NAG Library Function Document

nag_dgbmv (f16pbc)

1 Purpose

nag_dgbmv (f16pbc) performs matrix-vector multiplication for a real band matrix.

2 Specification

```
#include <nag.h>
#include <nagf16.h>
```

```
void nag_dgbmv (Nag_OrderType order, Nag_TransType trans, Integer m,
     Integer n, Integer kl, Integer ku, double alpha, const double ab[],
     Integer pdab, const double x[], Integer incx, double beta, double y[],
     Integer incy, NagError *fail)
```

3 Description

nag_dgbmv (f16pbc) performs one of the matrix-vector operations

$$y \leftarrow \alpha Ax + \beta y, \qquad \text{or} \qquad y \leftarrow \alpha A^{\mathrm{T}}x + \beta y,$$

where A is an m by n real band matrix with k_l subdiagonals and k_u superdiagonals, x and y are real vectors, and α and β are real scalars.

If $m = 0$ or $n = 0$, no operation is performed.

4 References

Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001) *Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard* University of Tennessee, Knoxville, Tennessee http://www.netlib.org/blas/blast-forum/blas-report.pdf

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: **trans** – Nag_TransType *Input*

On entry: specifies the operation to be performed.

trans = Nag_NoTrans
$y \leftarrow \alpha Ax + \beta y$.

trans = Nag_Trans or Nag_ConjTrans
$y \leftarrow \alpha A^{\mathrm{T}}x + \beta y$.

Constraint: **trans** = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.

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: **kl** – Integer *Input*

On entry: k_l, the number of subdiagonals within the band of A.

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

6: **ku** – Integer *Input*

On entry: k_u, the number of superdiagonals within the band of A.

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

7: **alpha** – double *Input*

On entry: the scalar α.

8: **ab**[*dim*] – const double *Input*

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 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 $\mathbf{ab}[(i-1) \times \mathbf{pdab} + \mathbf{kl} + j - i]$.

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: **x**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **x** must be at least

$\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incx}|)$ when **trans** = Nag_NoTrans;
$\max(1, 1 + (\mathbf{m} - 1)|\mathbf{incx}|)$ when **trans** = Nag_Trans or Nag_ConjTrans.

On entry: the incremented array **x** must contain the vector x.

11: **incx** – Integer *Input*

On entry: the increment in the subscripts of **x** between successive elements of x.

Constraint: $\mathbf{incx} \neq 0$.

12: **beta** – double *Input*

On entry: the scalar β.

13: **y**[dim] – double *Input/Output*

Note: the dimension, *dim*, of the array **y** must be at least

$\max(1, 1 + (\mathbf{m} - 1)|\mathbf{incy}|)$ when **trans** = Nag_NoTrans;
$\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incy}|)$ when **trans** = Nag_Trans or Nag_ConjTrans.

On entry: the incremented array **y** must contain the vector x.

If **beta** = 0, **y** need not be set.

On exit: the updated vector y.

14: **incy** – Integer *Input*

On entry: the increment in the subscripts of **y** between successive elements of y.

Constraint: **incy** $\neq 0$.

15: **fail** – NagError * *Input/Output*

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_INT

On entry, **incx** = ⟨*value*⟩.
Constraint: **incx** $\neq 0$.

On entry, **incy** = ⟨*value*⟩.
Constraint: **incy** $\neq 0$.

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

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

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

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

NE_INT_3

On entry, **pdab** = ⟨*value*⟩, **kl** = ⟨*value*⟩, **ku** = ⟨*value*⟩.
Constraint: **pdab** $\geq$ **kl** + **ku** + 1.

7 Accuracy

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001)).

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

A vector y, of length 6, is updated using $y \leftarrow 2y + Ax$, where A is a 6 by 4 banded matrix with two subdiagonals and one superdiagonal, and x is a vector of length 4.

10.1 Program Text

```
/* nag_dgbmv (f16pbc) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2005.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf16.h>

int main(void)
{

  /* Scalars */
  double        alpha, beta;
  Integer       ab_size, exit_status, i, incx, incy, j, kl, ku;
  Integer       m, n, pdab, xlen, ylen;

  /* Arrays */
  double        *ab = 0, *x = 0, *y = 0;
  char          nag_enum_arg[40];

  /* Nag Types */
  NagError      fail;
  Nag_OrderType order;
  Nag_TransType trans;

#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

  exit_status = 0;
  INIT_FAIL(fail);

  printf("nag_dgbmv (f16pbc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");

  /* Read the problem dimensions */
  scanf("%ld%ld%ld%ld%*[^\n] ",
         &m, &n, &kl, &ku);
  /* Read the transpose parameter */
  scanf("%39s%*[^\n] ", nag_enum_arg);
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  trans = (Nag_TransType) nag_enum_name_to_value(nag_enum_arg);
  /* Read scalar parameters */
  scanf("%lf %lf%*[^\n] ", &alpha, &beta);
  /* Read increment parameters */
  scanf("%ld%ld%*[^\n] ", &incx, &incy);

  pdab = kl + ku + 1;
#ifdef NAG_COLUMN_MAJOR
  ab_size = pdab*n;
#else
  ab_size = pdab*m;
#endif

  if (trans == Nag_NoTrans)
    {
      xlen = MAX(1, 1 + (n - 1)*ABS(incx));
      ylen = MAX(1, 1 + (m - 1)*ABS(incy));
    }
  else
    {
      xlen = MAX(1, 1 + (m - 1)*ABS(incx));
      ylen = MAX(1, 1 + (n - 1)*ABS(incy));
    }

  if (m > 0 && n > 0)
    {
      /* Allocate memory */
      if (!(ab = NAG_ALLOC(ab_size, double)) ||
          !(x = NAG_ALLOC(xlen, double)) ||
          !(y = NAG_ALLOC(ylen, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      printf("Invalid m or n\n");
      exit_status = 1;
      return exit_status;
    }

  /* Input matrix A and vectors x and y */

  for (i = 1; i <= m; ++i)
    {
      for (j = MAX(1, i-kl); j <= MIN(n, i+ku); ++j)
        scanf("%lf", &AB(i, j));
      scanf("%*[^\n] ");
    }
  for (i = 1; i <= xlen; ++i)
    scanf("%lf%*[^\n] ", &x[i - 1]);
  for (i = 1; i <= ylen; ++i)
    scanf("%lf%*[^\n] ", &y[i - 1]);

  /* nag_dgbmv (f16pbc).
   * real valued band matrix-vector multiply.
   *
   */
  nag_dgbmv(order, trans, m, n, kl, ku, alpha, ab, pdab, x,
            incx, beta, y, incy, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgbmv.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print output vector y */
  printf("Updated vector y:\n\n");
  for (i = 1; i <= ylen; ++i)
    {
      printf("%11f\n", y[i-1]);
    }

 END:
  NAG_FREE(ab);
  NAG_FREE(x);
  NAG_FREE(y);

  return exit_status;
}
```

10.2 Program Data

```
nag_dgbmv (f16pbc) Example Program Data
  6 4 2 1          :Values of m, n, kl, ku
  Nag_NoTrans      : trans
  1.0  2.0         : alpha, beta
  1 1              : incx, incy
  1.0 1.0
  2.0 2.0 2.0
  3.0 3.0 3.0 3.0
      4.0 4.0 4.0
          5.0 5.0
              6.0  : the end of matrix A
  1.0
  2.0
  3.0
  4.0              : the end of vector x
  -0.5
  -4.5
  -13.0
  -15.5
  -14.5
  -8.5            : the end of vector y
```

10.3 Program Results

```
nag_dgbmv (f16pbc) Example Program Results

Updated vector y:

   2.000000
   3.000000
   4.000000
   5.000000
   6.000000
   7.000000
```