NAG Library Routine Document

F01FPF

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

F01FPF computes the principal matrix square root, $A^{1/2}$, of a complex upper triangular n by n matrix A.

2 Specification

```
SUBROUTINE F01FPF (N, A, LDA, IFAIL)

INTEGER               N, LDA, IFAIL
COMPLEX (KIND=nag_wp) A(LDA,*)
```

3 Description

A square root of a matrix A is a solution X to the equation $X^2 = A$. A nonsingular matrix has multiple square roots. For a matrix with no eigenvalues on the closed negative real line, the principal square root, denoted by $A^{1/2}$, is the unique square root whose eigenvalues lie in the open right half-plane.

F01FPF computes $A^{1/2}$, where A is an upper triangular matrix. $A^{1/2}$ is also upper triangular.

The algorithm used by F01FPF is described in BjÎrck and Hammarling (1983). In addition a blocking scheme described in Deadman *et al.* (2013) is used.

4 References

BjÎrck Ð and Hammarling S (1983) A Schur method for the square root of a matrix *Linear Algebra Appl.* **52/53** 127–140

Deadman E, Higham N J and Ralha R (2013) Blocked Schur Algorithms for Computing the Matrix Square Root *Applied Parallel and Scientific Computing: 11th International Conference, (PARA 2012, Helsinki, Finland)* P. Manninen and P. Úster, Eds *Lecture Notes in Computer Science* **7782** 171–181 Springer–Verlag

Higham N J (2008) *Functions of Matrices: Theory and Computation* SIAM, Philadelphia, PA, USA

5 Arguments

1: N – INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: $N \geq 0$.

2: $A(LDA, *)$ – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least N.

On entry: the n by n upper triangular matrix A.

On exit: contains, if $IFAIL = 0$, the n by n principal matrix square root, $A^{1/2}$. Alternatively, if $IFAIL = 1$, contains an n by n non-principal square root of A.

3: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01FPF is called.

Constraint: LDA $\geq$ N.

4: 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 $= 1$

A has negative or semisimple, vanishing eigenvalues. The principal square root is not defined in this case; a non-principal square root is returned.

IFAIL $= 2$

A has a defective vanishing eigenvalue. The square root cannot be found in this case.

IFAIL $= 3$

An internal error occurred. It is likely that the routine was called incorrectly.

IFAIL $= -1$

On entry, N $= \langle$*value*$\rangle$.
Constraint: N ≥ 0.

IFAIL $= -3$

On entry, LDA $= \langle$*value*$\rangle$ and N $= \langle$*value*$\rangle$.
Constraint: LDA $\geq$ N.

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 computed square root $\hat{X}$ satisfies $\hat{X}^2 = A + \Delta A$, where $|\Delta A| \approx O(\epsilon)n|\hat{X}|^2$, where ϵ is ***machine precision***. The order of the change in A is to be interpreted elementwise.

8 Parallelism and Performance

F01FPF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01FPF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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 cost of the algorithm is $n^3/3$ complex floating-point operations; see Algorithm 6.3 in Higham (2008). $O(2 \times n^2)$ of complex allocatable memory is required by the routine.

If A is a full matrix, then F01FNF should be used to compute the principal square root.

If condition number and residual bound estimates are required, then F01KDF should be used. For further discussion of the condition of the matrix square root see Section 6.1 of Higham (2008).

10 Example

This example finds the principal matrix square root of the matrix

$$A = \begin{pmatrix} 2i & 14+2i & 12+3i & 6+4i \\ 0 & -5+12i & 6+18i & 9+16i \\ 0 & 0 & 3-4i & 16-4i \\ 0 & 0 & 0 & 4 \end{pmatrix}.$$

10.1 Program Text

```
    Program f01fpfe

!     F01FPF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f01fpf, nag_wp, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lda, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: cmplx
!     .. Executable Statements ..
```

```
      Write (nout,*) 'F01FPF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n

      lda = n
      Allocate (a(lda,n))
      a(1:lda,1:n) = cmplx(0,kind=nag_wp)

!     Read A from data file
      Read (nin,*)(a(i,i:n),i=1,n)

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0

!     Find sqrt(A)
      Call f01fpf(n,a,lda,ifail)

!     Print solution
      If (ifail==0) Then
        ifail = 0
        Call x04daf('G','N',n,n,a,lda,'sqrt(A)',ifail)
      End If

    End Program f01fpfe
```

10.2 Program Data

```
F01FPF Example Program Data

4                                                     :Value of N

( 0.0,  2.0) ( 14.0,  2.0) ( 12.0,  3.0) ( 6.0, 4.0)
(-5.0, 12.0) (  6.0, 18.0) (  9.0, 16.0)
( 3.0, -4.0) ( 16.0, -4.0)
( 4.0,  0.0)                                          :End of matrix A
```

10.3 Program Results

```
 F01FPF Example Program Results

 sqrt(A)
               1          2          3          4
 1       1.0000     2.0000    -0.0000     1.0000
         1.0000    -2.0000     1.0000    -1.0000

 2       0.0000     2.0000     3.0000     0.0000
         0.0000     3.0000     3.0000     1.0000

 3       0.0000     0.0000     2.0000     4.0000
         0.0000     0.0000    -1.0000     0.0000

 4       0.0000     0.0000     0.0000     2.0000
         0.0000     0.0000     0.0000     0.0000
```