NAG C Library Function Document

nag_dgetrf (f07adc)

1 Purpose

nag_dgetrf (f07adc) computes the \( LU \) factorization of a real \( m \) by \( n \) matrix.

2 Specification

```c
void nag_dgetrf (Nag_OrderType order, Integer m, Integer n, double a[],
                Integer pda, Integer ipiv[], NagError *fail)
```

3 Description

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.

4 References


5 Parameters

1: `order` – Nag_OrderType

   - Input

   On entry: the `order` parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

   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 \( \max(1,pda \times n) \) when `order = Nag_ColMajor` and at least \( \max(1,pda \times m) \) when `order = Nag_RowMajor`.

   If `order = Nag_ColMajor`, the \((i,j)\)th element of the matrix \( A \) is stored in \( a[(j-1) \times pda + i - 1] \) and if `order = Nag_RowMajor`, the \((i,j)\)th element of the matrix \( A \) is stored in \( a[(i-1) \times pda + j - 1] \).

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

   On exit: \( a \) is overwritten by the factors \( L \) and \( U \); the unit diagonal elements of \( L \) are not stored.
5: \textbf{pda} – Integer \hfill \textit{Input}

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

Constraints:
    
    if \textbf{order} = \text{Nag\_ColMajor}, \textbf{pda} \geq \max(1, m);
    if \textbf{order} = \text{Nag\_RowMajor}, \textbf{pda} \geq \max(1, n).

6: \textbf{ipiv}[\textit{dim}] – Integer \hfill \textit{Output}

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

On exit: the pivot indices. Row \textit{i} of the matrix \textbf{A} was interchanged with row \textbf{ipiv}[	extit{i} – 1] for 
\textit{i} = 1, 2, \ldots, \min(m, n).

7: \textbf{fail} – \textbf{NagError} * \hfill \textit{Output}

The NAG error parameter (see the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

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

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

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

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

\textbf{NE\_SINGULAR}

\(u(\langle\textit{value}\rangle, \langle\textit{value}\rangle)\) is exactly zero. The factorization has been completed but the factor \textit{U} is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to invert \textit{A}.

\textbf{NE\_ALLOC\_FAIL}

Memory allocation failed.

\textbf{NE\_BAD\_PARAM}

On entry, parameter \langle\textit{value}\rangle had an illegal value.

\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 consult NAG for assistance.
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 Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^3$ if $m = n$ (the usual case), $\frac{4}{3}n^2(3n - m)$ if $m > n$ and $\frac{1}{3}m^2(3n - m)$ 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)`.

9 Example

To compute the $LU$ factorization of the matrix $A$, where

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

9.1 Program Text

```c
/* nag_dgetrf (f07adc) Example Program. */
* Copyright 2001 Numerical Algorithms Group.
* /
#include <stdio.h>
#include <nag.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;

    INIT_FAIL(fail);
    #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);
```
Vprintf("f07adc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[\n"] ;

Vscanf("%ld%ld%*[\n"] , &m, &n);
ipiv_len = MIN(m,n);
#endif NAG_COLUMN_MAJOR
pda = m;
#else
pda = n;
#endif

/* Allocate memory */
if ( !(a = NAG_ALLOC(m * n, double)) ||
     !(ipiv = NAG_ALLOC(ipiv_len, Integer)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &A(i,j));
}
Vscanf("%*[\n"] ;

/* Factorize A */
f07adc(order, m, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07adc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print details of factorization */
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n, a, m,
        "Details of factorization", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print pivot indices */
Vprintf("\nIPIV\n");
for (i = 1; i <= MIN(m,n); ++i)
    Vprintf("%6ld%s", ipiv[i-1], i%7==0 ?"\n":" ");
Vprintf("\n");

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

9.2 Program Data

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
### 9.3 Program Results

**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**

2 2 3 4