NAG Library Function Document
nag_dgbtrs (f07bec)

1 Purpose
nag_dgbtrs (f07bec) solves a real band system of linear equations with multiple right-hand sides,

\[ AX = B \quad \text{or} \quad A^T X = B, \]

where \( A \) has been factorized by nag_dgbtrf (f07bdc).

2 Specification
#include <nag.h>
#include <nagf07.h>
void nag_dgbtrs (Nag_OrderType order, Nag_TransType trans, Integer n,
   Integer kl, Integer ku, Integer nrhs, const double ab[], Integer pdab,
   const Integer ipiv[], double b[], Integer pdb, NagError *fail)

3 Description
nag_dgbtrs (f07bec) is used to solve a real band system of linear equations \( AX = B \) or \( A^T X = B \), the function must be preceded by a call to nag_dgbtrf (f07bdc) which computes the \( LU \) factorization of \( A \) as \( A = PLU \). The solution is computed by forward and backward substitution.

If \( \text{trans} = \text{Nag_NoTrans} \), the solution is computed by solving \( PLY = B \) and then \( UX = Y \).

If \( \text{trans} = \text{Nag_Trans} \) or \( \text{Nag_ConjTrans} \), the solution is computed by solving \( U^T Y = B \) and then \( L^T P^T X = Y \).

4 References

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: indicates the form of the equations.
   \( \text{trans} = \text{Nag_NoTrans} \)
   \( AX = B \) is solved for \( X \).
   \( \text{trans} = \text{Nag_Trans} \) or \( \text{Nag_ConjTrans} \)
   \( A^T X = B \) is solved for \( X \).
   Constraint: trans = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.
3: \[ n \text{ – Integer} \]  
On entry: \( n \), the order of the matrix \( A \).  
Constraint: \( n \geq 0 \).

4: \[ kl \text{ – Integer} \]  
On entry: \( k_l \), the number of subdiagonals within the band of the matrix \( A \).  
Constraint: \( kl \geq 0 \).

5: \[ ku \text{ – Integer} \]  
On entry: \( k_u \), the number of superdiagonals within the band of the matrix \( A \).  
Constraint: \( ku \geq 0 \).

6: \[ nrhs \text{ – Integer} \]  
On entry: \( r \), the number of right-hand sides.  
Constraint: \( nrhs \geq 0 \).

7: \[ ab[\text{dim}] \text{ – const double} \]  
Note: the dimension, \( \text{dim} \), of the array \( ab \) must be at least \( \max(1, pdab \times n) \).  
On entry: the \( LU \) factorization of \( A \), as returned by \text{nag_dgbtrf} (f07bdc).

8: \[ pdab \text{ – Integer} \]  
On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) of the matrix in the array \( ab \).  
Constraint: \( pdab \geq 2 \times kl + ku + 1 \).

9: \[ ipiv[\text{dim}] \text{ – const Integer} \]  
Note: the dimension, \( \text{dim} \), of the array \( ipiv \) must be at least \( \max(1, n) \).  
On entry: the pivot indices, as returned by \text{nag_dgbtrf} (f07bdc).

10: \[ b[\text{dim}] \text{ – double} \]  
Note: the dimension, \( \text{dim} \), of the array \( b \) must be at least \( \max(1, pdb \times nrhs) \) when \( \text{order} = \text{Nag\_ColMajor} \);  
\( \max(1, n \times pdb) \) when \( \text{order} = \text{Nag\_RowMajor} \).  
The \( (i,j) \text{th} \) element of the matrix \( B \) is stored in  
\( b[(j-1) \times pdb + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);  
\( b[(i-1) \times pdb + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).  
On entry: the \( n \) by \( r \) right-hand side matrix \( B \).  
On exit: the \( n \) by \( r \) solution matrix \( X \).

11: \[ pdb \text{ – Integer} \]  
On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( b \).  
Constraints:  
if \( \text{order} = \text{Nag\_ColMajor} \), \( pdb \geq \max(1, n) \);  
if \( \text{order} = \text{Nag\_RowMajor} \), \( pdb \geq \max(1, nrhs) \).
6 Error Indicators and Warnings

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

NE_BAD_PARAM
On entry, argument <value> had an illegal value.

NE_INT
On entry, \( kl = \langle value \rangle \).
Constraint: \( kl \geq 0 \).
On entry, \( ku = \langle value \rangle \).
Constraint: \( ku \geq 0 \).
On entry, \( n = \langle value \rangle \).
Constraint: \( n \geq 0 \).
On entry, \( nrhs = \langle value \rangle \).
Constraint: \( nrhs \geq 0 \).
On entry, \( pdab = \langle value \rangle \).
Constraint: \( pdab > 0 \).
On entry, \( pdb = \langle value \rangle \).
Constraint: \( pdb > 0 \).

NE_INT_2
On entry, \( pdb = \langle value \rangle \) and \( n = \langle value \rangle \).
Constraint: \( pdb \geq \max(1, n) \).
On entry, \( pdb = \langle value \rangle \) and \( nrhs = \langle value \rangle \).
Constraint: \( pdb \geq \max(1, nrhs) \).

NE_INT_3
On entry, \( pdab = \langle value \rangle \), \( kl = \langle value \rangle \) and \( ku = \langle value \rangle \).
Constraint: \( pdab \geq 2 \times kl + ku + 1 \).

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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
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.
7 Accuracy

For each right-hand side vector $b$, the computed solution $x$ is the exact solution of a perturbed system of equations $(A + E)x = b$, where

$$|E| \leq c(k)\epsilon P[L]|U|,$$

$c(k)$ is a modest linear function of $k = k_l + k_u + 1$, and $\epsilon$ is the machine precision. This assumes $k \ll n$.

If $\hat{x}$ is the true solution, then the computed solution $x$ satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq c(k)\text{cond}(A, x)\epsilon,$$

where $\text{cond}(A, x) = \|A^{-1}\|\|A\|\|x\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \|A^{-1}\|\|A\|\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A, x)$ can be much smaller than $\text{cond}(A)$, and $\text{cond}(A^T)$ can be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling nag_dgbrfs (f07bhc), and an estimate for $\kappa_\infty(A)$ can be obtained by calling nag_dgbcon (f07bgc) with norm = Nag_InfNorm.

8 Parallelism and Performance

nag_dgbtrs (f07bec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dgbtrs (f07bec) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

The total number of floating-point operations is approximately $2n(2k_l + k_u)r$, assuming $n \gg k_l$ and $n \gg k_u$.

This function may be followed by a call to nag_dgbrfs (f07bhc) to refine the solution and return an error estimate.

The complex analogue of this function is nag_zgbtrs (f07bsc).

10 Example

This example solves the system of equations $AX = B$, where

$$A = \begin{pmatrix} -0.23 & 2.54 & -3.66 & 0.00 \\ -6.98 & 2.64 & -2.73 & -2.13 \\ 0.00 & 2.56 & 2.46 & 4.07 \\ 0.00 & 0.00 & -4.78 & -3.82 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 4.42 & -36.01 \\ 27.13 & -31.67 \\ -6.14 & -1.16 \\ 10.50 & -25.82 \end{pmatrix}.$$  

Here $A$ is nonsymmetric and is treated as a band matrix, which must first be factorized by nag_dgbtrf (f07bdc).
10.1 Program Text

/* nag_dgbtrs (f07bec) Example Program.  
 * Copyright 2014 Numerical Algorithms Group.  
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, ipiv_len, j, kl, ku, n, nrhs, pdab, pdb;
    Integer exit_status = 0;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    double *ab = 0, *b = 0;
    Integer *ipiv = 0;
    #ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker  
     * to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */
    
    #ifdef NAG_COLUMN_MAJOR
    #define AB(I, J) ab[(J-1)*pdab + kl + ku + I - J]
    #define B(I, J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
    #else
    #define AB(I, J) ab[(I-1)*pdab + kl + J - I]
    #define B(I, J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    printf("nag_dgbtrs (f07bec) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%\n");
    #else
    scanf("%\n");
    #endif
    #ifdef _WIN32
    scanf_s("%d%d%d%d%\n", &n, &nrhs, &kl, &ku);
    #else
    scanf("%d%d%d%d%\n", &n, &nrhs, &kl, &ku);
    #endif
    ipiv_len = n;
    #ifdef NAG_COLUMN_MAJOR
    pdab = 2*kl + ku + 1;
    pdb = n;
    #else
    pdab = 2*kl + ku + 1;
    pdb = nrhs;
    #endif

    /* Allocate memory */
    if (!ab = NAG_ALLOC((2*kl+ku+1) * n, double)) ||
        !b = NAG_ALLOC(nrhs * n, double)) ||
!(ipiv = NAG_ALLOC(ipiv_len, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
for (i = 1; i <= n; ++i)
{
    for (j = MAX(i-kl, 1); j <= MIN(i+ku, n); ++j)
        #ifdef _WIN32
            scanf_s("%lf", &AB(i, j));
        #else
            scanf("%lf", &AB(i, j));
        #endif
    }
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif

/* Read B from data file */
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        #ifdef _WIN32
            scanf_s("%lf", &B(i, j));
        #else
            scanf("%lf", &B(i, j));
        #endif
    }
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif

/* Factorize A */
/* nag_dgbtrf (f07bdc).
   * LU factorization of real m by n band matrix
*/
nag_dgbtrf(order, n, n, kl, ku, ab, pdab, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    print("Error from nag_dgbtrf (f07bdc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute solution */
/* nag_dgbtrs (f07bec).
   * Solution of real band system of linear equations,
   * multiple right-hand sides, matrix already factorized by
   * nag_dgbtrf (f07bdc)
*/
nag_dgbtrs(order, Nag_NoTrans, n, kl, ku, nrhs, ab, pdab, ipiv,
    b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    print("Error from nag_dgbtrs (f07bec).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print solution */
/* nag_gen_real_mat_print (x04cac).
   * Print real general matrix (easy-to-use)
*/
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb, "Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ab);
NAG_FREE(b);
NAG_FREE(ipiv);
return exit_status;
}

10.2 Program Data
nag_dgbtrs (f07bec) Example Program Data
4 2 1 2 :Values of N, NRHS, KL and KU
-0.23  2.54 -3.66
-6.98  2.46 -2.73 -2.13
  2.56  2.46  4.07       -4.78 -3.82 :End of matrix A
  4.42 -36.01
 27.13 -31.67
-6.14  -1.16
10.50 -25.82 :End of matrix B

10.3 Program Results
nag_dgbtrs (f07bec) Example Program Results

Solution(s)

1  2
1  -2.0000  1.0000
2   3.0000 -4.0000
3   1.0000  7.0000
4  -4.0000 -2.0000