

NAG Library Function Document

nag_dtbmv (f16pgc)

1 Purpose

nag_dtbmv (f16pgc) performs matrix-vector multiplication for a real triangular band matrix.

2 Specification

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

void nag_dtbmv (Nag_OrderType order, Nag_UploType uplo, Nag_TransType trans,
               Nag_DiagType diag, Integer n, Integer k, double alpha,
               const double ab[], Integer pdab, double x[], Integer incx,
               NagError *fail)
```

3 Description

nag_dtbmv (f16pgc) performs one of the matrix-vector operations

$$x \leftarrow \alpha Ax \quad \text{or} \quad x \leftarrow \alpha A^T x,$$

where A is an n by n real triangular band matrix with k subdiagonals or superdiagonals, x is an n -element real vector and α is a real scalar.

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: **uplo** – Nag_UploType *Input*
On entry: specifies whether A is upper or lower triangular.
uplo = Nag_Upper
 A is upper triangular.
uplo = Nag_Lower
 A is lower triangular.
Constraint: **uplo** = Nag_Upper or Nag_Lower.

- 3: **trans** – Nag_TransType Input
On entry: specifies the operation to be performed.
trans = Nag_NoTrans
 $x \leftarrow \alpha Ax.$
trans = Nag_Trans or Nag_ConjTrans
 $x \leftarrow \alpha A^T x.$
Constraint: **trans** = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.
- 4: **diag** – Nag_DiagType Input
On entry: specifies whether A has nonunit or unit diagonal elements.
diag = Nag_NonUnitDiag
The diagonal elements are stored explicitly.
diag = Nag_UnitDiag
The diagonal elements are assumed to be 1 and are not referenced.
Constraint: **diag** = Nag_NonUnitDiag or Nag_UnitDiag.
- 5: **n** – Integer Input
On entry: n , the order of the matrix A .
Constraint: $n \geq 0$.
- 6: **k** – Integer Input
On entry: k , the number of subdiagonals or superdiagonals of the matrix A .
Constraint: $k \geq 0$.
- 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 $\max(1, \mathbf{pdab} \times \mathbf{n})$.
On entry: the n by n triangular band matrix A .
This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of A_{ij} , depends on the **order** and **uplo** arguments as follows:
if **order** = 'Nag_ColMajor' and **uplo** = 'Nag_Upper',
 A_{ij} is stored in **ab**[$k + i - j + (j - 1) \times \mathbf{pdab}$], for $j = 1, \dots, n$ and
 $i = \max(1, j - k), \dots, j$;
if **order** = 'Nag_ColMajor' and **uplo** = 'Nag_Lower',
 A_{ij} is stored in **ab**[$i - j + (j - 1) \times \mathbf{pdab}$], for $j = 1, \dots, n$ and
 $i = j, \dots, \min(n, j + k)$;
if **order** = 'Nag_RowMajor' and **uplo** = 'Nag_Upper',
 A_{ij} is stored in **ab**[$j - i + (i - 1) \times \mathbf{pdab}$], for $i = 1, \dots, n$ and
 $j = i, \dots, \min(n, i + k)$;
if **order** = 'Nag_RowMajor' and **uplo** = 'Nag_Lower',
 A_{ij} is stored in **ab**[$k + j - i + (i - 1) \times \mathbf{pdab}$], for $i = 1, \dots, n$ and
 $j = \max(1, i - k), \dots, i$.
If **diag** = 'Nag_UnitDiag', the diagonal elements of AB are assumed to be 1, and are not referenced.

- 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: $\mathbf{pdab} \geq \mathbf{k} + 1$.
- 10: $\mathbf{x}[\mathit{dim}]$ – double *Input/Output*
Note: the dimension, dim , of the array \mathbf{x} must be at least $\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incx}|)$.
On entry: the right-hand side vector b .
On exit: the solution vector x .
- 11: **incx** – Integer *Input*
On entry: the increment in the subscripts of \mathbf{x} between successive elements of x .
Constraint: $\mathbf{incx} \neq 0$.
- 12: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

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

NE_INT

On entry, $\mathbf{incx} = \langle \mathit{value} \rangle$.

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

On entry, $\mathbf{k} = \langle \mathit{value} \rangle$.

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

On entry, $\mathbf{n} = \langle \mathit{value} \rangle$.

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

NE_INT_2

On entry, $\mathbf{pdab} = \langle \mathit{value} \rangle$, $\mathbf{k} = \langle \mathit{value} \rangle$.

Constraint: $\mathbf{pdab} \geq \mathbf{k} + 1$.

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

This example computes the matrix-vector product

$$y = \alpha Ax$$

where

$$A = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 2.0 & 2.0 & 0.0 & 0.0 \\ 0.0 & 3.0 & 3.0 & 0.0 \\ 0.0 & 0.0 & 4.0 & 4.0 \end{pmatrix},$$

$$x = \begin{pmatrix} -1.0 \\ 2.0 \\ -3.0 \\ 4.0 \end{pmatrix}$$

and

$$\alpha = 1.5.$$

10.1 Program Text

```

/* nag_dtbmv (f16pgc) 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;
    Integer     exit_status, i, incx, j, k, kd, n, pdab, xlen;

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

    /* Nag Types */
    NagError    fail;
    Nag_DiagType diag;
    Nag_OrderType order;
    Nag_TransType trans;
    Nag_UploType uplo;

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

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_dtbmv (f16pgc) Example Program Results\n\n");

```

```

/* Skip heading in data file */
scanf("%*[\n] ");
/* Read the problem dimension */
scanf("%ld%ld%*[\n] ", &n, &kd);
/* Read uplo */
scanf("%39s%*[\n] ", nag_enum_arg);
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
/* Read trans */
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 diag */
scanf("%39s%*[\n] ", nag_enum_arg);
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
diag = (Nag_DiagType) nag_enum_name_to_value(nag_enum_arg);
/* Read scalar parameters */
scanf("%lf%*[\n] ", &alpha);
/* Read increment parameters */
scanf("%ld%*[\n] ", &incx);

pdab = kd + 1;
xlen = MAX(1, 1 + (n - 1)*ABS(incx));

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

/* Read A from data file */
k = kd + 1;
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= MIN(i+kd, n); ++j)
            scanf("%lf", &AB_UPPER(i, j));
    }
    scanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = MAX(1, i-kd); j <= i; ++j)
            scanf("%lf", &AB_LOWER(i, j));
    }
    scanf("%*[\n] ");
}

/* Input vector x */

```

```

for (i = 1; i <= xlen; ++i)
    scanf("%lf%*[^\\n] ", &x[i - 1]);

/* nag_dtbmv (f16pgc).
 * Triangular banded matrix-vector multiply.
 *
 */
nag_dtbmv(order, uplo, trans, diag, n, kd, alpha, ab, pdab,
          x, incx, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dtbmv (f16pgc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print output vector x */
printf("%s\n", " x");
for (i = 1; i <= xlen; ++i)
{
    printf("%11f\n", x[i-1]);
}

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

return exit_status;
}

```

10.2 Program Data

```

nag_dtbmv (f16pgc) Example Program Data
4 1 :Values of n, kd
Nag_Lower :Value of uplo
Nag_NoTrans :Value of trans
Nag_NonUnitDiag :Value of diag
1.5 :Value of alpha
1 :Value of incx
1.0
2.0 2.0
3.0 3.0
4.0 4.0 :End of matrix A
-1.0
2.0
-3.0
4.0 :End of vector x

```

10.3 Program Results

```

nag_dtbmv (f16pgc) Example Program Results

x
-1.500000
3.000000
-4.500000
6.000000

```
