NAG C Library Function Document

nag_zgetrf (f07arc)

1 Purpose
nag_zgetrf (f07arc) computes the $LU$ factorization of a complex $m$ by $n$ matrix.

2 Specification

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

3 Description
nag_zgetrf (f07arc) forms the $LU$ factorization of a complex $m$ 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
   
   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
   
   On entry: $m$, the number of rows of the matrix $A$.

   Constraint: $m \geq 0$.

3:  
   n – Integer
   
   On entry: $n$, the number of columns of the matrix $A$.

   Constraint: $n \geq 0$.

4:  
   a[dim] – Complex
   
   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: \hspace{1em} pda – Integer

\textit{Input}

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

\textit{Constraints}:

\begin{align*}
\text{if } \texttt{order} = \text{Nag\_ColMajor}, & \quad \texttt{pda} \geq \max(1, m); \\
\text{if } \texttt{order} = \text{Nag\_RowMajor}, & \quad \texttt{pda} \geq \max(1, n).
\end{align*}

6: \hspace{1em} \texttt{ipiv}[\text{dim}] – Integer

\textit{Output}

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

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

7: \hspace{1em} fail – NagError *

\textit{Output}

The NAG error parameter (see the Essential Introduction).

6 \hspace{1em} \textbf{Error Indicators and Warnings}

\textbf{NE_INT}

On entry, \(m = (\text{value})\).

Constraint: \(m \geq 0\).

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

Constraint: \(n \geq 0\).

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

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

\textbf{NE_INT_2}

On entry, \(\texttt{pda} = (\text{value}), m = (\text{value})\).

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

On entry, \(\texttt{pda} = (\text{value}), n = (\text{value})\).

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

\textbf{NE_SINGULAR}

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

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

On entry, parameter \((\text{value})\) 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 real floating-point operations is approximately $\frac{8}{3}n^3$ if $m = n$ (the usual case), $\frac{4}{3}n^2(3n - m)$ if $m > n$ and $\frac{4}{3}m^2(3n - m)$ if $m < n$.

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

- nag_zgetrs (f07asc) to solve $AX = B$, $A^TX = B$ or $AHX = B$;
- nag_zgecon (f07auc) to estimate the condition number of $A$;
- nag_zgetri (f07awc) to compute the inverse of $A$.

The real analogue of this function is nag_dgetrf (f07adc).

9 Example

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

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix}.$$ 

9.1 Program Text

/* nag_zgetrf (f07arc) Example Program. * *
 * Copyright 2001 Numerical Algorithms Group. *
 * Mark 7, 2001. */

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

int main(void)
{
    /* Scalars */
    Integer i, j, m, n, pda, ipiv_len;
    Integer exit_status=0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Complex *a=0;
    Integer *ipiv=0;

#define NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
#else
#define A(I,J) a[(I-1)*pda + J - 1]
#endif
    order = Nag_ColMajor;
    INIT_FAIL(fail);
Vprintf("f07arc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[\n ]");
Vscanf("%ld%ld%*[\n ]", &m, &n);
#endif
pda = m;
#else
pda = n;
#endif
ipiv_len = MIN(m,n);
/* Allocate memory */
if ( !(a = NAG_ALLOC(m * n, Complex)) ||
!(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 , %lf )", &A(i,j).re, &A(i,j).im);
}
Vscanf("%*[\n ]");
/* Factorize A */
f07arc(order, m, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07arc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Print details of factorization */
x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n, a, pda,
       Nag_BracketForm, "%7.4f", "Details of factorization",
       Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80, 0, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04dbc.\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("%12ld%s", ipiv[i - 1], i%4 == 0 ?"\n": "");
Vprintf("\n");
END:
if (a) NAG_FREE(a);
if (ipiv) NAG_FREE(ipiv);
return exit_status;
}

9.2 Program Data

<table>
<thead>
<tr>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

:Values of M and N
(-1.34, 2.55) (0.28, 3.17) (-6.39, -2.20) (0.72, -0.92)
(-0.17, -1.41) (3.31, -0.15) (-0.15, 1.34) (1.29, 1.38)
(-3.29, -2.39) (-1.91, 4.42) (-0.14, -1.35) (1.72, 1.35)
(2.41, 0.39) (-0.56, 1.47) (-0.83, -0.69) (-1.96, 0.67)

:End of matrix A
### 9.3 Program Results

f07arc 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>(-3.2900, -2.3900)</td>
<td>(-1.9100, 4.4200)</td>
<td>(-0.1400, -1.3500)</td>
<td>(1.7200, 1.3500)</td>
</tr>
<tr>
<td>2</td>
<td>(0.2376, 0.2560)</td>
<td>(4.8952, -0.7114)</td>
<td>(-0.4623, 1.6966)</td>
<td>(1.2269, 0.6190)</td>
</tr>
<tr>
<td>3</td>
<td>(-0.1020, -0.7010)</td>
<td>(-0.6691, 0.3689)</td>
<td>(-5.1414, -1.1300)</td>
<td>(0.9983, 0.3850)</td>
</tr>
<tr>
<td>4</td>
<td>(-0.5359, 0.2707)</td>
<td>(-0.2040, 0.8601)</td>
<td>(0.0082, 0.1211)</td>
<td>(0.1482, -0.1252)</td>
</tr>
</tbody>
</table>

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

IPIV:

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