F07JRFP (PZPTTRF)

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

F07JRFP (PZPTTRF) computes the Cholesky factorization of an n by n complex tridiagonal Hermitian positive-definite matrix A_s, where A_s is a submatrix of a larger n by n_A matrix A, i.e.,

$$A_s(1:n, 1:n) \equiv A(1:n, j_A : j_A + n - 1).$$

Note: if $j_A = 1$ and $n = n_A$, then $A_s = A$.

The factorization has the form $A_s = PLL^H P^T$ or $A_s = PU^H U P^T$, where P is a permutation matrix, U is a complex tridiagonal upper triangular matrix and L is a complex tridiagonal lower triangular matrix, where $L = U^H$.

2 Specification

```
SUBROUTINE F07JRFP(N, D, E, JA, IDESCA, AF, LAF, WORK, LWORK, INFO)
ENTRY      PZPTTRF(N, D, E, JA, IDESCA, AF, LAF, WORK, LWORK, INFO)
INTEGER            N, JA, IDESCA(*), LAF, LWORK, INFO
DOUBLE PRECISION   D(*)
COMPLEX*16         E(*), AF(*), WORK(*)
```

The ENTRY statement enables the routine to be called by its ScaLAPACK name.

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.
M_b^X	–	the blocking factor for the distribution of the rows of a matrix X.
N_b^X	–	the blocking factor for the distribution of the columns of a matrix X.

3.2 Global and Local Arguments

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

Global input arguments: N, JA, some elements of IDESCA (see Section 4 for details of IDESCA)

Global output arguments: INFO

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. **It is important that** $p \times N_b^A \geq \mathrm{mod}(j_A - 1,\ N_b^A) + n$. This restriction states that the mapping for matrices must be blocked, reflecting the nature of the **divide and conquer algorithm** as a task-parallel algorithm (see Section 6.1). This means that no processor may store more than one block of the matrix.

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 F07JRFP (PZPTTRF):

Real and complex matrix generation: column block distribution: F01ZZFP and F01ZYFP for the distribution of respectively the vectors d and e.

4 Arguments

1: N — INTEGER *Global Input*

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

Constraint: N ≥ 0.

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

Note: the dimension of array D must be at least N_b^A.

On entry: the local part of the distributed vector d which contains the diagonal of the matrix A. See the F07 Chapter Introduction for further detail.

On exit: information containing the factors of the matrix A_s. See Section 6.1.

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

Note: the dimension of array E must be at least N_b^A.

On entry: the local part of the distributed vector e which contains the off-diagonal of the matrix A. E should be aligned with D and E(N) may be set to 0. E can store either the upper off-diagonal or the lower off-diagonal of the matrix A. In the first case, the factorization $A = PU^HUP^T$ is performed and details of U are returned in E; in the second case, the factorization $A = PLL^HP^T$ is performed and details of L are returned in E. See the F07 Chapter Introduction for further detail.

On exit: information containing the factors of the matrix A_s. See Section 6.1.

4: JA — INTEGER *Global Input*

On entry: j_A, the column index of matrix A that identifies the first column of the submatrix A_s to be factorized.

Constraints: $1 \leq$ JA $\leq n_A -$ N+1.

5: IDESCA($*$) — INTEGER array *Local Input*

Note: the dimension of the array IDESCA must be at least 5 when IDESCA(1) = 501 or 502 and must be at least 9 when IDESCA(1) = 1.

Distribution: if IDESCA(1) = 1, the array elements IDESCA(3:8) must be global to the processor grid. If IDESCA(1) = 501 or 502, then only the array elements IDESCA(3:5) must be global. In either case IDESCA(2) is local to each processor.

On entry: the description array for the matrix A. This array must contain details of the distribution of the matrix A (characterized by d and e) and the logical processor grid.

IDESCA(1), the descriptor type.

If IDESCA(1) = 1, then $p = 1 \times n_p$ and:

IDESCA(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;

IDESCA(3), the number of rows, m_A, of the matrix A;

IDESCA(4), the number of columns, n_A, of the matrix A;

IDESCA(5), the blocking factor, M_b^A, used to distribute the rows of the matrix A (in that case, IDESCA(5) = 1);

IDESCA(6), the blocking factor, N_b^A, used to distribute the columns of the matrix A;

IDESCA(7), the processor row index over which the first row of the matrix A is distributed (since the logical grid is one-dimensional, IDESCA(7) = 0);
IDESCA(8), the processor column index over which the first column of the matrix A is distributed;
IDESCA(9), the leading dimension of the conceptual two-dimensional array A (in this case, IDESCA(9) is not referenced).

If IDESCA(1) = 501 or 502, then $p = 1 \times n_p$ or $p = m_p \times 1$, and:
IDESCA(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;
IDESCA(3), the size n_A, of the matrix A;
IDESCA(4), the blocking factor, N_b^A, used to distribute the matrix A;
IDESCA(5), the processor column index over which the first element of the matrix A is distributed;
IDESCA(6:9) are not referenced.

Suggested value: IDESCA(1) = 501 and $p = 1 \times n_p$.

Constraints:

IDESCA(1) = 1, 501 or 502;
if IDESCA(1) = 1, then $p = 1 \times n_p$;
if IDESCA(1) = 501 or 502; then $p = m_p \times 1$ or $p = 1 \times n_p$;
if IDESCA(1) = 501 or 502, then
$1 \leq$ IDESCA(3) $\leq$ N + JA−1;
IDESCA(4) ≥ 2 and $p\times$IDESCA(4) $\geq$ mod(JA−1,IDESCA(4)) + N;
IDESCA(5) ≥ 0;
if IDESCA(1) = 1, then
$1 \leq$ IDESCA(4) $\leq$ N + JA−1;
IDESCA(6) ≥ 2 and $p\times$IDESCA(6) $\geq$ mod(JA−1,IDESCA(4)) + N;
IDESCA(8) ≥ 0.

6: AF(∗) — COMPLEX*16 array *Local Output*

On exit: the auxiliary fill-in space. Fill-in is created and stored during the factorization. If LAF is not large enough, after an unsuccesful exit, INT(AF(1)) will contain the minimum acceptable size of AF.

7: LAF — INTEGER *Local Input*

On entry: the dimension of the array AF .

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

8: WORK(∗) — COMPLEX*16 array *Local Workspace*

Note: the dimension of WORK must be at least max(1,LWORK). The minimum value of LWORK required to successfully call this routine can be obtained by setting LWORK = −1. The required size is returned in the real part of array element WORK(1).

On exit: the real part of WORK(1) contains the minimum dimension of the array WORK required to successfully complete the task.

9: LWORK — INTEGER *Local Input*

On entry: either −1 (see WORK) or the dimension of the array WORK required to successfully complete the task. If LWORK is set to −1 on entry this routine simply performs some initial error checking and then, if these checks are successful, calculates the minimum size of LWORK required.

Constraints:

either LWORK = −1,
or LWORK = 8×NPROW×NPCOL, where NPROW = m_p and NPCOL = n_p.

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

5 Errors and Warnings

If INFO $\neq 0$ 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).

INFO $= -i$

On entry, one of the arguments was invalid:

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

This error occurred 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.

INFO > 0

If INFO $= k \leq p$ the submatrix stored and factorized locally on processor $\{0,k\}$ or $\{k,0\}$ was not positive definite, and the factorization was not completed. If INFO $= k \geq p$ 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 $16n$. A call to this routine may be followed by a call to the routine F07JSFP (PZPTTRS) to solve $A_sX = B_s$.

Because of permutation performed by F07HDFP (PZPTTRF) to achieve better parallelism, the factors produced (which are stored in the arrays D, E and AF) will be **different** to those produced by equivalent sequential codes. However, they can be used in a subsequent call to the corresponding solver routine, F07JSFP (PZPTTRS).

Users should not access the factorized matrix directly, and the arrays D, E and AF **must not** be altered between calls to the factorize and the solver routines.

6.1 Parallelism Detail

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

(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^A should not be to small. Otherwise the divide and conquer algorithm performs poorly.

6.2 Accuracy

The computed factors d and e are the exact factor of a perturbed matrix $A + \Sigma$, where

$$|\Sigma| \leq C\epsilon|E^H| \cdot |E|,$$

C is a constant, ϵ is the ***machine precision***, and E is the matrix, with zero everywhere, except the diagonal set to 1 and the off diagonal set to the vector e.

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.

8.1 Example Text

```
*     F07JRFP 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          IAROW, LAF, NRHSMAX, LW
      PARAMETER        (IAROW=0,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, IB, ICNTXT, IFAIL, INFO, JA, LEVEL, LWORK, MP,
     +                 MYCOL, MYROW, NB, NP, NPCOL, NPROW, NRHS
      LOGICAL          ROOT
      CHARACTER        UPLO
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      COMPLEX*16       AF(LAF), B(LDB,NRHSMAX), E(TDA),
     +                 SOL(LDB,NRHSMAX), WORK(LW)
      DOUBLE PRECISION D(TDA)
      INTEGER          IDESCA(9), IDESCB(9)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         DGERV2D, DGESD2D, EXACT, F01YZFP, F01ZYFP,
     +                 F01ZZFP, F07JRFP, F07JSFP, GD, GE, GRHSB,
     +                 X04BZFP, Z01AAFP, Z01ABFP, Z01ZAFP, Z02EAFP
*     .. Intrinsic Functions ..
      INTRINSIC        DCONJG, DBLE, DSQRT
*     .. Common blocks ..
      COMMON           /DIM/N
*     .. Executable Statements ..
*
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F07JRFP Example Program Results'

      MP = MG
      NP = NG
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      CALL Z01ZAFP(ICNTXT,NPROW,NPCOL,MYROW,MYCOL)
*
*     Set error checking level
*
      LEVEL = 1
      CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
*     Initialization of Data
*
      N = 12
      NB = N/(MG*NG)
      LWORK = LW
      NRHS = 1
      FORMAT = '(2F12.4)'
      UPLO = 'L'

      IF (N.LE.NMAX .AND. NB.LE.NBMAX) THEN
*
*       Array Descriptor for  A
*
         IDESCA(1) = 501
```

```
            IDESCA(2) = ICNTXT
            IDESCA(3) = N
            IDESCA(4) = NB
            IDESCA(5) = IAROW
            JA = 1
*
*           Array Descriptor for B
*
            IDESCB(1) = 502
            IDESCB(2) = ICNTXT
            IDESCB(3) = N
            IDESCB(4) = NB
            IDESCB(5) = IAROW
            IDESCB(6) = LDB
            IB = 1
*
*           Generation of the tridiagonal matrix A
*
            IFAIL = 0
            CALL F01ZZFP(GD,N,D,IDESCA,IFAIL)
*
            IFAIL = 0
            CALL F01ZYFP(GE,N,E,IDESCA,IFAIL)
*
*           Generation of RHS
*
            IFAIL = 0
            CALL F01YZFP(GRHSB,N,NRHS,B,IDESCB,IFAIL)
*
*           Generation of the exact solution
*
            IFAIL = 0
            CALL F01YZFP(EXACT,N,NRHS,SOL,IDESCB,IFAIL)
*
*            Factorize A
*
            CALL F07JRFP(N,D,E,JA,IDESCA,AF,LAF,WORK,LWORK,INFO)
*
            IF (INFO.EQ.0) THEN
*
*               Solve AX = B
*
               CALL F07JSFP(UPLO,N,NRHS,D,E,JA,IDESCA,B,IB,IDESCB,AF,LAF,
     +                      WORK,LWORK,INFO)
*
*     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,*) 'Solutions'
               WRITE (NOUT,*)
               WRITE (NOUT,*) ' Computed '
               WRITE (NOUT,*)
            END IF
*
            IFAIL = 0
*
            CALL X04BZFP(NOUT,N,NRHS,B,IDESCB,FORMAT,WORK,IFAIL)
*
            IF (ROOT) THEN
               WRITE (NOUT,*)
               WRITE (NOUT,*) ' Exact '
               WRITE (NOUT,*)
            END IF
*
            IFAIL = 0
            CALL X04BZFP(NOUT,N,NRHS,SOL,IDESCB,FORMAT,WORK,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
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
*
      SUBROUTINE GRHSB(J1,JL,NRHS,BL,LDBL)
*     .. 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.D0,2.D0)
         K = K + 1
   20 CONTINUE
*
*     End of GRHSB
*
      RETURN
      END
*
      SUBROUTINE EXACT(J1,JL,NRHS,BL,LDBL)
*     .. 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.D0/DBLE(N+1)
      DO 20 J = J1, JL
         BL(K,1) = DSIN(DBLE(J)*PI*H)*(1.D0,2.D0)
         K = K + 1
   20 CONTINUE
*
*     End of EXACT
*
      RETURN
      END
*
```

```
      SUBROUTINE GD(J1,JL,BL)
*     .. Scalar Arguments ..
      INTEGER       J1, JL
*     .. Array Arguments ..
      DOUBLE PRECISION BL(*)
*     .. Local Scalars ..
      INTEGER       J, K
*     .. Executable Statements ..
      K = 1
      DO 20 J = J1, JL
         BL(K) = 2.D0
         K = K + 1
   20 CONTINUE
*
*  End of GD
*
      RETURN
      END
*
      SUBROUTINE GE(J1,JL,BL)
*     .. Scalar Arguments ..
      INTEGER       J1, JL
*     .. Array Arguments ..
      COMPLEX*16    BL(*)
*     .. Local Scalars ..
      INTEGER       J, K
*     .. Executable Statements ..
      K = 1
      DO 20 J = J1, JL
         BL(K) = -1.D0
         K = K + 1
   20 CONTINUE
*
*  End of GE
*
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
F07JRFP Example Program Results

Solutions

 Computed

     0.2405      0.4810
     0.4670      0.9340
     0.6664      1.3327
     0.8270      1.6540
     0.9396      1.8792
     0.9976      1.9951
     0.9976      1.9951
```

```
    0.9396      1.8792
    0.8270      1.6540
    0.6664      1.3327
    0.4670      0.9340
    0.2405      0.4810

Exact

    0.2393      0.4786
    0.4647      0.9294
    0.6631      1.3262
    0.8230      1.6460
    0.9350      1.8700
    0.9927      1.9854
    0.9927      1.9854
    0.9350      1.8700
    0.8230      1.6460
    0.6631      1.3262
    0.4647      0.9294
    0.2393      0.4786

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