

## NAG Library Function Document

### nag\_dge\_load (f16qhc)

#### 1 Purpose

nag\_dge\_load (f16qhc) initializes a real general matrix.

#### 2 Specification

```
#include <nag.h>
#include <nagf16.h>

void nag_dge_load (Nag_OrderType order, Integer m, Integer n, double alpha,
                  double diag, double a[], Integer pda, NagError *fail)
```

#### 3 Description

nag\_dge\_load (f16qhc) forms the real  $m$  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 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: **alpha** – double *Input*  
*On entry:* the value,  $\alpha$ , to be assigned to the off-diagonal elements of  $A$ .
- 5: **diag** – double *Input*  
*On entry:* the value,  $d$ , to be assigned to the diagonal elements of  $A$ .

- 6: **a**[*dim*] – double *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*.
- 7: **pda** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) of the matrix *A* in the array **a**.  
*Constraint:* **pda** ≥ max(1, **m**).
- 8: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.  
See Section 3.2.1.2 in the Essential Introduction 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.

On entry, **pda** = *<value>*.  
Constraint: **pda** ≥ max(1, **m**).

On entry, **pda** = *<value>*.  
Constraint: **pda** ≥ max(1, **n**).

### NE\_INT\_2

On entry, **pda** = *<value>*, **m** = *<value>*.  
Constraint: **pda** ≥ max(1, **m**).

### 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.

## 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 real general matrix,  $A$ , with diagonal off-diagonal value,  $\alpha = 1.23$  and diagonal value,  $d = 3.45$ .

### 10.1 Program Text

```

/* nag_dge_load (f16qhc) 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 */
    double      alpha, diag;
    Integer     exit_status, m, n, pda;
    /* Arrays */
    double      *a = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError    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

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_dge_load (f16qhc) 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%*[\n] ", &alpha, &diag);
#else
    scanf("%lf%lf%*[\n] ", &alpha, &diag);
#endif

    if (order == Nag_ColMajor)
        pda = m;
    else
        pda = n;

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

    /* nag_dge_load (f16qhc).
     * General matrix initialisation.
     */
    nag_dge_load(order, m, n, alpha, diag, a, pda, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dge_load (f16qhc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print output */
    /* nag_gen_real_mat_print (x04cac).
     * Print real general matrix (easy-to-use)
     */
    fflush(stdout);
    nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                           m, n, a, pda, "Matrix A", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
              fail.message);
        exit_status = 1;
        goto END;
    }

END:
    NAG_FREE(a);

    return exit_status;
}

```

## 10.2 Program Data

```
nag_dge_load (f16qhc) Example Program Data
  4 3                               :Values of m, n
  1.23 3.45                          :Values of alpha, diag
```

## 10.3 Program Results

nag\_dge\_load (f16qhc) Example Program Results

```
Matrix A
      1      2      3
1      3.4500      1.2300      1.2300
2      1.2300      3.4500      1.2300
3      1.2300      1.2300      3.4500
4      1.2300      1.2300      1.2300
```

---