nag_dgetrf (f07adc) forms the LU factorization of a real \( m \) by \( n \) matrix \( A \) as \( A = PLU \), where \( P \) is a permutation matrix, \( L \) is lower triangular with unit diagonal elements (lower trapezoidal if \( m > n \)) and \( U \) is upper triangular (upper trapezoidal if \( m < n \)). Usually \( A \) is square \( (m = n) \), and both \( L \) and \( U \) are triangular. The function uses partial pivoting, with row interchanges.

### References

### 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. **m** – Integer
   
   *Input*
   
   *On entry:* \( m \), the number of rows of the matrix \( A \).

   *Constraint:* \( m \geq 0 \).

3. **n** – Integer
   
   *Input*
   
   *On entry:* \( n \), the number of columns of the matrix \( A \).

   *Constraint:* \( n \geq 0 \).

4. **a[dim]** – double
   
   *Input/Output*

   *Note:* the dimension, \( dim \), of the array \( a \) must be at least
   \[
   \max(1, pda \times n) \text{ when } order = \text{Nag_ColMajor}; \\
   \max(1, m \times pda) \text{ when } order = \text{Nag_RowMajor}.
   \]
The \((i,j)\)th element of the matrix \(A\) is stored in

\[
\begin{align*}
\text{a}\left[(j-1) \times \text{pda} + i - 1\right] & \text{ when order = Nag.ColMajor;} \\
\text{a}\left[(i-1) \times \text{pda} + j - 1\right] & \text{ when order = Nag.RowMajor.}
\end{align*}
\]

On entry: the \(m\) by \(n\) matrix \(A\).

On exit: the factors \(L\) and \(U\) from the factorization \(A = PLU\); the unit diagonal elements of \(L\) are not stored.

5: \textbf{pda} – Integer \hspace{1cm} \textit{Input}

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

Constraints:

- if \textbf{order} = Nag.ColMajor, \textbf{pda} \geq \max(1, m);
- if \textbf{order} = Nag.RowMajor, \textbf{pda} \geq \max(1, n).

6: \textbf{ipiv} [\min(m, n)] – Integer \hspace{1cm} \textit{Output}

On exit: the pivot indices that define the permutation matrix. At the \(i\)th step, if \textbf{ipiv}\[i - 1\] > \(i\) then row \(i\) of the matrix \(A\) was interchanged with row \textbf{ipiv}\[i - 1\], for \(i = 1, 2, \ldots, \min(m, n)\).

\textbf{ipiv}\[i - 1\] \leq \(i\) indicates that, at the \(i\)th step, a row interchange was not required.

7: \textbf{fail} – NagError * \hspace{1cm} \textit{Input/Output}

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

6 \quad \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\textit{value}\rangle had an illegal value.

\textbf{NE_INT}

On entry, \textbf{m} = \langle\textit{value}\rangle.

Constraint: \textbf{m} \geq 0.

On entry, \textbf{n} = \langle\textit{value}\rangle.

Constraint: \textbf{n} \geq 0.

On entry, \textbf{pda} = \langle\textit{value}\rangle.

Constraint: \textbf{pda} > 0.

\textbf{NE_INT_2}

On entry, \textbf{pda} = \langle\textit{value}\rangle and \textbf{m} = \langle\textit{value}\rangle.

Constraint: \textbf{pda} \geq \max(1, \textbf{m}).

On entry, \textbf{pda} = \langle\textit{value}\rangle and \textbf{n} = \langle\textit{value}\rangle.

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

\textbf{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 factor $U$ is exactly singular, and division by zero will occur if it is used to solve a system of equations.

### 7 Accuracy

The computed factors $L$ and $U$ are the exact factors of a perturbed matrix $A + E$, where

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

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

### 8 Parallelism and Performance

`nag_dgetrf (f07adc)` is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_dgetrf (f07adc)` 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 $\frac{2}{3}n^3$ if $m = n$ (the usual case), $\frac{1}{3}n^2(3m - n)$ if $m > n$ and $\frac{2}{3}m^2(3n - m)$ if $m < n$.

A call to this function with $m = n$ may be followed by calls to the functions:

- `nag_dgetrs (f07aec)` to solve $AX = B$ or $A^TX = B$;
- `nag_dgecon (f07agc)` to estimate the condition number of $A$;
- `nag_dgetri (f07ajc)` to compute the inverse of $A$.

The complex analogue of this function is `nag_zgetrf (f07arc)`.

### 10 Example

This example computes the $LU$ factorization of the matrix $A$, where

$$ A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix}. $$
10.1 Program Text

/* nag_dgetrf (f07adc) 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, m, n, pda;
    Integer exit_status = 0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    double *a = 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]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I-1)*pda +J-1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    printf("nag_dgetrf (f07adc) 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"]", &m, &n);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n"]", &m, &n);
    #endif
    ipiv_len = MIN(m, n);
    #ifdef NAG_COLUMN_MAJOR
    pda = m;
    #else
    pda = n;
    #endif
    /* Allocate memory */
    if (!(a = NAG_ALLOC(m * n, double)) ||
        !(ipiv = NAG_ALLOC(ipiv_len, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    #include <nagf07.h>
    ...
/* Read A from data file */
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= n; ++j)
#ifdef _WIN32
        scanf_s("%lf", &A(i, j));
#else
        scanf("%lf", &A(i, j));
#endif
#endif
/* Factorize A */
/* nag_dgetrf (f07adc).
 * LU factorization of real m by n matrix */
# ifndef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
# endif
# ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
/* Print details of factorization */
/* nag_gen_real_mat_print (x04cac).
 * Print real general matrix (easy-to-use) */
fflush(stdout);
# ifndef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
/* Print pivot indices */
printf("%6"NAG_IFMT"%s", ipiv[i-1], i%7 == 0?"\n":" ");
printf("\n");
END:
NAG_FREE(a);
NAG_FREE(ipiv);
return exit_status;

10.2 Program Data

nag_dgetrf (f07adc) Example Program Data

4 4 :Values of M and N
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
# 10.3 Program Results

nag_dgetrf (f07adc) Example Program Results

Details of factorization

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.2500</td>
<td>-2.9500</td>
<td>-0.9500</td>
<td>-3.8000</td>
</tr>
<tr>
<td>2</td>
<td>0.3429</td>
<td>3.8914</td>
<td>2.3757</td>
<td>0.4129</td>
</tr>
<tr>
<td>3</td>
<td>0.3010</td>
<td>-0.4631</td>
<td>-1.5139</td>
<td>0.2948</td>
</tr>
<tr>
<td>4</td>
<td>-0.2114</td>
<td>-0.3299</td>
<td>0.0047</td>
<td>0.1314</td>
</tr>
</tbody>
</table>

ipiv

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
</table>