NAG Library Function Document

nag_dsysv (f07mac)

1 Purpose

nag_dsysv (f07mac) computes the solution to a real system of linear equations

\[ AX = B, \]

where \( A \) is an \( n \) by \( n \) symmetric matrix and \( X \) and \( B \) are \( n \) by \( r \) matrices.

2 Specification

```c
#include <nag.h>
#include <nagf07.h>
void nag_dsysv (Nag_OrderType order, Nag_UploType uplo, Integer n, 
    Integer nrhs, double a[], Integer pda, Integer ipiv[], double b[], 
    Integer pdb, NagError *fail)
```

3 Description

nag_dsysv (f07mac) uses the diagonal pivoting method to factor \( A \) as

\[
\begin{align*}
& \text{order} = \text{Nag_ColMajor} & & \text{uplo} = \text{Nag_Upper} & & A = UDU^T \\
& \text{order} = \text{Nag_ColMajor} & & \text{uplo} = \text{Nag_Lower} & & A = LDL^T \\
& \text{order} = \text{Nag_RowMajor} & & \text{uplo} = \text{Nag_Upper} & & A = UD^T \\
& \text{order} = \text{Nag_RowMajor} & & \text{uplo} = \text{Nag_Lower} & & A = L^TDL
\end{align*}
\]

where \( U \) (or \( L \)) is a product of permutation and unit upper (lower) triangular matrices, and \( D \) is symmetric and block diagonal with 1 by 1 and 2 by 2 diagonal blocks. The factored form of \( A \) is then used to solve the system of equations \( AX = B \).

Note that, in general, different permutations (pivot sequences) and diagonal block structures are obtained for \( \text{uplo} = \text{Nag_Upper} \) or \( \text{Nag_Lower} \).

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

\( \text{On entry: the order argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.} \)

\( \text{Constraint: order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).
2: \texttt{uplo} – Nag_UploType \hspace{1em} Input

On entry: if \texttt{uplo} = Nag_Upper, the upper triangle of \text{A} is stored.
If \texttt{uplo} = Nag_Lower, the lower triangle of \text{A} is stored.

Constraint: \texttt{uplo} = Nag_Upper or Nag_Lower.

3: \textbf{n} – Integer \hspace{1em} Input

On entry: \textit{n}, the number of linear equations, i.e., the order of the matrix \text{A}.

Constraint: \textit{n} \geq 0.

4: \textbf{nrhs} – Integer \hspace{1em} Input

On entry: \textit{r}, the number of right-hand sides, i.e., the number of columns of the matrix \text{B}.

Constraint: \textbf{nrhs} \geq 0.

5: \textit{a}[\textit{dim}] – double \hspace{1em} Input/Output

Note: the dimension, \textit{dim}, of the array \textit{a} must be at least \text{max}(1, \text{pda} \times \textit{n}).

On entry: the \textit{n} by \textit{n} symmetric matrix \text{A}.

If \texttt{order} = Nag_ColMajor, \textit{A}_{ij} is stored in \textit{a}[(\textit{j} - 1) \times \text{pda} + i - 1].

If \texttt{order} = Nag_RowMajor, \textit{A}_{ij} is stored in \textit{a}[(\textit{i} - 1) \times \text{pda} + \textit{j} - 1].

If \texttt{uplo} = Nag_Upper, the upper triangular part of \text{A} must be stored and the elements of the array below the diagonal are not referenced.

If \texttt{uplo} = Nag_Lower, the lower triangular part of \text{A} must be stored and the elements of the array above the diagonal are not referenced.

On exit: if \texttt{fail.code} = NE_NOERROR, the block diagonal matrix \text{D} and the multipliers used to obtain the factor \text{U} or \text{L} from the factorization \text{A} = \text{UDU}^T, \text{A} = \text{LDL}^T, \text{A} = \text{U}^T \text{DU} \text{ or } \text{A} = \text{L}^T \text{DL} as computed by \text{nag_dsytrf (f07mdc)}.

6: \textbf{pda} – Integer \hspace{1em} Input

On entry: the stride separating row or column elements (depending on the value of \texttt{order}) of the matrix \text{A} in the array \textit{a}.

Constraint: \textbf{pda} \geq \text{max}(1, \textit{n}).

7: \textit{ipiv}[\textit{dim}] – Integer \hspace{1em} Output

Note: the dimension, \textit{dim}, of the array \textit{ipiv} must be at least \text{max}(1, \textit{n}).

On exit: details of the interchanges and the block structure of \text{D}. More precisely,

if \textit{ipiv}[i - 1] = k > 0, \textit{d}_i is a 1 by 1 pivot block and the \textit{i}th row and column of \text{A} were interchanged with the \textit{k}th row and column;

if \texttt{uplo} = Nag_Upper and \textit{ipiv}[i - 2] = \textit{ipiv}[i - 1] = -l < 0, \begin{pmatrix} \textit{d}_{i-1,i-1} & \textit{d}_{i,i-1} \\ \textit{d}_{i,i-1} & \textit{d}_{i,i} \end{pmatrix} is a 2 by 2 pivot block and the \textit{(i - 1)}th row and column of \text{A} were interchanged with the \textit{l}th row and column;

if \texttt{uplo} = Nag_Lower and \textit{ipiv}[i - 1] = \textit{ipiv}[i] = -m < 0, \begin{pmatrix} \textit{d}_{i,i} & \textit{d}_{i+1,i} \\ \textit{d}_{i+1,i} & \textit{d}_{i+1,i+1} \end{pmatrix} is a 2 by 2 pivot block and the \textit{(i + 1)}th row and column of \text{A} were interchanged with the \textit{m}th row and column.
8: \( \mathbf{b}[\text{dim}] \) – double

\textbf{Note:} the dimension, \( \text{dim} \), of the array \( \mathbf{b} \) must be at least
\[ \max(1, \text{pdb} \times \text{nrhs}) \text{ when order = Nag\_ColMajor; } \]
\[ \max(1, n \times \text{pdb}) \text{ when order = Nag\_RowMajor. } \]

The \((i,j)\)th element of the matrix \( \mathbf{B} \) is stored in
\[ \mathbf{b}[(j - 1) \times \text{pdb} + i - 1] \text{ when order = Nag\_ColMajor; } \]
\[ \mathbf{b}[(i - 1) \times \text{pdb} + j - 1] \text{ when order = Nag\_RowMajor. } \]

\textit{On entry:} the \( n \) by \( r \) right-hand side matrix \( \mathbf{B} \).

\textit{On exit:} if \( \text{fail\_code} = \text{NE\_NOERROR} \), the \( n \) by \( r \) solution matrix \( \mathbf{X} \).

9: \( \text{pdb} \) – Integer

\textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \text{order}) in the array \( \mathbf{b} \).

\textit{Constraints:}
- if \( \text{order} = \text{Nag\_ColMajor} \), \( \text{pdb} \geq \max(1, n) \);
- if \( \text{order} = \text{Nag\_RowMajor} \), \( \text{pdb} \geq \max(1, \text{nrhs}) \).

10: \( \text{fail} \) – NagError *

\textit{Input/Output}

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

6 \hspace{10pt} \textbf{Error Indicators and Warnings}

\textbf{NE\_ALLOC\_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE\_BAD\_PARAM}

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\textbf{NE\_INT}

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{nrhs} = \langle \text{value} \rangle \).

Constraint: \( \text{nrhs} \geq 0 \).

On entry, \( \text{pda} = \langle \text{value} \rangle \).

Constraint: \( \text{pda} > 0 \).

On entry, \( \text{pdb} = \langle \text{value} \rangle \).

Constraint: \( \text{pdb} > 0 \).

\textbf{NE\_INT\_2}

On entry, \( \text{pda} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

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

On entry, \( \text{pdb} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

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

On entry, \( \text{pdb} = \langle \text{value} \rangle \) and \( \text{nrhs} = \langle \text{value} \rangle \).

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

NE_SINGULAR
   Element (value) of the diagonal is exactly zero. The factorization has been completed, but the
   block diagonal matrix $D$ is exactly singular, so the solution could not be computed.

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 $\epsilon$ is the machine precision. An approximate error bound for the computed solution is given by
   $$\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq \kappa(A)\|E\|_1\|A\|_1,$$
where $\kappa(A) = \|A^{-1}\|_1\|A\|_1$, 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.

nag_dsysvx (f07mbc) is a comprehensive LAPACK driver that returns forward and backward error
   bounds and an estimate of the condition number. Alternatively, nag_real_sym_lin_solve (f04bhc) solves
   $Ax = b$ and returns a forward error bound and condition estimate. nag_real_sym_lin_solve (f04bhc) calls
   nag_dsysv (f07mac) to solve the equations.

8  Parallelism and Performance
nag_dsysv (f07mac) is not threaded by NAG in any implementation.

nag_dsysv (f07mac) 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 $\frac{1}{3}n^3 + 2n^2r$, where $r$ is the number of
   right-hand sides.

The complex analogues of nag_dsysv (f07mac) are nag_zhesv (f07mnc) for Hermitian matrices, and
   nag_zsysv (f07nnc) for symmetric matrices.
10  Example

This example solves the equations

\[ Ax = b, \]

where \( A \) is the symmetric matrix

\[
A = \begin{pmatrix}
-1.81 & 2.06 & 0.63 & -1.15 \\
2.06 & 1.15 & 1.87 & 4.20 \\
0.63 & 1.87 & -0.21 & 3.87 \\
-1.15 & 4.20 & 3.87 & 2.07
\end{pmatrix}
\]

and \( b = \begin{pmatrix} 0.96 \\ 6.07 \\ 8.38 \\ 9.50 \end{pmatrix} \).

Details of the factorization of \( A \) are also output.

10.1  Program Text

/* nag_dsysv (f07mac) Example Program. */
* Copyright 2014 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, pda, pdb;

    /* Arrays */
    double *a = 0, *b = 0;
    Integer *ipiv = 0;
    char nag_enum_arg[40];

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

    #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_dsysv (f07mac) 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

Mark 25
scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n"]", &n, &nrhs);
#endif
if (n < 0 || nrhs < 0)
{
    printf("Invalid n or nrhs\n");
    exit_status = 1;
    goto END;
}
#ifdef _WIN32
    scanf_s(" %39s%*[\n"]", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n"]", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).  
* Converts NAG enum member name to value 
*/
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
/* 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;
}
pda = n;
#ifdef NAG_COLUMN_MAJOR
    pdb = n;
#else
    pdb = nrhs;
#endif
/* Read the triangular part of the matrix A from data file */
if (uplo == Nag_Upper)
    for (i = 1; i <= n; ++i)
        #ifdef _WIN32
            for (j = i; j <= n; ++j) scanf_s("%lf", &A(i, j));
        #else
            for (j = i; j <= n; ++j) scanf("%lf", &A(i, j));
        #endif
else
    #ifdef _WIN32
        for (j = 1; j <= i; ++j) scanf_s("%lf", &A(i, j));
    #else
        for (j = 1; j <= i; ++j) scanf("%lf", &A(i, j));
    #endif
        #ifdef _WIN32
              scanf_s("%*[\n"]");
        #else
              scanf("%*[\n"]");
        #endif
/* Read b from data file */
for (i = 1; i <= n; ++i)
    #ifdef _WIN32
        for (j = 1; j <= nrhs; ++j) scanf_s("%lf", &B(i, j));
    #else
        for (j = 1; j <= nrhs; ++j) scanf("%lf", &B(i, j));
    #endif
        #ifdef _WIN32
              scanf_s("%*[\n"]");
        #else
              scanf("%*[\n"]");
        #endif
/* Solve the equations Ax = b for x using nag_dsysv (f07mac). */
    nag_dsysv(order, uplo, n, nrhs, a, pda, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsysv (f07mac)\n\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(" %10.4f%s", B(i, j), j%7 == 0?"\n":"\n");
    printf("\n");
}

END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(ipiv);
return exit_status;
}

10.2 Program Data
nag_dsysv (f07mac) Example Program Data
               4  l : n, nrhs
     Nag_Lower : uplo
-1.81
  2.06  1.15
  0.63  1.87  -0.21
-1.15  4.20  3.87  2.07 : matrix A
  0.96  6.07  8.38  9.50 : vector b

10.3 Program Results
nag_dsysv (f07mac) Example Program Results

Solution
-5.0000
-2.0000
  1.0000
  4.0000