NAG Library Routine Document

F01CWF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F01CWF adds two complex matrices, each one optionally transposed and multiplied by a scalar.

2 Specification

```
SUBROUTINE F01CWF (TRANSA, TRANSB, M, N, ALPHA, A, LDA, BETA, B, LDB, C,    &
                   LDC, IFAIL)

INTEGER                 M, N, LDA, LDB, LDC, IFAIL
COMPLEX (KIND=nag_wp) ALPHA, A(LDA,*), BETA, B(LDB,*), C(LDC,*)
CHARACTER(1)            TRANSA, TRANSB
```

3 Description

F01CWF 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, α and β are scalars, T denotes transposition and H denotes conjugate transposition. For efficiency, the routine contains special code for the cases when one or both of α, β 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 Arguments

1: TRANSA – CHARACTER(1) *Input*

2: TRANSB – CHARACTER(1) *Input*

On entry: 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 *Input*

On entry: 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 ≥ 0.

4: N – INTEGER *Input*

On entry: 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 ≥ 0.

5: ALPHA – COMPLEX (KIND=nag_wp) *Input*

On entry: the scalar α, by which matrix A is multiplied before addition.

6: A(LDA, $*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the second dimension of the array A must be at least $\max(1, \mathrm{N})$ if ALPHA $\neq 0$ and TRANSA = 'N', $\max(1, \mathrm{M})$ if ALPHA $\neq 0$ and TRANSA = 'T' or 'C' and at least 1 if ALPHA = 0.

On entry: the matrix A. If $\alpha = 0$, the array A is not referenced.

7: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01CWF is called.

Constraints:

if TRANSA = 'N', LDA $\geq \max(1, \mathrm{M})$;
otherwise LDA $\geq \max(1, \mathrm{N})$.

8: BETA – COMPLEX (KIND=nag_wp) *Input*

On entry: the scalar β, by which matrix B is multiplied before addition.

9: B(LDB, $*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the second dimension of the array B must be at least $\max(1, \mathrm{N})$ if BETA $\neq 0$ and TRANSB = 'N', $\max(1, \mathrm{M})$ if BETA $\neq 0$ and TRANSB = 'T' or 'C' and at least 1 if BETA = 0.

On entry: the matrix B. If $\beta = 0$, the array B is not referenced.

10: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F01CWF is called.

Constraints:

if TRANSB = 'N', LDB $\geq \max(1, \mathrm{M})$;
otherwise LDB $\geq \max(1, \mathrm{N})$.

11: C(LDC, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array C must be at least max(1, N).

On exit: the elements of the m by n matrix C.

12: LDC – INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F01CWF is called.

Constraint: LDC $\geq$ max(1, M).

13: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

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.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The results returned by F01CWF are accurate to ***machine precision***.

8 Parallelism and Performance

F01CWF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken for a call of F01CWF varies with M, N and the values of α and β. The routine is quickest if either or both of α and β are equal to zero, or plus or minus unity.

10 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.

10.1 Program Text

```
    Program f01cwfe

!     F01CWF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f01cwf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: alpha, beta
      Integer                          :: i, ifail, lda, ldb, ldc, m, n,       &
                                          ncola, ncolb, nrowa, nrowb
      Character (1)                    :: transa, transb
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:,:), c(:,:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max
!     .. Executable Statements ..
      Write (nout,*) 'F01CWF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
100   Continue
!     Skip Subexample heading
      Read (nin,*,End=110)
      Read (nin,*) nrowa, ncola, transa, alpha
      Read (nin,*) nrowb, ncolb, transb, beta
      lda = max(nrowa,ncola)
      ldb = max(nrowb,ncolb)
```

```
      ldc = lda
      Allocate (a(lda,max(nrowa,ncola)),b(ldb,max(nrowb,                    &
        ncolb)),c(ldc,max(nrowa,ncola)))
!     Read matrices A and B.
      Do i = 1, nrowa
        Read (nin,*) a(i,1:ncola)
      End Do
      Do i = 1, nrowb
        Read (nin,*) b(i,1:ncolb)
      End Do
      If (transa=='N' .Or. transa=='n') Then
        m = nrowa
        n = ncola
      Else
        m = ncola
        n = nrowa
      End If

!     ifail: behaviour on error exit
!            =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
!     Add the two matrices A and B.
      Call f01cwf(transa,transb,m,n,alpha,a,lda,beta,b,ldb,c,ldc,ifail)

!     Print the result matrix C.
      Write (nout,99999) transa, transb
      Write (nout,99998) alpha, beta
      Flush (nout)
      Call x04daf('G','X',m,n,c,ldc,'Matrix C:',ifail)
      Write (nout,*)
      Deallocate (a,b,c)
      Go To 100
110   Continue

99999 Format (1X,'TRANSA = ''',A,''', TRANSB = ''',A,''',')
99998 Format (1X,'ALPHA = (',1P,E11.4,',',E11.4,'), BETA = (',E11.4,',',E11.4, &
        ')')
    End Program f01cwfe
```

10.2 Program Data

```
F01CWF Example Program Data
Example 1:
4 3 'N'   (1.0, 0.0)                          : nrowa, ncola, transa, alpha
4 3 'N'   (1.0, 0.0)                          : nrowb, ncolb, transb, beta
  ( 1.0, 2.0)  ( 2.5,-1.5)  ( 2.5,-1.0)      : Matrix A
  (-2.0,-2.0)  ( 2.0,-1.0)  (-1.5,-1.0)
  ( 3.5,-1.5)  ( 2.0, 1.5)  ( 2.0, 3.0)
  (-2.5, 0.0)  (-3.0, 2.5)  (-2.0, 2.0)
  ( 2.0, 1.0)  (-2.5, 3.0)  (-0.5, 0.0)      : Matrix B
  ( 1.0, 0.0)  ( 1.0,-1.5)  ( 1.5,-1.5)
  (-1.5,-0.5)  ( 2.5,-2.0)  (-0.5, 1.0)
  ( 2.5,-1.5)  (-1.0, 1.5)  ( 2.0, 3.0)
Example 2:
2 3 'N'   (1.0, 0.0)                          : nrowa, ncola, transa, alpha
3 2 'T'  (-1.0, 0.0)                          : nrowb, ncolb, transb, beta
  ( 1.0, 1.0)  ( 2.5,-1.5)  ( 3.0, 1.5)      : Matrix A
  (-2.0,-0.5)  ( 2.0, 1.5)  (-1.5,-2.5)
  ( 2.0, 1.0)  (-2.5, 2.0)                    : Matrix B
  ( 1.0, 0.0)  ( 1.0, 1.5)
  (-1.5,-0.5)  ( 2.5,-1.0)
```

10.3 Program Results

```
 F01CWF Example Program Results

 TRANSA = 'N', TRANSB = 'N',
 ALPHA = ( 1.0000E+00, 0.0000E+00), BETA = ( 1.0000E+00, 0.0000E+00)
 Matrix C:
             1          2          3
```

```
 1      3.0000     0.0000     2.0000
        3.0000     1.5000    -1.0000

 2     -1.0000     3.0000     0.0000
       -2.0000    -2.5000    -2.5000

 3      2.0000     4.5000     1.5000
       -2.0000    -0.5000     4.0000

 4      0.0000    -4.0000     0.0000
       -1.5000     4.0000     5.0000

 TRANSA = 'N', TRANSB = 'T',
 ALPHA = ( 1.0000E+00, 0.0000E+00), BETA = (-1.0000E+00, 0.0000E+00)
 Matrix C:
             1          2          3
 1     -1.0000     1.5000     4.5000
        0.0000    -1.5000     2.0000

 2      0.5000     1.0000    -4.0000
       -2.5000     0.0000    -1.5000
```