

NAG C Library Function Document

nag_zher2 (f16src)

1 Purpose

nag_zher2 (f16src) performs a Hermitian rank-2 update on a complex Hermitian matrix.

2 Specification

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

```
void nag_zher2 (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex alpha,
               const Complex x[], Integer incx, const Complex y[], Integer incy, double beta,
               Complex a[], Integer pda, NagError *fail)
```

3 Description

nag_zher2 (f16src) performs the Hermitian rank-2 update operation

$$A \leftarrow \alpha xy^H + \bar{\alpha}yx^H + \beta A$$

where A is an n by n complex Hermitian matrix, x and y are n element complex vectors, α is a complex scalar and β is a real scalar.

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** – Complex *Input*
On entry: the scalar α .
- 5: **x**[*dim*] – const Complex *Input*
Note: the dimension, *dim*, of the array **x** must be at least $\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incx}|)$.
On entry: the vector x .
- 6: **incx** – Integer *Input*
On entry: the increment in the subscripts of **x** between successive elements of x .
Constraint: **incx** $\neq 0$.
- 7: **y**[*dim*] – const Complex *Input*
Note: the dimension, *dim*, of the array **y** must be at least $\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incy}|)$.
On entry: the vector y .
- 8: **incy** – Integer *Input*
On entry: the increment in the subscripts of **y** between successive elements of y .
Constraint: **incy** $\neq 0$.
- 9: **beta** – double *Input*
On entry: the scalar β .
- 10: **a**[*dim*] – Complex *Input/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 entry: the n by n Hermitian matrix A .
If **uplo** = **Nag_Upper**, the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced.
If **uplo** = **Nag_Lower**, the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.
On exit: the updated matrix A . The imaginary parts of the diagonal elements are set to zero.
- 11: **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** $\geq \max(1, \mathbf{n})$.
- 12: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_INT

On entry, **incx** = $\langle value \rangle$.

Constraint: **incx** $\neq 0$.

On entry, **incy** = $\langle value \rangle$.

Constraint: **incy** $\neq 0$.

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 0 .

NE_INT_2

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.

Constraint: **pda** $\geq \max(1, \mathbf{n})$.

7 Accuracy

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of The BLAS Technical Forum Standard (2001)).

8 Further Comments

None.

9 Example

Perform rank-2 update of complex Hermitian matrix A using vectors x and y :

$$A \leftarrow A - xy^H - yx^H,$$

where A is the 4 by 4 matrix given by

$$A = \begin{pmatrix} 23.0 + 0.0i & 10.0 - 17.0i & 13.0 + 14.2i & -19.0 + 8.0i \\ 10.0 + 17.0i & 1.0 + 0.0i & 0.3 + 1.2i & -4.7 + 2.1i \\ 13.0 - 14.2i & 0.3 - 1.2i & 1.0 + 0.0i & -5.9 + 0.1i \\ -19.0 - 8.0i & -4.7 + 2.1i & -5.9 + 0.1i & 1.0 + 0.0i \end{pmatrix},$$

and where

$$x = \begin{pmatrix} 2.0 + 1.0i \\ 2.0 + 3.0i \\ 0.2 - 1.0i \\ -1.0 - 2.0i \end{pmatrix}$$

and

$$y = \begin{pmatrix} 5.0 + 1.0i \\ -2.0 + 1.0i \\ 7.0 - 1.0i \\ -5.0 - 2.0i \end{pmatrix}.$$

The vector y is stored in every second element of array **y** (**incy** = 2).

9.1 Program Text

```

/* nag_zher2 (fl6src) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2005.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagfl6.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Complex alpha;
    double beta;
    Integer exit_status, i, incx, incy, j, n, pda, xlen, ylen;

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

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

#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_zher2 (fl6src) Example Program Results\n\n");

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

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

    /* Read the uplo storage 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 )%*[\n] ", &alpha.re, &alpha.im);
    Vscanf("%lf%*[\n] ", &beta);
    /* Read increment parameters */
    Vscanf("%ld%ld%*[\n] ", &incx, &incy);

    pda = n;

    xlen = MAX(1, 1 + (n - 1)*ABS(incx));
    ylen = MAX(1, 1 + (n - 1)*ABS(incy));

    if (n > 0)

```

```

{
    /* Allocate memory */
    if ( !(a = NAG_ALLOC(pda*n, Complex)) ||
         !(x = NAG_ALLOC(xlen, Complex)) ||
         !(y = NAG_ALLOC(ylen, Complex)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else
{
    Vprintf("Invalid n\n");
    exit_status = 1;
    return exit_status;
}

/* Input matrix A and vector x */

if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
        Vscanf("%*[\n] ");
    }
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
        Vscanf("%*[\n] ");
    }
}
for (i = 0; i < xlen; ++i)
    Vscanf(" ( %lf , %lf )%*[\n] ", &x[i].re, &x[i].im);
for (i = 0; i < ylen; ++i)
    Vscanf(" ( %lf , %lf )%*[\n] ", &y[i].re, &y[i].im);

/* nag_zher2(f16src).
 * Rank two update of complex Hermitian matrix.
 */
nag_zher2(order, uplo, n, alpha, x, incx, y, incy, beta, a, pda,
          &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_zher2.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

if (uplo == Nag_Upper)
{
    matrix = Nag_UpperMatrix;
}
else
{
    matrix = Nag_LowerMatrix;
}
/* Print updated 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, "%5.1f",

```

```

                                "Updated Matrix A", Nag_IntegerLabels,
                                0, Nag_IntegerLabels, 0, 80, 0, 0,
                                &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_gen_complex_mat_print_comp (x04dbc).\n%s"
           "\n",fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
if (x) NAG_FREE(x);
if (y) NAG_FREE(y);

return exit_status;
}

```

9.2 Program Data

```

nag_zher2 (f16src) Example Program Data
  4                               :Value of n
  Nag_Lower                       :Storage of A
(-1.0, 0.0)                       :Value of alpha
  1.0                             :Value of beta
  1 2                             :Values of incx and incy
( 23.0,  0.0)
( 10.0, 17.0) ( 1.0,  0.0)
( 13.0,-14.2) ( 0.3,-1.2) ( 1.0,  0.0)
(-19.0, -8.0) (-4.7, 2.1) (-5.9, 0.1) ( 1.0,  0.0) :End of matrix A
( 2.0,  1.0)
( 2.0,  3.0)
( 0.2,-1.0)
(-1.0,-2.0)                               :End of vector x
( 5.0,  1.0)
( 0.0,  0.0)
(-2.0,  1.0)
( 0.0,  0.0)
( 7.0,-1.0)
( 0.0,  0.0)
(-5.0,-2.0)                               :End of vector y

```

9.3 Program Results

nag_zher2 (f16src) Example Program Results

Updated Matrix A

	1	2	3	4
1	(1.0, 0.0)			
2	(0.0, 0.0)	(3.0, 0.0)		
3	(0.0, 0.0)	(-9.3, 20.0)	(-3.8, 0.0)	
4	(0.0, 0.0)	(11.3,-13.9)	(-1.9, 20.5)	(-17.0, 0.0)
