

# NAG Library Routine Document

## F01FQF

**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

F01FQF computes an arbitrary real power  $A^p$  of a complex  $n$  by  $n$  matrix  $A$ .

### 2 Specification

```
SUBROUTINE F01FQF (N, A, LDA, P, IFAIL)
  INTEGER          N, LDA, IFAIL
  REAL (KIND=nag_wp) P
  COMPLEX (KIND=nag_wp) A(LDA,*)
```

### 3 Description

For a matrix  $A$  with no eigenvalues on the closed negative real line,  $A^p$  ( $p \in \mathbb{R}$ ) can be defined as

$$A^p = \exp(p \log(A))$$

where  $\log(A)$  is the principal logarithm of  $A$  (the unique logarithm whose spectrum lies in the strip  $\{z : -\pi < \text{Im}(z) < \pi\}$ ).

$A^p$  is computed using the Schur–Padé algorithm described in Higham and Lin (2011) and Higham and Lin (2013).

The real number  $p$  is expressed as  $p = q + r$  where  $q \in (-1, 1)$  and  $r \in \mathbb{Z}$ . Then  $A^p = A^q A^r$ . The integer power  $A^r$  is found using a combination of binary powering and, if necessary, matrix inversion. The fractional power  $A^q$  is computed using a Schur decomposition and a Padé approximant.

### 4 References

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

Higham N J and Lin L (2011) A Schur–Padé algorithm for fractional powers of a matrix *SIAM J. Matrix Anal. Appl.* **32(3)** 1056–1078

Higham N J and Lin L (2013) An improved Schur–Padé algorithm for fractional powers of a matrix and their Fréchet derivatives *MIMS Eprint 2013.1* Manchester Institute for Mathematical Sciences, School of Mathematics, University of Manchester <http://eprints.ma.man.ac.uk/>

### 5 Parameters

- 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  $n$  matrix  $A$ .  
*On exit:* if IFAIL = 0, the  $n$  by  $n$  matrix  $p$ th power,  $A^p$ . Alternatively, if IFAIL = 1, contains an  $n$  by  $n$  non-principal power of  $A$ .

- 3: LDA – INTEGER *Input*  
*On entry:* the first dimension of the array *A* as declared in the (sub)program from which F01FQF is called.  
*Constraint:*  $LDA \geq N$ .
- 4: P – REAL (KIND=nag\_wp) *Input*  
*On entry:* the required power of *A*.
- 5: IFAIL – INTEGER *Input/Output*  
*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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 eigenvalues on the negative real line. The principal *p*th power is not defined so a non-principal power is returned.

IFAIL = 2

*A* is singular so the *p*th power cannot be computed.

IFAIL = 3

$A^p$  has been computed using an IEEE double precision Padé approximant, although the arithmetic precision is higher than IEEE double precision.

IFAIL = 4

An unexpected internal error occurred. This failure should not occur and suggests that the routine has been called incorrectly.

IFAIL = -1

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

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 3.8 in the Essential Introduction for further information.

IFAIL = -399

Your licence key may have expired or may not have been installed correctly.  
See Section 3.7 in the Essential Introduction for further information.

IFAIL = -999

Dynamic memory allocation failed.  
See Section 3.6 in the Essential Introduction for further information.

## 7 Accuracy

For positive integer  $p$ , the algorithm reduces to a sequence of matrix multiplications. For negative integer  $p$ , the algorithm consists of a combination of matrix inversion and matrix multiplications.

For a normal matrix  $A$  (for which  $A^H A = A A^H$ ) and non-integer  $p$ , the Schur decomposition is diagonal and the algorithm reduces to evaluating powers of the eigenvalues of  $A$  and then constructing  $A^p$  using the Schur vectors. This should give a very accurate result. In general however, no error bounds are available for the algorithm.

## 8 Parallelism and Performance

F01FQF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F01FQF 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  $O(n^3)$ . The exact cost depends on the matrix  $A$  but if  $p \in (-1, 1)$  then the cost is independent of  $p$ .  $O(4 \times n^2)$  complex allocatable memory is required by the routine.

If estimates of the condition number of  $A^p$  are required then F01KEF should be used.

## 10 Example

This example finds  $A^p$  where  $p = 0.2$  and

$$A = \begin{pmatrix} 2 & 3 & 2 & 1 + 3i \\ 2 + i & 1 & 1 & 2 + 2i \\ 2 + i & 2 + 2i & 2i & 2 + 4i \\ 3 & 2 + 2i & 3 & 1 \end{pmatrix}.$$

### 10.1 Program Text

```

Program f01fqfe

!      F01FQF Example Program Text
!
!      Mark 25 Release. NAG Copyright 2014.
!
!      .. Use Statements ..
!      Use nag_library, Only: f01fqf, nag_wp, x04daf
!      .. Implicit None Statement ..
!      Implicit None
!      .. Parameters ..

```

```

Integer, Parameter          :: nin = 5, nout = 6
! .. Local Scalars ..
Real (Kind=nag_wp)         :: p
Integer                     :: i, ifail, lda, n
! .. Local Arrays ..
Complex (Kind=nag_wp), Allocatable :: a(:, :)
! .. Executable Statements ..
Write (nout,*) 'F01FQF Example Program Results'
Write (nout,*)
Flush (nout)
! Skip heading in data file
Read (nin,*)
Read (nin,*) n, p
lda = n
Allocate (a(lda,n))
! Read A from data file
Read (nin,*)(a(i,1:n),i=1,n)

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

! Find A^p
Call f01fqf(n,a,lda,p,ifail)

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

End Program f01fqfe

```

## 10.2 Program Data

F01FQF Example Program Data

```

4      0.2                               : Values of N and P

(2.0,0.0)  (3.0,0.0)  (2.0,0.0)  (1.0,3.0)
(2.0,1.0)  (1.0,0.0)  (1.0,0.0)  (2.0,2.0)
(2.0,1.0)  (2.0,2.0)  (0.0,2.0)  (2.0,4.0)
(3.0,0.0)  (2.0,2.0)  (3.0,0.0)  (1.0,0.0) : End of matrix A

```

## 10.3 Program Results

F01FQF Example Program Results

```

A^p
      1      2      3      4
1     1.1766  0.1375  0.2742 -0.1435
     -0.0758  0.2241 -0.2223  0.0816

2     0.2074  1.1118 -0.1343  0.1794
     -0.0998 -0.0039 -0.0404  0.3590

3     -0.0859  0.5224  1.0616  0.2308
     -0.0824 -0.0530  0.3921  0.1856

4     0.3313 -0.1507  0.2178  1.1710
     0.1303  0.0982 -0.0061 -0.2136

```

---