NAG Library Function Document

nag_dgtsv (f07cac)

1 Purpose

nag_dgtsv (f07cac) computes the solution to a real system of linear equations

$$AX = B,$$

where A is an n by n tridiagonal matrix and X and B are n by r matrices.

2 Specification

```
#include <nag.h>
#include <nagf07.h>
```

```
void nag_dgtsv (Nag_OrderType order, Integer n, Integer nrhs, double dl[],
     double d[], double du[], double b[], Integer pdb, NagError *fail)
```

3 Description

nag_dgtsv (f07cac) uses Gaussian elimination with partial pivoting and row interchanges to solve the equations $AX = B$. The matrix A is factorized as $A = PLU$, where P is a permutation matrix, L is unit lower triangular with at most one nonzero subdiagonal element per column, and U is an upper triangular band matrix, with two superdiagonals.

Note that the equations $A^{\mathrm{T}}X = B$ may be solved by interchanging the order of the arguments **du** and **dl**.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

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: **n** – Integer *Input*

On entry: n, the number of linear equations, i.e., the order of the matrix A.

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

3: **nrhs** – Integer *Input*

On entry: r, the number of right-hand sides, i.e., the number of columns of the matrix B.

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

4: **dl**$[dim]$ – double *Input/Output*

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

On entry: must contain the $(n-1)$ subdiagonal elements of the matrix A.

On exit: if no constrains are violated, **dl** is overwritten by the $(n-2)$ elements of the second superdiagonal of the upper triangular matrix U from the LU factorization of A, in $\mathbf{dl}[0], \mathbf{dl}[1], \ldots, \mathbf{dl}[n-3]$.

5: **d**[*dim*] – double *Input/Output*

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

On entry: must contain the n diagonal elements of the matrix A.

On exit: if no constraints are violated, **d** is overwritten by the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

6: **du**[*dim*] – double *Input/Output*

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

On entry: must contain the $(n-1)$ superdiagonal elements of the matrix A.

On exit: if no constraints are violated, **du** is overwritten by the $(n-1)$ elements of the first superdiagonal of U.

7: **b**[*dim*] – double *Input/Output*

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

$\max(1, \mathbf{pdb} \times \mathbf{nrhs})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{n} \times \mathbf{pdb})$ when **order** = Nag_RowMajor.

The (i,j)th element of the matrix B is stored in

$\mathbf{b}[(j-1) \times \mathbf{pdb} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{b}[(i-1) \times \mathbf{pdb} + j - 1]$ when **order** = Nag_RowMajor.

On entry: the n by r right-hand side matrix B.

On exit: if **fail.code** = NE_NOERROR, the n by r solution matrix X.

8: **pdb** – Integer *Input*

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

Constraints:

if **order** = Nag_ColMajor, $\mathbf{pdb} \geq \max(1, \mathbf{n})$;
if **order** = Nag_RowMajor, $\mathbf{pdb} \geq \max(1, \mathbf{nrhs})$.

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

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

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

NE_INT_2

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

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

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.

NE_SINGULAR

$U(\langle value\rangle, \langle value\rangle)$ is exactly zero, and the solution has not been computed. The factorization has not been completed unless **n** = ⟨*value*⟩.

7 Accuracy

The computed solution for a single right-hand side, $\hat{x}$, satisfies an equation of the form

$$(A + E)\hat{x} = b,$$

where

$$\|E\|_1 = O(\epsilon)\|A\|_1$$

and ϵ is the ***machine precision***. An approximate error bound for the computed solution is given by

$$\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq \kappa(A)\frac{\|E\|_1}{\|A\|_1},$$

where $\kappa(A) = \left\|A^{-1}\right\|_1\|A\|_1$, the condition number of A with respect to the solution of the linear equations. See Section 4.4 of Anderson *et al.* (1999) for further details.

Alternatives to nag_dgtsv (f07cac), which return condition and error estimates are nag_real_tridiag_lin_solve (f04bcc) and nag_dgtsvx (f07cbc).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The total number of floating-point operations required to solve the equations $AX = B$ is proportional to nr.

The complex analogue of this function is nag_zgtsv (f07cnc).

10 Example

This example solves the equations

$$Ax = b,$$

where A is the tridiagonal matrix

$$A = \begin{pmatrix} 3.0 & 2.1 & 0 & 0 & 0 \\ 3.4 & 2.3 & -1.0 & 0 & 0 \\ 0 & 3.6 & -5.0 & 1.9 & 0 \\ 0 & 0 & 7.0 & -0.9 & 8.0 \\ 0 & 0 & 0 & -6.0 & 7.1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 2.7 \\ -0.5 \\ 2.6 \\ 0.6 \\ 2.7 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_dgtsv (f07cac) Example Program.
 *
 * Copyright 2004 Numerical Algorithms Group.
 *
 * Mark 23, 2011
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>

int main(void)
{

  /* Scalars */
  Integer       exit_status = 0, i, j, n, nrhs, pdb;

  /* Arrays */
  double        *b = 0, *d = 0, *dl = 0, *du = 0;

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

#ifdef NAG_COLUMN_MAJOR
#define B(I, J) b[(J-1)*pdb + I - 1]
  order = Nag_ColMajor;
#else
#define B(I, J) b[(I-1)*pdb + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_dgtsv (f07cac) Example Program Results\n\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld%ld%*[^\n]", &n, &nrhs);
  if (n < 0 || nrhs < 0)
    {
      printf("Invalid n or nrhs\n");
      exit_status = 1;
      goto END;
    }
  /* Allocate memory */
  if (!(b = NAG_ALLOC(n*nrhs, double)) ||
      !(d = NAG_ALLOC(n, double)) ||
      !(dl = NAG_ALLOC(n-1, double)) ||
      !(du = NAG_ALLOC(n-1, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
```

```
      goto END;
    }

#ifdef NAG_COLUMN_MAJOR
      pdb = n;
#else
      pdb = nrhs;
#endif

  /* Read the tridiagonal matrix A and the right hand side B from data file */
  for (i = 0; i < n - 1; ++i) scanf("%lf", &du[i]);
  scanf("%*[^\n]");
  for (i = 0; i < n; ++i) scanf("%lf", &d[i]);
  scanf("%*[^\n]");
  for (i = 0; i < n - 1; ++i) scanf("%lf", &dl[i]);
  scanf("%*[^\n]");

  for (i = 1; i <= n; ++i)
    for (j = 1; j <= nrhs; ++j) scanf("%lf", &B(i, j));
  scanf("%*[^\n]");

  /* Solve the equations Ax = b for x using nag_dgtsv (f07cac). */
  nag_dgtsv(order, n, nrhs, dl, d, du, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgtsv (f07cac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print solution */
  printf("Solution\n");
  for (i = 1; i <= n; ++i) {
    for (j = 1; j <= nrhs; ++j) printf("%11.4f%s", B(i, j), j%7 == 0?"\n":" ");
    printf("\n");
  }
 END:
  NAG_FREE(b);
  NAG_FREE(d);
  NAG_FREE(dl);
  NAG_FREE(du);
  return exit_status;
}
```

10.2 Program Data

```
nag_dgtsv (f07cac) Example Program Data

  5     1                         : n, nrhs
        2.1  -1.0   1.9   8.0   : superdiag of A
  3.0   2.3  -5.0  -0.9   7.1   : main diag of A
  3.4   3.6   7.0  -6.0         :   subdiag of A

  2.7
 -0.5
  2.6
  0.6
  2.7                           : vector b
```

10.3 Program Results

```
nag_dgtsv (f07cac) Example Program Results

Solution
    -4.0000
     7.0000
     3.0000
    -4.0000
    -3.0000
```