

# NAG Library Function Document

## nag\_dgetrs (f07aec)

### 1 Purpose

nag\_dgetrs (f07aec) solves a real 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\_dgetrf (f07adc).

### 2 Specification

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

void nag_dgetrs (Nag_OrderType order, Nag_TransType trans, Integer n,
                 Integer nrhs, const double a[], Integer pda, const Integer ipiv[],
                 double b[], Integer pdb, NagError *fail)
```

### 3 Description

nag\_dgetrs (f07aec) is used to solve a real system of linear equations  $AX = B$  or  $A^T X = B$ , the function must be preceded by a call to nag\_dgetrf (f07adc) which computes the  $LU$  factorization of  $A$  as  $A = PLU$ . The solution is computed by forward and backward substitution.

If **trans** = Nag\_NoTrans, the solution is computed by solving  $PLY = B$  and then  $UX = Y$ .

If **trans** = Nag\_Trans or Nag\_ConjTrans, the solution is computed by solving  $U^T Y = B$  and then  $L^T P^T X = Y$ .

### 4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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.

**trans** = Nag\_NoTrans  
 $AX = B$  is solved for  $X$ .

**trans** = Nag\_Trans or Nag\_ConjTrans  
 $A^T X = B$  is solved for  $X$ .

*Constraint:* **trans** = Nag\_NoTrans, Nag\_Trans or Nag\_ConjTrans.

- 3: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $n \geq 0$ .
- 4: **nrhs** – Integer *Input*  
*On entry:*  $r$ , the number of right-hand sides.  
*Constraint:* **nrhs**  $\geq 0$ .
- 5: **a**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **a** must be at least  $\max(1, \mathbf{pda} \times \mathbf{n})$ .  
The  $(i, j)$ th element of the matrix  $A$  is stored in  
 $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$  when **order** = Nag\_RowMajor.  
*On entry:* the  $LU$  factorization of  $A$ , as returned by nag\_dgetrf (f07adc).
- 6: **pda** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **a**.  
*Constraint:* **pda**  $\geq \max(1, \mathbf{n})$ .
- 7: **ipiv**[*dim*] – const Integer *Input*  
**Note:** the dimension, *dim*, of the array **ipiv** must be at least  $\max(1, \mathbf{n})$ .  
*On entry:* the pivot indices, as returned by nag\_dgetrf (f07adc).
- 8: **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:* the  $n$  by  $r$  solution matrix  $X$ .
- 9: **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, **pdb**  $\geq \max(1, \mathbf{n})$ ;  
if **order** = Nag\_RowMajor, **pdb**  $\geq \max(1, \mathbf{nrhs})$ .
- 10: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

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

### NE\_INT

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

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

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

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

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

Constraint:  $\mathbf{pda} > 0$ .

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

Constraint:  $\mathbf{pdb} > 0$ .

### NE\_INT\_2

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

Constraint:  $\mathbf{pda} \geq \max(1, \mathbf{n})$ .

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

Constraint:  $\mathbf{pdb} \geq \max(1, \mathbf{n})$ .

On entry,  $\mathbf{pdb} = \langle value \rangle$  and  $\mathbf{nrhs} = \langle value \rangle$ .

Constraint:  $\mathbf{pdb} \geq \max(1, \mathbf{nrhs})$ .

### 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.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

### NE\_NO\_LICENCE

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation 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

$$|E| \leq c(n)\epsilon P|L||U|,$$

$c(n)$  is a modest linear function of  $n$ , and  $\epsilon$  is the *machine precision*.

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

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

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_dgerfs` (f07ahc), and an estimate for  $\kappa_\infty(A)$  can be obtained by calling `nag_dgecon` (f07agc) with `norm = Nag_InfNorm`.

## 8 Parallelism and Performance

`nag_dgetrs` (f07aec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_dgetrs` (f07aec) 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  $2n^2r$ .

This function may be followed by a call to `nag_dgerfs` (f07ahc) to refine the solution and return an error estimate.

The complex analogue of this function is `nag_zgetrs` (f07asc).

## 10 Example

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

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 9.52 & 18.47 \\ 24.35 & 2.25 \\ 0.77 & -13.28 \\ -6.22 & -6.21 \end{pmatrix}.$$

Here  $A$  is nonsymmetric and must first be factorized by `nag_dgetrf` (f07adc).

### 10.1 Program Text

```
/* nag_dgetrs (f07aec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

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

    /* Arrays */
```

```

    double *a = 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 A(I, J) a[(J-1)*pda + I - 1]
#define B(I, J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
#else
#define A(I, J) a[(I-1)*pda + J - 1]
#define B(I, J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);

    printf("nag_dgetrs (f07aec) Example Program Results\n\n");

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &nrhs);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &nrhs);
#endif

    /* Allocate memory */
    if (!(a = NAG_ALLOC(n * n, double)) ||
        !(b = NAG_ALLOC(n * nrhs, double)) || !(ipiv = NAG_ALLOC(n, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
#ifdef NAG_COLUMN_MAJOR
    pda = n;
    pdb = n;
#else
    pda = n;
    pdb = nrhs;
#endif

    /* Read A and B from data file */
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j)
#ifdef _WIN32
            scanf_s("%lf", &A(i, j));
#else
            scanf("%lf", &A(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
    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_dgetrf (f07adc).
 * LU factorization of real m by n matrix
 */
nag_dgetrf(order, n, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgetrf (f07adc).\\n%s\\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute solution */
/* nag_dgetrs (f07aec).
 * Solution of real system of linear equations, multiple
 * right-hand sides, matrix already factorized by nag_dgetrf
 * (f07adc)
 */
nag_dgetrs(order, Nag_NoTrans, n, nrhs, a, pda, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dgetrs (f07aec).\\n%s\\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(a);
    NAG_FREE(b);
    NAG_FREE(ipiv);
    return exit_status;
}

```

## 10.2 Program Data

```

nag_dgetrs (f07aec) Example Program Data
  4  2                               :Values of N and NRHS
  1.80  2.88  2.05 -0.89
  5.25 -2.95 -0.95 -3.80
  1.58 -2.69 -2.90 -1.04
-1.11 -0.66 -0.59  0.80      :End of matrix A
  9.52 18.47
 24.35  2.25
  0.77 -13.28
-6.22 -6.21                  :End of matrix B

```

### 10.3 Program Results

nag\_dgetrs (f07aec) Example Program Results

Solution(s)

	1	2
1	1.0000	3.0000
2	-1.0000	2.0000
3	3.0000	4.0000
4	-5.0000	1.0000

---