# F04JZFP

# NAG Parallel Library Routine Document

**Note:** before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

## 1 Description

F04JZFP solves a complex Hermitian tridiagonal positive-definite linear system of equations

$$AX = B;$$

with multiple right-hand sides, using a Cholesky factorization, where $A$ and $B$ are $n$ by $n$ and $n$ by $r$ matrices respectively.

The routine first computes a Cholesky factorization of $A$ as $A = PU^HUP^T$, where $P$ is a permutation matrix, and $U$ is a tridiagonal upper triangular matrix. Then forward and backward substitutions are used to compute $X$.

## 2 Specification

```
 SUBROUTINE F04JZFP(ICNTXT, N, NCR, D, E, NRHS, B, LDB, AF, LAF,
1                   WORK, LWORK, IFAIL)
 DOUBLE PRECISION   D(*)
 COMPLEX*16         E(*), B(LDB,NRHS), AF(LAF), WORK(LWORK)
 INTEGER            ICNTXT, N, NCR, NRHS, LDB, LAF, LWORK, IFAIL
```

## 3 Usage

### 3.1 Definitions

The following definitions are used in describing the data distribution within this document:

| | | |
|---|---|---|
| $m_p$ | – | the number of rows in the Library Grid, for this routine $m_p = 1$ or $m_p = p$; |
| $n_p$ | – | the number of columns in the Library Grid, for this routine $n_p = 1$ or $n_p = p$. |
| $p$ | – | $m_p \times n_p$, the total number of processors in the Library Grid. |
| $N_b$ | – | the maximum number of columns of the matrix $A$, or the maximum number of rows of the matrix $B$ held locally on a logical processor; |
| $N_x$ | – | the number of columns of the matrix $A$ and the number of rows of the matrix $B$ held locally on a logical processor, where $0 \leq N_x \leq N_b$. |
| $\lceil x \rceil$ | – | the ceiling function of $x$, which gives the smallest integer which is not less than $x$. |

### 3.2 Global and Local Arguments

The following global **input** argument(s) must have the same value on entry to the routine on each processor and the global **output** argument(s) will have the same value on exit from the routine on each processor:

Global input arguments: N, NRHS, IFAIL.

Global output arguments: IFAIL.

The remaining arguments are local.

### 3.3 Distribution Strategy

The matrix $A$ is represented by two vectors $e$ (off-diagonal elements) and $d$ (diagonal elements). These vectors should be distributed over a one-dimensional array of processors, assuming a column block distribution (see the F04 Chapter Introduction). The right-hand sides of the equation are stored in the array B in a row block distribution. Each logical processor contains at most $N_b = \lceil n/p \rceil$ columns

of the matrix $A$ or rows of the right-hand side $B$. Some logical processor may not contain any columns of $A$ or rows) of $B$ if $n$ is not large relative to $p$, but if $n > (p-1)^2$ then all processors will certainly contain some columns and rows of $A$ and $B$ respectively. The mapping for matrices is blocked, reflecting the nature of the **divide and conquer algorithm** as a task-parallel algorithm, see Section 6.2.

### 3.4 Related Routines

The Library provides many support routines for the generation/distribution and input/output of data in column or row block form.

The following routines may be used in conjunction with F04JZFP :

| | |
|---|---|
| Complex matrix generation: | column block distribution : F01ZWFP |
| Complex matrix generation: | row block distribution : F01ZNFP |
| Complex matrix output: | row block distribution : X04BUFP |

## 4 Arguments

**1:** ICNTXT — INTEGER *Local Input*

*On entry:* the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP.

**Note:** the value of ICNTXT **must not** be changed.

**2:** N — INTEGER *Global Input*

*On entry:* $n$ the order of the matrix $A$.

*Constraint:* N $\geq 1$.

**3:** NCR — INTEGER *Local Output*

*On exit:* $N_x$ the number of columns of the matrix $A$ and of rows of the matrix $B$ held on the logical processor.

**4:** D($*$) — DOUBLE PRECISION array *Local Input/Local Output*

**Note:** the dimension of array D must be at least $N_b$.

*On entry:* the local part of the distributed vector $d$ which contains the diagonal elements of the matrix $A$.

*On exit:* the permuted diagonal part of the factorized matrix $A$, using the same distribution strategy.

**5:** E($*$) — COMPLEX*16 array *Local Input/Local Output*

**Note:** the dimension of array E must be at least $N_b$.

*On entry:* the local part of the distributed vector $e$ which contains the upper diagonal elements of the matrix $A$. E should be aligned with D, and E(N) may be set to 0.

*On exit:* the factor $U$ from the Cholesky factorisation of $A = PU^HUP^T$, using the same distribution strategy.

**6:** NRHS — INTEGER *Global Input*

*On entry:* $r$, the number of right-hand sides.

*Constraint:* NRHS $\geq 1$.

**7:** B(LDB, NRHS) — COMPLEX*16 array *Local Input/Local Output*

*On entry:* the local part of the right-hand side $B$ which is stored in row block fashion.

*On exit:* the $n$ by $r$ solution matrix $X$ distributed in the same row block distribution.

**8:** LDB — INTEGER *Local Input*

*On entry:* the size of the first dimension of the array B as declared in the (sub)program from which F04JZFP is called.

*Constraint:* LDB $\geq N_b$.

**9:** AF(LAF) — COMPLEX*16 array *Local Output*

*On exit:* the auxiliary fill-in space. Fill-in is created and stored during the factorisation. If LAF is not large enough, an error code will be returned and the minimum acceptable value of LAF will be returned in AF(1).

**10:** LAF — INTEGER *Local Input*

*On entry:* the dimension of the array AF.

*Constraint:* LAF $\geq 12\times p + 3 \times N_b$.

**11:** WORK(LWORK) — COMPLEX*16 array *Local Workspace*

*On exit:* if LWORK is not large enough, an error code will be returned and the minimum accpetable value of LWORK will be returned in the real part of WORK(1).

**12:** LWORK — INTEGER *Local Input*

**Note:** this routine provides a facility to obtain the minimal size of the array WORK by setting LWORK = 1. In that case LWORK must be treated as a **global** input and INT(WORK(1)), on exit provides the optimal size of the array WORK. However no other useful computations are performed by the routine.

*On entry:* size of the workspace array WORK.

*Constraint:*

LWORK $\geq \max(8\times p,(10+2\times\min(100,r))\times p+4\times r)$,
or LWORK = 1.

**13:** IFAIL — INTEGER *Global Input/Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this argument (described in the Essential Introduction) the recommended values are:

IFAIL = 0, if multigridding is **not** employed;
IFAIL = −1, if multigridding is employed.

*On exit:* IFAIL = 0 (or −9999 if reduced error checking is enabled) unless the routine detects an error (see Section 5).

# 5 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output from the root processor (or processor {0,0} when the root processor is not available) on the current error message unit (as defined by X04AAF).

### 5.1 Full Error Checking Mode Only

IFAIL $= -2000$

The routine has been called with an invalid value of ICNTXT on one or more processors.

IFAIL $= -1000$

The logical processor grid and library mechanism (Library Grid) have not been correctly defined, see Z01AAFP.

IFAIL $= -i$

On entry, one of the arguments was invalid:

if the $k$th argument is a scalar IFAIL $= -k$;
if the $k$th argument is an array and its $j$th element is invalid, IFAIL $= -(100 \times k + j)$.

This error occured either because a global argument did not have the same value on all logical processors, or because its value on one or more processors was incorrect. An explanatory message distinguishes between these two cases.

### 5.2 Any Error Checking Mode

IFAIL $> 0$

If IFAIL $= k \leq p$ the submatrix stored and factored locally on processor $\{0, k\}$ or $\{k, 0\}$ was not positive definite, and the factorization was not completed. If IFAIL $= k > p$ the submatrix stored on processor $\{0, (k-p)\}$ or $\{(k-p), 0\}$ representing interactions with other processors was not positive definite and the factorization was not completed.

## 6 Further Comments

The total number of floating-point operations is approximately $4n + 4nr$.

### 6.1 Algorithmic Detail

The matrix $A$ is first factorized using the Cholesky algorithms. Then forward and backward substitutions are used to calculate the solution. Assuming the decomposition of the matrix $A = PU^H UP^H$, where $P$ is a permutation matrix and $U$ is upper triangular. Then the solution $X$ is computed by solving $PU^H Y = B$ and then $UP^H X = Y$.

### 6.2 Parallelism Detail

This routine uses a divide and conquer algorithm for the factorisation of the matrix. This algorithm is well suited for narrow-band matrices. The matrix is distributed one-dimensionally, with rows divided amongst the processes. Hence the matrix is divided into a few pieces (usually $p$, with one stored on each processor) formed by some of its rows and then the algorithm proceeds in two phases:

(1) Local phase : The individual pieces (in fact only the diagonal blocks of the matrix) are factorized independently and in parallel. These factors are applied to the matrix creating fill-in, which is stored in a non-inspectable way in the array AF. Mathematically, this is equivalent to reordering the matrix $A$ as $PAP^T$ and then factorizing the principal leading submatrix of size equal to the sum of the sizes of the matrices factorized on each process.

(2) Reduced system phase : A small $(p-1) \times (p-1)$ system is formed representing interaction of the larger blocks, and is stored (as are its factors) in the space AF. A parallel **block cyclic reduction** algorithm is then used to complete the factorization.

It is also important to note that the block size $N_b$ should not be to small. Otherwise the divide and conquer algorithm performs poorly.

The Level-3 BLAS operations used in this routine are carried out in parallel.

### 6.3 Accuracy

For each right-hand side vector $b$, the computed solution $x$ is the exact solution of a perturbed system of equations $(A + E)x = b$, where

$$\|E\| \leq \epsilon c(n)\|A\|,$$

$c(n)$ is a modest function of $n$, $\epsilon$ is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution $x$ satisfies the bound

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq c(n)\text{cond}(A)\epsilon$$

where $\text{cond}(A) = \|A\| \cdot \|A^{-1}\|$.

## 7 References

**[1]** Blackford L S, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D and Whaley R C (1997) ScaLAPACK Users' Guide *SIAM* 3600 University City Science Center, Philadelpia, PA 19104-2688, USA. URL: http://www.netlib.org/scalapack/slug/scalapack_slug.html

**[2]** Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia

**[3]** Golub G H and van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

## 8 Example

This example is from the numerical solution of the Laplacian equation in one space dimension. The differential equation has the form :

$$\begin{cases} -\Delta u &= (1+2i)\pi^2 sin(\pi x) \text{ in } \Omega, \\ u &= 0 \quad \text{on } \Gamma; \end{cases}$$

where $\Omega = ]0;1[$, $\Gamma = \partial\Omega = \{0;1\}$ and with the true solution $u(x) = (1+2i)sin(\pi x)$.

This problem is solved numerically using a three-point finite difference scheme. The matrix $A$ has the form :

$$\frac{1}{h^2} \times \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

In this example the mesh has $N = 12$ inner points and its size is $h = 1/(N+1)$ (the precision of the computed solution is $O(h^2)$ in the Euclidian norm). The example prints the solution computed by solving the linear equations and the true solution $U(x)$, together with the 2-norm of the error and $h^2$.

### 8.1 Example Text

```
*     F04JZFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=15)
```

```
      INTEGER          MG, NG
      PARAMETER        (MG=1,NG=4)
      INTEGER          TDA, NBMAX, TDB, LDB
      PARAMETER        (TDA=NMAX/(MG*NG),NBMAX=TDA,TDB=TDA,LDB=TDB)
      INTEGER          LAF, NRHSMAX, LW
      PARAMETER        (LAF=12*NG+3*NBMAX,NRHSMAX=2,LW=(10+2*NRHSMAX)
     +                 *4+4*NRHSMAX)
*     .. Scalars in Common ..
      INTEGER          N
*     .. Local Scalars ..
      DOUBLE PRECISION ERROR, ERROR0, ERRORP, ERRORP0
      INTEGER          I, ICNTXT, ICOFF, IFAIL, INFO, LWORK, MP, MX,
     +                 MYCOL, MYROW, NB, NCR, NCX, NP, NRHS
      LOGICAL          ROOT
      CHARACTER        CNUMOP, TITOP
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      COMPLEX*16       AF(LAF), B(LDB,NRHSMAX), E(TDA),
     +                 SOL(LDB,NRHSMAX), W(LDB,NRHSMAX), WORK(LW)
      DOUBLE PRECISION D(TDA)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         DGERV2D, DGESD2D, EXACT, F01ZNFP, F01ZRFP,
     +                 F01ZWFP, F04JZFP, GD, GE, GRHSB, X04BUFP,
     +                 Z01AAFP, Z01ABFP, Z01ZAFP
*     .. Intrinsic Functions ..
      INTRINSIC        DCONJG, DBLE, DSQRT
*     .. Common blocks ..
      COMMON           /DIM/N
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F04JZFP Example Program Results'
*
      MP = MG
      NP = NG
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Obtain information about this processor and ensure that all
*     computations are within the Library Grid
*
      CALL Z01ZAFP(ICNTXT,MP,NP,MYROW,MYCOL)
      IF (MYROW.LT.MP .AND. MYCOL.LT.NP) THEN
*
*        Initialization of Data
*
         N = 12
         NB = N/(MG*NG)
         LWORK = LW
         NRHS = 1
         FORMAT = '(F12.4)'
         INFO = -1
*
         IF (N.LE.NMAX .AND. NRHS.LE.NRHSMAX .AND. NB.LE.NBMAX) THEN
*
```

```
            CALL Z01ZAFP(ICNTXT,MP,NP,MYROW,MYCOL)
*
*           Generation of the tridiagonal matrix A
*
            IFAIL = 0
            CALL F01ZRFP(ICNTXT,GD,1,N,D,1,NCX,IFAIL)
*
            IFAIL = 0
            CALL F01ZWFP(ICNTXT,GE,1,N,E,1,NCX,IFAIL)
*
*           Generation of RHS
*
            IFAIL = 0
            CALL F01ZNFP(ICNTXT,GRHSB,N,NRHS,B,LDB,MX,IFAIL)
*
*           Generation of the exact solution of the PDE
*
            IFAIL = 0
            CALL F01ZNFP(ICNTXT,EXACT,N,NRHS,SOL,LDB,MX,IFAIL)
*
*           Solve AX=B
*
            CALL F04JZFP(ICNTXT,N,NCR,D,E,NRHS,B,LDB,AF,LAF,WORK,LWORK,
     +                   INFO)
*
            IF (INFO.EQ.0) THEN
*
*              Compute the error
*
               ERROR = 0.D0
               ERROR0 = 0.D0
               DO 20 I = 1, NB
                  ERROR = ERROR + (B(I,1)-SOL(I,1))*DCONJG(B(I,1)
     +                    -SOL(I,1))
                  ERROR0 = ERROR0 + SOL(I,1)*DCONJG(SOL(I,1))
   20          CONTINUE
*
               IF (MYCOL.GT.0) THEN
                  CALL DGESD2D(ICNTXT,1,1,ERROR,1,0,0)
                  CALL DGESD2D(ICNTXT,1,1,ERROR0,1,0,0)
               END IF
*
*              Print Solution and the error
*
               IF (ROOT) THEN
                  DO 40 I = 1, NP - 1
                     CALL DGERV2D(ICNTXT,1,1,ERRORP,1,0,I)
                     CALL DGERV2D(ICNTXT,1,1,ERRORP0,1,0,I)
                     ERROR = ERROR + ERRORP
                     ERROR0 = ERROR0 + ERRORP0
   40             CONTINUE
*
                  ERROR = ERROR/ERROR0
                  ERROR = DSQRT(ERROR)
*
                  WRITE (NOUT,*)
                  WRITE (NOUT,*)
     +              ' Computed solution of differential equation'
```

```
                  WRITE (NOUT,*)
*
                  TITOP = 'Y'
                  CNUMOP = 'L'
               END IF
*
               IFAIL = 0
               ICOFF = 0
*
               CALL X04BUFP(ICNTXT,NOUT,MX,NRHS,B,LDB,FORMAT,TITOP,
     +                     CNUMOP,ICOFF,W,LDB,IFAIL)
*
               IF (ROOT) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,*)
     +              ' Exact solution of differential equation'
                  WRITE (NOUT,*)
               END IF
*
               CALL X04BUFP(ICNTXT,NOUT,MX,NRHS,SOL,LDB,FORMAT,TITOP,
     +                     CNUMOP,ICOFF,W,LDB,IFAIL)
*
               IF (ROOT) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) ' L2 error and H^2 ', ERROR,
     +              1.D0/(DBLE(N+1)*DBLE(N+1))
               END IF
*
            ELSE IF (INFO.GT.0) THEN
               IF (ROOT) WRITE (NOUT,*)
     +             'Matrix is not positive-definite'
            END IF
*
         END IF
      END IF
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
*
      SUBROUTINE GRHSB(J1,JL,NRHS,BL,LDBL)
*
*     This routine generates the right-hand side
*
*     .. Scalar Arguments ..
      INTEGER          J1, JL, LDBL, NRHS
*     .. Array Arguments ..
      COMPLEX*16       BL(LDBL,*)
*     .. Scalars in Common ..
      INTEGER          N
*     .. Local Scalars ..
      DOUBLE PRECISION H, PI, PI2
      INTEGER          J, K
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
```

```
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE, DSIN
*     .. Common blocks ..
      COMMON           /DIM/N
*     .. Executable Statements ..
      K = 1
      PI = X01AAF(0.0D0)
      PI2 = PI*PI
      H = 1.D0/DBLE(N+1)
      DO 20 J = J1, JL
         BL(K,1) = H*H*PI2*DSIN(DBLE(J)*PI*H)*(1.0D0,2.0D0)
         K = K + 1
   20 CONTINUE
*
*     End of GRHSB
*
      RETURN
      END
*
      SUBROUTINE EXACT(J1,JL,NRHS,BL,LDBL)
*
*     This routine generates the exact solution of the differential
*     equation
*
*     .. Scalar Arguments ..
      INTEGER          J1, JL, LDBL, NRHS
*     .. Array Arguments ..
      COMPLEX*16       BL(LDBL,*)
*     .. Scalars in Common ..
      INTEGER          N
*     .. Local Scalars ..
      DOUBLE PRECISION H, PI
      INTEGER          J, K
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE, DSIN
*     .. Common blocks ..
      COMMON           /DIM/N
*     .. Executable Statements ..
      K = 1
      PI = X01AAF(0.0D0)
      H = 1.0D0/DBLE(N+1)
      DO 20 J = J1, JL
         BL(K,1) = DSIN(DBLE(J)*PI*H)*(1.0D0,2.0D0)
         K = K + 1
   20 CONTINUE
*
*     End of EXACT
*
      RETURN
      END
*
      SUBROUTINE GD(M,J1,JL,BL,LDBL)
*
*     This routine generates the diagonal, D, of the tridiagonal matrix
*
```

```
*     .. Scalar Arguments ..
      INTEGER       J1, JL, LDBL, M
*     .. Array Arguments ..
      DOUBLE PRECISION BL(*)
*     .. Local Scalars ..
      INTEGER       J, K
*     .. Executable Statements ..
      K = 1
      DO 20 J = J1, JL
         BL(K) = 2.0D0
         K = K + 1
   20 CONTINUE
*
*     End of GD
*
      RETURN
      END
*
      SUBROUTINE GE(M,J1,JL,BL,LDBL)
*
*     This routine generates the vector E, the off-diagonal elements of
*     the tridiagonal matrix
*
*     .. Scalar Arguments ..
      INTEGER       J1, JL, LDBL, M
*     .. Array Arguments ..
      COMPLEX*16    BL(*)
*     .. Local Scalars ..
      INTEGER       J, K
*     .. Executable Statements ..
      K = 1
      DO 20 J = J1, JL
         BL(K) = -1.0D0
         K = K + 1
   20 CONTINUE
*
*     End of GE
*
      RETURN
      END
```

## 8.2 Example Data

None.

## 8.3 Example Results

```
 F04JZFP Example Program Results

  Computed solution of differential equation

   Array from logical processor    0,   0

                              1
   (      0.2405,      0.4810)
   (      0.4670,      0.9340)
   (      0.6664,      1.3327)
```

```
 Array from logical processor    0,   1

                           1
(      0.8270,      1.6540)
(      0.9396,      1.8792)
(      0.9976,      1.9951)

 Array from logical processor    0,   2

                           1
(      0.9976,      1.9951)
(      0.9396,      1.8792)
(      0.8270,      1.6540)

 Array from logical processor    0,   3

                           1
(      0.6664,      1.3327)
(      0.4670,      0.9340)
(      0.2405,      0.4810)


Exact solution of differential equation

 Array from logical processor    0,   0

                           1
(      0.2393,      0.4786)
(      0.4647,      0.9294)
(      0.6631,      1.3262)

 Array from logical processor    0,   1

                           1
(      0.8230,      1.6460)
(      0.9350,      1.8700)
(      0.9927,      1.9854)

 Array from logical processor    0,   2

                           1
(      0.9927,      1.9854)
(      0.9350,      1.8700)
(      0.8230,      1.6460)

 Array from logical processor    0,   3

                           1
(      0.6631,      1.3262)
(      0.4647,      0.9294)
(      0.2393,      0.4786)


L2 error and H^2   4.880912516310382E-003  5.917159763313609E-003
```