

# NAG Library Function Document

## nag\_zhpr2 (f16ssc)

### 1 Purpose

nag\_zhpr2 (f16ssc) performs a Hermitian rank-2 update on a complex Hermitian matrix stored in packed form.

### 2 Specification

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

void nag_zhpr2 (Nag_OrderType order, Nag_UploType uplo, Integer n,
               Complex alpha, const Complex x[], Integer incx, const Complex y[],
               Integer incy, double beta, Complex ap[], NagError *fail)
```

### 3 Description

nag\_zhpr2 (f16ssc) 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, stored in packed form,  $x$  and  $y$  are  $n$ -element complex vectors, while  $\alpha$  is a complex scalar and  $\beta$  is a real scalar.

### 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 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.  
*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  $\alpha$ .
- 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 *n*-element vector *x*.  
 If  $\mathbf{incx} > 0$ ,  $x_i$  must be stored in  $\mathbf{x}[(i - 1) \times \mathbf{incx}]$ , for  $i = 1, 2, \dots, \mathbf{n}$ .  
 If  $\mathbf{incx} < 0$ ,  $x_i$  must be stored in  $\mathbf{x}[(\mathbf{n} - i) \times |\mathbf{incx}|]$ , for  $i = 1, 2, \dots, \mathbf{n}$ .  
 Intermediate elements of **x** are not referenced. If  $\mathbf{n} = 0$ , **x** is not referenced and may be **NULL**.
- 6: **incx** – Integer *Input*  
*On entry:* the increment in the subscripts of **x** between successive elements of *x*.  
*Constraint:*  $\mathbf{incx} \neq 0$ .
- 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 *n*-element vector *y*.  
 If  $\mathbf{incy} > 0$ ,  $y_i$  must be stored in  $\mathbf{y}[(i - 1) \times \mathbf{incy}]$ , for  $i = 1, 2, \dots, \mathbf{n}$ .  
 If  $\mathbf{incy} < 0$ ,  $y_i$  must be stored in  $\mathbf{y}[(\mathbf{n} - i) \times |\mathbf{incy}|]$ , for  $i = 1, 2, \dots, \mathbf{n}$ .  
 Intermediate elements of **y** are not referenced. If  $\alpha = 0.0$  or  $\mathbf{n} = 0$ , **y** is not referenced and may be **NULL**.
- 8: **incy** – Integer *Input*  
*On entry:* the increment in the subscripts of **y** between successive elements of *y*.  
*Constraint:*  $\mathbf{incy} \neq 0$ .
- 9: **beta** – double *Input*  
*On entry:* the scalar  $\beta$ .
- 10: **ap**[*dim*] – Complex *Input/Output*  
**Note:** the dimension, *dim*, of the array **ap** must be at least  $\max(1, \mathbf{n} \times (\mathbf{n} + 1)/2)$ .  
*On entry:* the *n* by *n* Hermitian matrix *A*, packed by rows or columns.  
 The storage of elements  $A_{ij}$  depends on the **order** and **uplo** arguments as follows:  
   if **order** = Nag\_ColMajor and **uplo** = Nag\_Upper,  
      $A_{ij}$  is stored in  $\mathbf{ap}[(j - 1) \times j/2 + i - 1]$ , for  $i \leq j$ ;  
   if **order** = Nag\_ColMajor and **uplo** = Nag\_Lower,  
      $A_{ij}$  is stored in  $\mathbf{ap}[(2\mathbf{n} - j) \times (j - 1)/2 + i - 1]$ , for  $i \geq j$ ;  
   if **order** = Nag\_RowMajor and **uplo** = Nag\_Upper,  
      $A_{ij}$  is stored in  $\mathbf{ap}[(2\mathbf{n} - i) \times (i - 1)/2 + j - 1]$ , for  $i \leq j$ ;  
   if **order** = Nag\_RowMajor and **uplo** = Nag\_Lower,  
      $A_{ij}$  is stored in  $\mathbf{ap}[(i - 1) \times i/2 + j - 1]$ , for  $i \geq j$ .  
*On exit:* the updated matrix *A*. The imaginary parts of the diagonal elements are set to zero.
- 11: **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 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

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

### NE\_INT

On entry,  $\mathbf{incx} = \langle value \rangle$ .

Constraint:  $\mathbf{incx} \neq 0$ .

On entry,  $\mathbf{incy} = \langle value \rangle$ .

Constraint:  $\mathbf{incy} \neq 0$ .

On entry,  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{n} \geq 0$ .

### NE\_INTERNAL\_ERROR

An unexpected error has been triggered by this function. Please contact NAG.

See Section 3.6.6 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 3.6.5 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\_zhpr2 (f16ssc) is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

Perform rank-2 update of complex Hermitian matrix  $A$ , stored using packed storage format, 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  $y$  ( $\text{incy} = 2$ ).

## 10.1 Program Text

```

/* nag_zhpr2 (f16ssc) 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 */
    Complex alpha;
    double beta;
    Integer exit_status, i, incx, incy, j, n, pda, xlen, ylen;

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

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

#ifdef NAG_COLUMN_MAJOR
#define A_UPPER(I, J) ap[J*(J-1)/2 + I - 1]
#define A_LOWER(I, J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_ColMajor;
#else
#define A_LOWER(I, J) ap[I*(I-1)/2 + J - 1]
#define A_UPPER(I, J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_RowMajor;
#endif

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_zhpr2 (f16ssc) Example Program Results\n\n");

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");

```

```

#endif

/* Read the problem dimension */
#ifdef _WIN32
scanf_s("%" NAG_IFMT "%*[\n] ", &n);
#else
scanf("%" NAG_IFMT "%*[\n] ", &n);
#endif

/* Read the uplo storage parameter */
#ifdef _WIN32
scanf_s("%39s%*[\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
scanf("%39s%*[\n] ", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

/* Read scalar parameters */
#ifdef _WIN32
scanf_s(" ( %lf , %lf )%*[\n] ", &alpha.re, &alpha.im);
#else
scanf(" ( %lf , %lf )%*[\n] ", &alpha.re, &alpha.im);
#endif
#ifdef _WIN32
scanf_s("%lf%*[\n] ", &beta);
#else
scanf("%lf%*[\n] ", &beta);
#endif
/* Read increment parameters */
#ifdef _WIN32
scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &incx, &incy);
#else
scanf("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &incx, &incy);
#endif

pda = n;

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

if (n > 0) {
/* Allocate memory */
if (!(ap = NAG_ALLOC(pda * n, Complex)) ||
!(x = NAG_ALLOC(xlen, Complex)) || !(y = NAG_ALLOC(ylen, Complex)))
{
printf("Allocation failure\n");
exit_status = -1;
goto END;
}
}
else {
printf("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)
#ifdef _WIN32
scanf_s(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(i, j).im);
#else
scanf(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(i, j).im);
#endif
}
#ifdef _WIN32
scanf_s("%*[\n] ");

```

```

#else
    scanf("%*[\n] ");
#endif
}
else {
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= i; ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &A_LOWER(i, j).re, &A_LOWER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &A_LOWER(i, j).re, &A_LOWER(i, j).im);
#endif
#ifdef _WIN32
            scanf_s("%*[\n] ");
#else
            scanf("%*[\n] ");
#endif
        }
        for (i = 0; i < xlen; ++i)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )%*[\n] ", &x[i].re, &x[i].im);
#else
            scanf(" ( %lf , %lf )%*[\n] ", &x[i].re, &x[i].im);
#endif
        for (i = 0; i < ylen; ++i)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )%*[\n] ", &y[i].re, &y[i].im);
#else
            scanf(" ( %lf , %lf )%*[\n] ", &y[i].re, &y[i].im);
#endif
    }

    /* nag_zhpr2 (f16ssc).
     * Rank two update of complex Hermitian matrix,
     * packed storage.
     */
    nag_zhpr2(order, uplo, n, alpha, x, incx, y, incy, beta, ap, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zhpr2 (f16ssc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print updated matrix A */
    /* nag_pack_complx_mat_print_comp (x04ddc).
     * Print complex packed triangular matrix (comprehensive)
     */
    fflush(stdout);
    nag_pack_complx_mat_print_comp(order, uplo, Nag_NonUnitDiag, n, ap,
                                   Nag_BracketForm, "%5.1f",
                                   "Updated Matrix A", Nag_IntegerLabels,
                                   0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_pack_complx_mat_print_comp (x04ddc).\n%s"
              "\n", fail.message);
        exit_status = 1;
        goto END;
    }
}
END:
NAG_FREE(ap);
NAG_FREE(x);
NAG_FREE(y);

return exit_status;
}

```

## 10.2 Program Data

```
nag_zhpr2 (f16ssc) 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
```

## 10.3 Program Results

```
nag_zhpr2 (f16ssc) 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)
```

---