

# NAG C Library Function Document

## nag\_ztr\_load (f16tgc)

### 1 Purpose

nag\_ztr\_load (f16tgc) initializes a complex triangular matrix.

### 2 Specification

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

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

### 3 Description

nag\_ztr\_load (f16tgc) forms the complex  $n$  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](http://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** – 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  $\max(1, \mathbf{pda} \times \mathbf{n})$ .  
 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  $A$  with diagonal elements set to **diag** and strictly upper or lower elements set to **alpha**.  
 If **uplo** = **Nag\_Upper**,  $A$  is upper triangular and the elements of the array below the diagonal are not referenced.  
 If **uplo** = **Nag\_Lower**,  $A$  is lower triangular and the elements of the array above the diagonal 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**  $\geq$   $\max(1, \mathbf{n})$ .
- 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**  $\geq$   $\max(1, \mathbf{n})$ .

## 7 Accuracy

Not applicable.

## 8 Further Comments

None.

## 9 Example

This example initializes a 4 by 4 lower triangular matrix  $A$ , setting diagonal elements to  $9.0 + 0.0i$  and strictly lower elements to  $0.5 - 0.3i$ .

## 9.1 Program Text

```

/* nag_ztr_load (f16tgc) 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 */
    Complex alpha, diag;
    Integer exit_status, n, pda;

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

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

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

    exit_status = 0;
    INIT_FAIL(fail);

    Vprintf( "nag_ztr_load (f16tgc) Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n] ");

    /* Read the problem dimension */
    Vscanf("%ld%*[\n] ", &n);

    /* Read the uplo parameter */
    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 ) ( %lf , %lf )%*[\n] ",
          &alpha.re, &alpha.im, &diag.re, &diag.im);

    pda = n;

    if (n > 0)
    {

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

```

```

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

/* nag_ztr_load(f16tgc).
 * Initialize complex triangular matrix.
 */
nag_ztr_load(order, uplo, n, alpha, diag, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_ztr_load.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

if (uplo == Nag_Upper)
{
    matrix = Nag_UpperMatrix;
}
else
{
    matrix = Nag_LowerMatrix;
}
/* Print generated matrix A */
/* nag_gen_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
nag_gen_complx_mat_print_comp(order, matrix, Nag_NonUnitDiag, n, 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)
{
    Vprintf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s"
           "\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (a) NAG_FREE(a);

return exit_status;
}

```

## 9.2 Program Data

```

nag_ztr_load (f16tgc) Example Program Data
4             : n the dimension of matrix A
Nag_Lower    : uplo
( 0.5,-0.3) ( 9.0, 0.0) : alpha, diag

```

## 9.3 Program Results

```

nag_ztr_load (f16tgc) Example Program Results

```

Generated Matrix A

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