
  d_len = MIN(m,n);
  e_len = MIN(m,n)-1;
  tauq_len = MIN(m,n);
  taup_len = MIN(m,n);

  /* Allocate memory */
```

```
  if ( !(a = NAG_ALLOC(m * n, double)) ||
       !(d = NAG_ALLOC(d_len, double)) ||
       !(e = NAG_ALLOC(e_len, double)) ||
       !(taup = NAG_ALLOC(taup_len, double)) ||
       !(tauq = NAG_ALLOC(tauq_len, double)) )
    {
      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", &A(i,j));
    }
  Vscanf("%*[^\n] ");

  /* Reduce A to bidiagonal form */
  f08kec(order, m, n, a, pda, d, e, tauq, taup, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08kec.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

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

 END:
  if (a) NAG_FREE(a);
  if (d) NAG_FREE(d);
  if (e) NAG_FREE(e);
  if (taup) NAG_FREE(taup);
  if (tauq) NAG_FREE(tauq);

  return exit_status;
}
```

9.2 Program Data

```
f08kec Example Program Data
  6  4                          :Values of M and N
 -0.57  -1.28  -0.39   0.25
 -1.93   1.08  -0.31  -2.14
  2.30   0.24   0.40  -0.35
 -1.93   0.64  -0.66   0.08
  0.15   0.30   0.15  -2.13
 -0.02   1.03  -1.43   0.50   :End of matrix A
```

9.3 Program Results

```
f08kec Example Program Results

Diagonal
   3.6177    2.4161   -1.9213   -1.4265
Super-diagonal
   1.2587    1.5262   -1.1895
```