

## NAG Toolbox

### nag\_matop\_real\_addsub (f01ct)

## 1 Purpose

nag\_matop\_real\_addsub (f01ct) adds two double matrices, each one optionally transposed and multiplied by a scalar.

## 2 Syntax

```
[c, ifail] = nag_matop_real_addsub(transa, transb, m, n, alpha, a, beta, b)
[c, ifail] = f01ct(transa, transb, m, n, alpha, a, beta, b)
```

## 3 Description

nag\_matop\_real\_addsub (f01ct) performs one of the operations

$$\begin{aligned}C &:= \alpha A + \beta B, \\C &:= \alpha A^T + \beta B, \\C &:= \alpha A + \beta B^T \text{ or } \\C &:= \alpha A^T + \beta B^T,\end{aligned}$$

where  $A$ ,  $B$  and  $C$  are matrices, and  $\alpha$  and  $\beta$  are scalars. 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' or 'C'

The matrix will be transposed.

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

5: **alpha** – REAL (KIND=nag\_wp)

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

6: **a**( $lda, :)$  – REAL (KIND=nag\_wp) array

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

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

The second dimension of the array **a** must be at least  $\max(1, n)$  if **transa** = 'N', and at least  $\max(1, m)$  otherwise.

If  $\alpha = 0.0$ , the elements of array **a** need not be assigned. If  $\alpha \neq 0.0$ , then if **transa** = 'N', the leading  $m$  by  $n$  part of **a** must contain the matrix  $A$ , otherwise the leading  $n$  by  $m$  part of **a** must contain the matrix  $A$ .

7: **beta** – REAL (KIND=nag\_wp)

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

8: **b**( $ldb, :)$  – REAL (KIND=nag\_wp) array

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

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

The second dimension of the array **b** must be at least  $\max(1, n)$  if **transb** = 'N', and at least  $\max(1, m)$  otherwise.

If  $\beta = 0.0$ , the elements of array **b** need not be assigned. If  $\beta \neq 0.0$ , then if **transa** = 'N', the leading  $m$  by  $n$  part of **b** must contain the matrix  $B$ , otherwise the leading  $n$  by  $m$  part of **b** must contain the matrix  $B$ .

## 5.2 Optional Input Parameters

None.

## 5.3 Output Parameters

1: **c**( $ldc, :)$  – REAL (KIND=nag\_wp) array

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

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

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,  $ldc < \max(1, m)$ .

**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\_real\_addsub (f01ct) are accurate to **machine precision**.

## 8 Further Comments

The time taken for a call of nag\_matop\_real\_addsub (f01ct) 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 f01ct_example

fprintf('f01ct example results\n\n');

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

```

-1.5   2.5  -2.5;
 2.0  -2.0   1.0];
[m,n] = size(a);
m = nag_int(m);
n = nag_int(n);

transa = 'N';
transb = 'N';
alpha  = 1;
beta   = 1;
[c1, ifail] = f01ct( ...
    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   2.5   3.0   1.5   2.5;
      -2.0   2.0  -1.5  -2.0  -1.0;
       3.5   2.0  -2.5  -1.5   2.5];
b = [ 2.0  -2.5  -2.0;
       1.0   1.0   1.0;
     -1.5   2.5  -2.5;
       2.0  -2.0   1.0;
       1.0   1.0   2.5];
[m,n] = size(a);
m = nag_int(m);
n = nag_int(n);

transa = 'N';
transb = 'T';
alpha  = 1;
beta   = -1;
[c2, ifail] = f01ct( ...
    transa, transb, m, n, alpha, a, beta, b);

disp('Example 2: C = A - B''');
disp(c2);

```

## 9.2 Program Results

f01ct example results

Example 1: C = A + B

3.0000	0	1.0000
-1.0000	3.0000	-0.5000
2.0000	4.5000	-5.0000
3.5000	-4.0000	2.0000

Example 2: C = A - B'

-1.0000	1.5000	4.5000	-0.5000	1.5000
0.5000	1.0000	-4.0000	0	-2.0000
5.5000	1.0000	0	-2.5000	0

---