# NAG Toolbox

# nag_matop_complex_addsub (f01cw)

## 1 Purpose

nag_matop_complex_addsub (f01cw) adds two complex matrices, each one optionally transposed and multiplied by a scalar.

## 2 Syntax

```
[c, ifail] = nag_matop_complex_addsub(transa, transb, m, n, alpha, a, beta, b)
[c, ifail] = f01cw(transa, transb, m, n, alpha, a, beta, b)
```

## 3 Description

nag_matop_complex_addsub (f01cw) performs one of the operations

$C := \alpha A + \beta B$,

$C := \alpha A^{\mathrm{T}} + \beta B$,

$C := \alpha A^{\mathrm{H}} + \beta B$,

$C := \alpha A + \beta B^{\mathrm{T}}$,

$C := \alpha A^{\mathrm{T}} + \beta B^{\mathrm{T}}$,

$C := \alpha A^{\mathrm{H}} + \beta B^{\mathrm{T}}$,

$C := \alpha A + \beta B^{\mathrm{H}}$,

$C := \alpha A^{\mathrm{T}} + \beta B^{\mathrm{H}}$ or

$C := \alpha A^{\mathrm{H}} + \beta B^{\mathrm{H}}$,

where $A$, $B$ and $C$ are matrices, $\alpha$ and $\beta$ are scalars, $T$ denotes transposition and $H$ denotes conjugate transposition. For efficiency, the function contains special code for the cases when one or both of $\alpha$, $\beta$ is equal to zero, unity or minus unity. The matrices, or their transposes, must be compatible for addition. $A$ and $B$ are either $m$ by $n$ or $n$ by $m$ matrices, depending on whether they are to be transposed before addition. $C$ is an $m$ by $n$ matrix.

## 4 References

None.

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **transa** – CHARACTER(1)

2: **transb** – CHARACTER(1)

**transa** and **transb** must specify whether or not the matrix $A$ and the matrix $B$, respectively, are to be transposed before addition.

**transa** or **transb** = 'N'
The matrix will not be transposed.

**transa** or **transb** = 'T'
The matrix will be transposed.

**transa** or **transb** = 'C'
The matrix will be transposed and conjugated.

*Constraint*: **transa** or **transb** = 'N', 'T' or 'C'.

3: **m** – INTEGER

$m$, the number of rows of the matrices $A$ and $B$ or their transposes. Also the number of rows of the matrix $C$.

*Constraint*: **m** $\geq 0$.

4: **n** – INTEGER

$n$, the number of columns of the matrices $A$ and $B$ or their transposes. Also the number of columns of the matrix $C$.

*Constraint*: **n** $\geq 0$.

5: **alpha** – COMPLEX (KIND=nag_wp)

The scalar $\alpha$, by which matrix $A$ is multiplied before addition.

6: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, $lda$, of the array **a** must satisfy

if **transa** = 'N', $lda \geq \max(1,\mathbf{m})$;
otherwise $lda \geq \max(1,\mathbf{n})$.

The second dimension of the array **a** must be at least $\max(1,\mathbf{n})$ if **alpha** $\neq 0$ and **transa** = 'N', $\max(1,\mathbf{m})$ if **alpha** $\neq 0$ and **transa** = 'T' or 'C' and at least 1 if **alpha** $= 0$.

The matrix $A$. If $\alpha = 0$, the array **a** is not referenced.

7: **beta** – COMPLEX (KIND=nag_wp)

The scalar $\beta$, by which matrix $B$ is multiplied before addition.

8: **b**$(ldb,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, $ldb$, of the array **b** must satisfy

if **transb** = 'N', $ldb \geq \max(1,\mathbf{m})$;
otherwise $ldb \geq \max(1,\mathbf{n})$.

The second dimension of the array **b** must be at least $\max(1,\mathbf{n})$ if **beta** $\neq 0$ and **transb** = 'N', $\max(1,\mathbf{m})$ if **beta** $\neq 0$ and **transb** = 'T' or 'C' and at least 1 if **beta** $= 0$.

The matrix $B$. If $\beta = 0$, the array **b** is not referenced.

## 5.2 Optional Input Parameters

None.

## 5.3 Output Parameters

1: **c**$(ldc,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **c** will be $\max(1,\mathbf{m})$.

The second dimension of the array **c** will be $\max(1,\mathbf{n})$.

The elements of the $m$ by $n$ matrix $C$.

2: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 1$

On entry, one or both of **transa** or **transb** is not equal to `N', `T' or `C'.

**ifail** $= 2$

On entry, one or both of **m** or **n** is less than 0.

**ifail** $= 3$

On entry, $lda < \max(1, P)$, where P $=$ **m** if **transa** $=$ 'N', and P $=$ **n** otherwise.

**ifail** $= 4$

On entry, $ldb < \max(1, P)$, where P $=$ **m** if **transb** $=$ 'N', and P $=$ **n** otherwise.

**ifail** $= 5$

On entry, $ldc < \max(1, \mathbf{m})$.

**ifail** $= -99$

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

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

The results returned by nag_matop_complex_addsub (f01cw) are accurate to ***machine precision***.

## 8 Further Comments

The time taken for a call of nag_matop_complex_addsub (f01cw) varies with **m**, **n** and the values of $\alpha$ and $\beta$. The function is quickest if either or both of $\alpha$ and $\beta$ are equal to zero, or plus or minus unity.

## 9 Example

The following program reads in a pair of matrices $A$ and **b**, along with values for **transa**, **transb**, **alpha** and **beta**, and adds them together, printing the result matrix $C$. The process is continued until the end of the input stream is reached.

## 9.1 Program Text

```
     function f01cw_example

fprintf('f01cw example results\n\n');

% Example 1: C = A + B
a = [ 1.0 + 2.0i   2.5 - 1.5i   2.5 - 1.0i;
     -2.0 - 2.0i   2.0 - 1.0i  -1.5 - 1.0i;
      3.5 - 1.5i   2.0 + 1.5i   2.0 + 3.0i;
     -2.5 + 0.0i  -3.0 + 2.5i  -2.0 + 2.0i];
b = [ 2.0 + 1.0i  -2.5 + 3.0i  -0.5 + 0.0i;
      1.0 + 0.0i   1.0 - 1.5i   1.5 - 1.5i;
     -1.5 - 0.5i   2.5 - 2.0i  -0.5 + 1.0i;
      2.5 - 1.5i  -1.0 + 1.5i   2.0 + 3.0i];

[m,n] = size(a);
m = nag_int(m);
n = nag_int(n);

transa = 'N';
transb = 'N';
alpha  = complex(1);
beta   = complex(1);
[c1, ifail] = f01cw( ...
                    transa, transb, m, n, alpha, a, beta, b);

disp('Example 1: C = A + B');
disp(c1);

% Example 2: C = A - B^T
a = [ 1.0 + 1.0i   2.5 - 1.5i   3.0 + 1.5i;
     -2.0 - 0.5i   2.0 + 1.5i  -1.5 - 2.5i];
b = [ 2.0 + 1.0i  -2.5 + 2.0i;
      1.0 + 0.0i   1.0 + 1.5i;
     -1.5 - 0.5i   2.5 - 1.0i];
[m,n] = size(a);
m = nag_int(m);
n = nag_int(n);

transa = 'N';
transb = 'T';
alpha  = complex(1);
beta   = complex(-1);
[c2, ifail] = f01cw( ...
                    transa, transb, m, n, alpha, a, beta, b);

disp('Example 2: C = A - B^T');
disp(c2);
```

## 9.2 Program Results

```
     f01cw example results

Example 1: C = A + B
   3.0000 + 3.0000i   0.0000 + 1.5000i   2.0000 - 1.0000i
  -1.0000 - 2.0000i   3.0000 - 2.5000i   0.0000 - 2.5000i
   2.0000 - 2.0000i   4.5000 - 0.5000i   1.5000 + 4.0000i
   0.0000 - 1.5000i  -4.0000 + 4.0000i   0.0000 + 5.0000i

Example 2: C = A - B^T
  -1.0000 + 0.0000i   1.5000 - 1.5000i   4.5000 + 2.0000i
   0.5000 - 2.5000i   1.0000 + 0.0000i  -4.0000 - 1.5000i
```