NAG Library Function Document

nag_zge_load (f16thc)

1 Purpose

nag_zge_load (f16thc) initializes a complex general matrix.

2 Specification

```c
#include <nag.h>
#include <nagf16.h>
void nag_zge_load (Nag_OrderType order, Integer m, Integer n, Complex alpha,
                  Complex diag, Complex a[], Integer pda, NagError *fail)
```

3 Description

nag_zge_load (f16thc) forms the complex $m$ by $n$ general matrix $A$ given by

$$a_{ij} = \begin{cases} 
  d & \text{if } i = j \\
  \alpha & \text{if } i \neq j 
\end{cases}$$

4 References


5 Arguments

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

   \textit{On entry}: the \textit{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 \textit{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}: \textit{order} = Nag_RowMajor or Nag_ColMajor.

2: \textbf{m} – Integer \hspace{1cm} \textit{Input}

   \textit{On entry}: $m$, the number of rows of the matrix $A$.

   \textit{Constraint}: $m \geq 0$.

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

   \textit{On entry}: $n$, the number of columns of the matrix $A$.

   \textit{Constraint}: $n \geq 0$.

4: \textbf{alpha} – Complex \hspace{1cm} \textit{Input}

   \textit{On entry}: the value, $\alpha$, to be assigned to the off-diagonal elements of $A$.

5: \textbf{diag} – Complex \hspace{1cm} \textit{Input}

   \textit{On entry}: the value, $d$, to be assigned to the diagonal elements of $A$. 
6:  \textit{a}[^{\text{dim}}] – Complex \hspace{1cm} \textit{Output}

\textbf{Note:} the dimension, \textit{dim}, of the array \textit{a} must be at least

\[
\max(1, \text{\textit{pda} } \times \text{\textit{n}}) \text{ when } \textit{order} = \text{Nag_ColMajor};
\max(1, \text{\textit{m} } \times \text{\textit{pda}}) \text{ when } \textit{order} = \text{Nag_RowMajor}.
\]

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

\textit{On exit:} the \textit{m} by \textit{n} general matrix \textit{A} with diagonal elements set to \textit{diag} and off-diagonal elements set to \textit{alpha}.

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

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

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

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

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

6 \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 value \rangle had an illegal value.

\textbf{NE_INT}

On entry, \textit{m} = \langle value \rangle.
Constraint: \textit{m} \geq 0.
On entry, \textit{n} = \langle value \rangle.
Constraint: \textit{n} \geq 0.

\textbf{NE_INT_2}

On entry, \textit{pda} = \langle value \rangle, \textit{m} = \langle value \rangle.
Constraint: \textit{pda} \geq \max(1, \textit{m}).

\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.

\textbf{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
The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001)).

8 Parallelism and Performance
Not applicable.

9 Further Comments
None.

10 Example
This example initializes a 4 by 3 complex matrix $A$, setting diagonal elements $9.0 + 0.0i$ and off-diagonal elements to $0.5 - 0.3i$.

10.1 Program Text
/* nag_zge_load (f16thc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 8, 2005. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf16.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Complex alpha, diag;
    Integer exit_status, m, n, pda;

    /* Arrays */
    Complex *a = 0;

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

    #ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
    #else
    order = Nag_RowMajor;
    #endif

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_zge_load (f16thc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef __WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif

    /* Read the problem dimensions */
ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%[\n] ", &m, &n);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%[\n] ", &m, &n);
#endif

/* Read scalar parameters */
ifdef _WIN32
    scanf_s(" ( %lf , %lf ) ( %lf , %lf )%[\n] ",
        &alpha.re, &alpha.im, &diag.re, &diag.im);
#else
    scanf(" ( %lf , %lf ) ( %lf , %lf )%[\n] ",
        &alpha.re, &alpha.im, &diag.re, &diag.im);
#endif
ifdef NAG_COLUMN_MAJOR
    pda = m;
#else
    pda = n;
#endif

if (m>0 && n>0)
{
    /* Allocate memory */
    if (!((a = NAG_ALLOC(m*n, Complex)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else
{
    printf("Invalid m or n\n");
    exit_status = 1;
    return exit_status;
}

/* nag_zge_load (f16thc).  
 * Initialize complex general matrix.  
 */
    nag_zge_load(order, m, n, alpha, diag, a, pda, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_zge_load.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Print generated matrix A */
/* nag_gen_complex_mat_print_comp (x04dbc).  
 * Print complex general matrix (comprehensive)  
 */
    fflush(stdout);
    nag_gen_complex_mat_print_comp(order, Nag_GeneralMatrix, 
        Nag_NonUnitDiag, m, n, a, pda, 
        Nag_BracketForm, "%5.2f", 
        "Generated Matrix A", Nag_IntegerLabels, 
        0, Nag_IntegerLabels, 0, 80, 0, 0, 
        &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_gen_complex_mat_print_comp (x04dbc).\n%s" "\n", fail.message);
        exit_status = 1;
        goto END;
    }
END:
    NAG_FREE(a);
    return exit_status;
}

10.2 Program Data

nag_zge_load (f16thc) Example Program Data
4 3 : m, n the dimensions of matrix A
( 0.5,-0.3) ( 9.0, 0.0) : alpha, diag

10.3 Program Results

nag_zge_load (f16thc) Example Program Results

Generated Matrix A

1     2     3
1 ( 9.00, 0.00) ( 0.50,-0.30) ( 0.50,-0.30)
2 ( 0.50,-0.30) ( 9.00, 0.00) ( 0.50,-0.30)
3 ( 0.50,-0.30) ( 0.50,-0.30) ( 9.00, 0.00)
4 ( 0.50,-0.30) ( 0.50,-0.30) ( 0.50,-0.30)