

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 | <i>Input</i> |
| | <i>On entry:</i> 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. | |
| | <i>Constraint:</i> order = Nag_RowMajor or Nag_ColMajor. | |
| 2: | m – Integer | <i>Input</i> |
| | <i>On entry:</i> m , the number of rows of the matrix A . | |
| | <i>Constraint:</i> $m \geq 0$. | |
| 3: | n – Integer | <i>Input</i> |
| | <i>On entry:</i> n , the number of columns of the matrix A . | |
| | <i>Constraint:</i> $n \geq 0$. | |
| 4: | alpha – double | <i>Input</i> |
| | <i>On entry:</i> the value, α , to be assigned to the off-diagonal elements of A . | |
| 5: | diag – double | <i>Input</i> |
| | <i>On entry:</i> 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**) 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_dge_load (f16qhc) is not threaded in any implementation.

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.
 *
 * 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 */
    double alpha, diag;
    Integer exit_status, m, n, pda;
    /* Arrays */
    double *a = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;

#ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
#else
    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 initialization.
     */
    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
```
