

NAG C Library Function Document

nag_dtr_load (f16qgc)

1 Purpose

nag_dtr_load (f16qgc) initializes a real triangular matrix.

2 Specification

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

```
void nag_dtr_load (Nag_OrderType order, Nag_UploType uplo, Integer n, double alpha,
                  double diag, double a[], Integer pda, NagError *fail)
```

3 Description

nag_dtr_load (f16qgc) forms the real n by n triangular matrix A given by

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

4 References

The BLAS Technical Forum Standard (2001) www.netlib.org/blas/blast-forum

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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this argument.
Constraint: **order = Nag_RowMajor** or **Nag_ColMajor**.
- 2: **uplo** – Nag_UploType *Input*
On entry: specifies whether the upper or lower triangular part of A is stored.
uplo = Nag_Upper
 The upper triangular part of A is stored.
uplo = Nag_Lower
 The lower triangular part of A is stored.
Constraint: **uplo = Nag_Upper** or **Nag_Lower**.
- 3: **n** – Integer *Input*
On entry: n , the order of the matrix A .
Constraint: $n \geq 0$.
- 4: **alpha** – double *Input*
On entry: the value, α , 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 $\max(1, pda \times n)$.
 If **order** = **Nag_ColMajor**, the (i,j) th element of the matrix A is stored in **a**[($j - 1$) \times **pda** + $i - 1$].
 If **order** = **Nag_RowMajor**, the (i,j) th element of the matrix A is stored in **a**[($i - 1$) \times **pda** + $j - 1$].
On exit: the n by n triangular matrix A .
 If **uplo** = **Nag_Upper**, A is upper triangular and the elements of the array corresponding to the lower triangular part of A are not referenced.
 If **uplo** = **Nag_Lower**, A is lower triangular and the elements of the array corresponding to the upper triangular part of A are not referenced.
- 7: **pda** – Integer *Input*
On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **a**.
Constraint: **pda** \geq $\max(1, n)$.
- 8: **fail** – NagError * *Input/Output*
 The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **n** = $\langle value \rangle$.
 Constraint: **n** \geq 0.

NE_INT_2

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.
 Constraint: **pda** \geq $\max(1, n)$.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

To initialise the lower triangular matrix A with diagonal elements given by $d = 3.45$ and off-diagonal elements given by $\alpha = 1.23$.

9.1 Program Text

```

/* nag_dtr_load (f16qgc) Example Program.
 *
 * Copyright 2005 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, n, pda;

    /* Arrays */
    double *a=0;
    char nag_enum_arg[40];

    /* Nag Types */
    NagError fail;
    Nag_MatrixType matrix;
    Nag_OrderType order;
    Nag_UploType uplo;

#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);
    Vprintf( "nag_dtr_load (f16qgc) Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n] ");
    /* Read the problem dimension */
    Vscanf("%ld%*[\n] ", &n);
    /* Read uplo */
    Vscanf("%s%*[\n] ", nag_enum_arg);
    /* nag_enum_name_to_value(x04nac).
     * Converts NAG enum member name to value
     */
    uplo = nag_enum_name_to_value(nag_enum_arg);
    /* Read scalar parameters */
    Vscanf("%lf%lf%*[\n] ", &alpha, &diag);

    pda = n;

    if (n > 0)
    {
        /* Allocate memory */
        if ( !(a = NAG_ALLOC(n*pda, double)) )
        {
            Vprintf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    else

```

```

    {
        Vprintf("Invalid n\n");
        exit_status = 1;
        return exit_status;
    }

/* nag_dtr_load(f16qgc).
 * Triangular matrix initialise.
 *
 */
nag_dtr_load(order, uplo, n, alpha, diag, a, pda, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from dtr_load.\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)
 */
if (uplo == Nag_Upper)
    matrix = Nag_UpperMatrix;
else
    matrix = Nag_LowerMatrix;

nag_gen_real_mat_print(order, matrix, Nag_NonUnitDiag,
                        n, n, a, pda, "Matrix A",
                        0, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
    }

END:
if (a) NAG_FREE(a);

return exit_status;
}

```

9.2 Program Data

```

nag_dtr_load (f16qgc) Example Program Data
4                               :Value of n
Nag_Lower                       :Value of uplo
1.23 3.45                       :Values of alpha, diag

```

9.3 Program Results

```

nag_dtr_load (f16qgc) Example Program Results

```

Matrix A

	1	2	3	4
1	3.4500			
2	1.2300	3.4500		
3	1.2300	1.2300	3.4500	
4	1.2300	1.2300	1.2300	3.4500
