NAG Library Function Document

nag_dtfsm (f16ylc)

1 Purpose

nag_dtfsm (f16ylc) performs one of the matrix-matrix operations

\[ B \leftarrow \alpha A^{-1}B, \quad B \leftarrow \alpha A^{-T}B, \]
\[ B \leftarrow \alpha BA^{-1} \quad \text{or} \quad B \leftarrow \alpha BA^{-T}, \]

where \( A \) is a real triangular matrix stored in Rectangular Full Packed (RFP) format, \( B \) is an \( m \) by \( n \) real matrix, and \( \alpha \) is a real scalar. \( A^{-T} \) denotes \((A^T)^{-1}\) or equivalently \((A^{-1})^T\).

No test for singularity or near-singularity of \( A \) is included in this function. Such tests must be performed before calling this function.

2 Specification

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

void nag_dtfsm (Nag_OrderType order, Nag_RFP_Store transr,
               Nag_SideType side, Nag_UploType uplo, Nag_TransType trans,
               Nag_DiagType diag, Integer m, Integer n, double alpha,
               const double ar[], double b[], Integer pdb, NagError *fail)
```

3 Description

nag_dtfsm (f16ylc) solves (for \( X \)) a triangular linear system of one of the forms

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

where \( A \) is a real triangular matrix stored in RFP format, \( B, X \) are \( m \) by \( n \) real matrices, and \( \alpha \) is a real scalar. The RFP storage format is described in Section 3.3.3 in the f07 Chapter Introduction.

4 References


5 Arguments

1: \textbf{order} – Nag_OrderType \hspace{1cm} \textit{Input}

\textit{On entry}: the \textbf{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 \textbf{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint}: \textbf{order} = Nag_RowMajor or Nag_ColMajor.

2: \textbf{transr} – Nag_RFP_Store \hspace{1cm} \textit{Input}

\textit{On entry}: specifies whether the RFP representation of \( A \) is normal or transposed.

\textbf{transr} = Nag_RFP_Normal

The matrix \( A \) is stored in normal RFP format.
\texttt{transr} = \texttt{Nag\_RFP\_Trans}

The matrix \( A \) is stored in transposed RFP format.

\textit{Constraint:} \( \texttt{transr} = \texttt{Nag\_RFP\_Normal} \) or \( \texttt{Nag\_RFP\_Trans} \).

3: \texttt{side} – \texttt{Nag\_SideType} \hspace{1cm} \textit{Input}

\textit{On entry:} specifies whether \( B \) is operated on from the left or the right, or similarly whether \( A \) (or its transpose) appears to the left or right of the solution matrix in the linear system to be solved.

\texttt{side} = \texttt{Nag\_LeftSide}

\( B \) is pre-multiplied from the left. The system to be solved has the form \( AX = \alpha B \) or \( A^TX = \alpha B \).

\texttt{side} = \texttt{Nag\_RightSide}

\( B \) is post-multiplied from the right. The system to be solved has the form \( XA = \alpha B \) or \( XA^T = \alpha B \).

\textit{Constraint:} \( \texttt{side} = \texttt{Nag\_LeftSide} \) or \( \texttt{Nag\_RightSide} \).

4: \texttt{uplo} – \texttt{Nag\_UploType} \hspace{1cm} \textit{Input}

\textit{On entry:} specifies whether \( A \) is upper or lower triangular.

\texttt{uplo} = \texttt{Nag\_Upper}

\( A \) is upper triangular.

\texttt{uplo} = \texttt{Nag\_Lower}

\( A \) is lower triangular.

\textit{Constraint:} \( \texttt{uplo} = \texttt{Nag\_Upper} \) or \( \texttt{Nag\_Lower} \).

5: \texttt{trans} – \texttt{Nag\_TransType} \hspace{1cm} \textit{Input}

\textit{On entry:} specifies whether the operation involves \( A^{-1} \) or \( A^{-T} \), i.e., whether or not \( A \) is transposed in the linear system to be solved.

\texttt{trans} = \texttt{Nag\_NoTrans}

The operation involves \( A^{-1} \), i.e., \( A \) is not transposed.

\texttt{trans} = \texttt{Nag\_Trans}

The operation involves \( A^{-T} \), i.e., \( A \) is transposed.

\textit{Constraint:} \( \texttt{trans} = \texttt{Nag\_NoTrans} \) or \( \texttt{Nag\_Trans} \).

6: \texttt{diag} – \texttt{Nag\_DiagType} \hspace{1cm} \textit{Input}

\textit{On entry:} specifies whether \( A \) has nonunit or unit diagonal elements.

\texttt{diag} = \texttt{Nag\_NonUnitDiag}

The diagonal elements of \( A \) are stored explicitly.

\texttt{diag} = \texttt{Nag\_UnitDiag}

The diagonal elements of \( A \) are assumed to be 1, the corresponding elements of \texttt{ar} are not referenced.

\textit{Constraint:} \( \texttt{diag} = \texttt{Nag\_NonUnitDiag} \) or \( \texttt{Nag\_UnitDiag} \).

7: \texttt{m} – \texttt{Integer} \hspace{1cm} \textit{Input}

\textit{On entry:} \( m \), the number of rows of the matrix \( B \).

\textit{Constraint:} \( m \geq 0 \).
8:  n – Integer

   On entry: n, the number of columns of the matrix B.
   Constraint: n \geq 0.

9:  alpha – double

   On entry: the scalar \alpha.

10:  ar[dim] – const double

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

   \max(1, m \times (m + 1)/2) when side = Nag_LeftSide;
   \max(1, n \times (n + 1)/2) when side = Nag_RightSide.

   On entry: the m by m triangular matrix A if side = Nag_LeftSide or the n by n triangular matrix
   A if side = Nag_RightSide, stored in RFP format (as specified by transr). The storage format is
described in detail in Section 3.3.3 in the f07 Chapter Introduction. If alpha = 0.0, ar is not
referenced.

11:  b[dim] – double

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

   \max(1, pdb \times n) when order = Nag_ColMajor;
   \max(1, m \times pdb) when order = Nag_RowMajor.

   On entry: the m by n matrix B.

   If alpha = 0.0, b need not be set.

   On exit: the updated matrix B, or similarly the solution matrix X.

   If order = Nag_ColMajor, B_{ij} is stored in b[(j - 1) \times pdb + i - 1].

   If order = Nag_RowMajor, B_{ij} is stored in b[(i - 1) \times pdb + j - 1].

12:  pdb – Integer

   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, m);
   if order = Nag_RowMajor, pdb \geq \max(1, n).

13:  fail – NagError *

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

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 \langle value \rangle had an illegal value.
NE_INT
On entry, $m = \langle value \rangle$.
Constraint: $m \geq 0$.
On entry, $n = \langle value \rangle$.
Constraint: $n \geq 0$.

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

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
Not applicable.

8 Parallelism and Performance
nag_dtfsm (f16ylc) is not threaded by NAG in any implementation.

nag_dtfsm (f16ylc) 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
None.

10 Example
This example reads in the lower triangular part of a symmetric matrix $A$ which it converts to RFP
format. It also reads in $\alpha$ and a 6 by 4 matrix $B$ and then performs the matrix-matrix operation
$B \leftarrow \alpha A^{-1} B$. 
10.1 Program Text
/* nag_dtfsm (f16ylc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 25, 2014.
*/
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <naqf16.h>
#include <nagx04.h>

int main(void)
{ /* Scalars */
    Integer exit_status = 0;
    double alpha;
    Integer i, j, m, n, pda, pdb;
    /* Arrays */
    double *a = 0, *ar = 0, *b = 0;
    char nag_enum_arg[40];
    /* Nag Types */
    Nag_OrderType order;
    Nag_RFP_Store transr;
    Nag_SideType side;
    Nag_UploType uplo;
    Nag_TransType trans;
    NagError fail;
    INIT_FAIL(fail);

    printf("nag_dtfsm (f16ylc) Example Program Results\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
    scanf("%"NAG_IFMT "%"NAG_IFMT "%*[\n] ", &m, &n);
    #else
    scanf("%"NAG_IFMT "%"NAG_IFMT "%*[\n] ", &m, &n);
    #endif
    pda = m;
    #ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
    pdb = m;
    #define A(I, J) a[(J-1)*pda + I-1]
    #define B(I, J) b[(J-1)*pdb + I-1]
    #else
    order = Nag_RowMajor;
    pdb = n;
    #define A(I, J) a[(I-1)*pda + J-1]
    #define B(I, J) b[(I-1)*pdb + J-1]
    #endif
    if (!(a = NAG_ALLOC(pda*m, double)) ||
        !(ar = NAG_ALLOC((m * (m + 1))/2, double)) ||
        !(b = NAG_ALLOC(m*n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    /* Nag_RFP_Store */
    #ifdef _WIN32
f16 – NAG Interface to BLAS
f16ylc.5
Mark 25
f16ylc
```c
    scanf_s("%39s ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s ", nag_enum_arg);
#endif
transr = (Nag_RFP_Store) nag_enum_name_to_value (nag_enum_arg);
/* Nag_SideType */
#ifdef _WIN32
    scanf_s("%39s %[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s %[\n] ", nag_enum_arg);
#endif
side = (Nag_SideType) nag_enum_name_to_value (nag_enum_arg);
/* Nag_UploType */
#ifdef _WIN32
    scanf_s("%39s %*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s %*[\n] ", nag_enum_arg);
#endif
uplo = (Nag_UploType) nag_enum_name_to_value (nag_enum_arg);
/* Nag_TransType */
#ifdef _WIN32
    scanf_s("%39s %*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s %*[\n] ", nag_enum_arg);
#endif
trans = (Nag_TransType) nag_enum_name_to_value (nag_enum_arg);
#ifdef _WIN32
    scanf_s("%lf%*[\n] ", &alpha);
#else
    scanf("%lf%*[\n] ", &alpha);
#endif
    /* Read upper or lower triangle of matrix A from data file */
    if (uplo == Nag_Lower) {
        for (i = 1; i <= m; i++) {
            for (j = 1; j <= i; j++) {
                #ifdef _WIN32
                    scanf_s("%lf", &A(i, j));
                #else
                    scanf("%lf", &A(i, j));
                #endif
            }
        }
    } else {
        for (i = 1; i <= m; i++) {
            for (j = i; j <= m; j++) {
                #ifdef _WIN32
                    scanf_s("%lf", &A(i, j));
                #else
                    scanf("%lf", &A(i, j));
                #endif
            }
        }
    }
    /* Convert real triangular matrix A from full to rectangular full packed
```

---

```
```
* storage format (stored in ar) using nag_dtrttf (f01vec).
*/
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dtrttf (f01vec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n");

nag_dtfsm(order, transr, side, uplo, trans, Nag_NonUnitDiag, m, n, alpha, ar, b, pdb, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dtfsm (f16ylc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
NAG_FREE(a);
NAG_FREE(ar);
NAG_FREE(b);
return exit_status;

10.2 Program Data

nag_dtfsm (f16ylc) Example Program Data
6 4 1.0
Nag_RFP_Normal Nag_LeftSide : transr, side
Nag_Lower Nag_NoTrans : uplo, trans
4.21 : alpha

1.0
2.0 2.0 4.0 4.0 5.0 5.0 6.0
3.0 3.0 3.0 4.0 5.0 5.0 6.0
4.0 4.0 4.0 5.0 6.0 6.0
5.0 5.0 5.0 6.0 6.0 6.0 6.0 : matrix A
6.0
3.22 1.37 2.31 0.29
1.64 1.80 0.38 -1.52
1.87 2.87 2.02 -0.80
5.20 -2.99 -0.91 -3.87
1.83 -2.71 -2.81 -1.13
-1.10 -0.63 -0.50 0.81 : matrix B
### 10.3 Program Results

*nag_dtfsm (f16ylc)* Example Program Results

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.5562</td>
<td>5.7677</td>
<td>9.7251</td>
<td>1.2209</td>
</tr>
<tr>
<td>2</td>
<td>-10.1040</td>
<td>-1.9787</td>
<td>-8.9252</td>
<td>-4.4205</td>
</tr>
<tr>
<td>3</td>
<td>-0.8280</td>
<td>0.2386</td>
<td>2.0348</td>
<td>2.0769</td>
</tr>
<tr>
<td>4</td>
<td>2.8488</td>
<td>-7.1745</td>
<td>-3.7925</td>
<td>-2.9505</td>
</tr>
<tr>
<td>5</td>
<td>-3.9321</td>
<td>0.8652</td>
<td>-1.4082</td>
<td>3.1217</td>
</tr>
<tr>
<td>6</td>
<td>-2.3127</td>
<td>1.8398</td>
<td>2.0152</td>
<td>1.5198</td>
</tr>
</tbody>
</table>