

# NAG Library Function Document

## nag\_zge\_load (f16thc)

### 1 Purpose

nag\_zge\_load (f16thc) initializes a complex general matrix.

### 2 Specification

```
#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

Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001) *Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard* University of Tennessee, Knoxville, Tennessee <http://www.netlib.org/blas/blast-forum/blas-report.pdf>

### 5 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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: **alpha** – Complex *Input*  
*On entry:* the value,  $\alpha$ , to be assigned to the off-diagonal elements of  $A$ .
- 5: **diag** – Complex *Input*  
*On entry:* the value,  $d$ , to be assigned to the diagonal elements of  $A$ .

6: **a**[*dim*] – Complex *Output*

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

$\max(1, \mathbf{pda} \times \mathbf{n})$  when **order** = Nag\_ColMajor;  
 $\max(1, \mathbf{m} \times \mathbf{pda})$  when **order** = Nag\_RowMajor.

If **order** = Nag\_ColMajor,  $A_{ij}$  is stored in **a**[(*j* – 1) × **pda** + *i* – 1].

If **order** = Nag\_RowMajor,  $A_{ij}$  is stored in **a**[(*i* – 1) × **pda** + *j* – 1].

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

7: **pda** – Integer *Input*

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

*Constraints:*

if **order** = Nag\_ColMajor, **pda** ≥ max(1, **m**);  
 if **order** = Nag\_RowMajor, **pda** ≥ max(1, **n**).

8: **fail** – NagError \* *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument *⟨value⟩* had an illegal value.

### NE\_INT

On entry, **m** = *⟨value⟩*.

Constraint: **m** ≥ 0.

On entry, **n** = *⟨value⟩*.

Constraint: **n** ≥ 0.

### NE\_INT\_2

On entry, **pda** = *⟨value⟩*, **m** = *⟨value⟩*.

Constraint: **pda** ≥ max(1, **m**).

On entry, **pda** = *⟨value⟩* and **n** = *⟨value⟩*.

Constraint: **pda** ≥ 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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.  
See Section 2.7.5 in How to Use the NAG Library and its Documentation 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**

nag\_zge\_load (f16thc) is not threaded in any implementation.

**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.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
fflush(stdout);
nag_gen_complx_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_complx_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)

```

---