```
      SUBROUTINE DGETRI( N, A, LDA, IPIV, WORK, LWORK, INFO )
*
*  -- LAPACK routine (version 3.2) --
*  -- LAPACK is a software package provided by Univ. of Tennessee,    --
*  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG Ltd..--
*     November 2006
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, LWORK, N
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * ), WORK( * )
*     ..
*
*  Purpose
*  =======
*
*  DGETRI computes the inverse of a matrix using the LU factorization
*  computed by DGETRF.
*
*  This method inverts U and then computes inv(A) by solving the system
*  inv(A)*L = inv(U) for inv(A).
*
*  Arguments
*  =========
*
*  N       (input) INTEGER
*          The order of the matrix A.  N >= 0.
*
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
*          On entry, the factors L and U from the factorization
*          A = P*L*U as computed by DGETRF.
*          On exit, if INFO = 0, the inverse of the original matrix A.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,N).
*
*  IPIV    (input) INTEGER array, dimension (N)
*          The pivot indices from DGETRF; for 1<=i<=N, row i of the
*          matrix was interchanged with row IPIV(i).
*
*  WORK    (workspace/output) DOUBLE PRECISION array, dimension (MAX(1,LWORK))
*          On exit, if INFO=0, then WORK(1) returns the optimal LWORK.
*
*  LWORK   (input) INTEGER
*          The dimension of the array WORK.  LWORK >= max(1,N).
*          For optimal performance LWORK >= N*NB, where NB is
*          the optimal blocksize returned by ILAENV.
*
*          If LWORK = -1, then a workspace query is assumed; the routine
*          only calculates the optimal size of the WORK array, returns
*          this value as the first entry of the WORK array, and no error
*          message related to LWORK is issued by XERBLA.
*
*  INFO    (output) INTEGER
*          = 0:  successful exit
*          < 0:  if INFO = -i, the i-th argument had an illegal value
*          > 0:  if INFO = i, U(i,i) is exactly zero; the matrix is
*                singular and its inverse could not be computed.
```

```
*
*  =====================================================================
*
*     .. Parameters ..
      DOUBLE PRECISION   ZERO, ONE
      PARAMETER          ( ZERO = 0.0D+0, ONE = 1.0D+0 )
*     ..
*     .. Local Scalars ..
      LOGICAL            LQUERY
      INTEGER            I, IWS, J, JB, JJ, JP, LDWORK, LWKOPT, NB,
     $                   NBMIN, NN
*     ..
*     .. External Functions ..
      INTEGER            ILAENV
      EXTERNAL           ILAENV
*     ..
*     .. External Subroutines ..
      EXTERNAL           DGEMM, DGEMV, DSWAP, DTRSM, DTRTRI, XERBLA
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
*     ..
*     .. Executable Statements ..
*
*     Test the input parameters.
*
      INFO = 0
      NB = ILAENV( 1, 'DGETRI', ' ', N, -1, -1, -1 )
      LWKOPT = N*NB
      WORK( 1 ) = LWKOPT
      LQUERY = ( LWORK.EQ.-1 )
      IF( N.LT.0 ) THEN
         INFO = -1
      ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
         INFO = -3
      ELSE IF( LWORK.LT.MAX( 1, N ) .AND. .NOT.LQUERY ) THEN
         INFO = -6
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETRI', -INFO )
         RETURN
      ELSE IF( LQUERY ) THEN
         RETURN
      END IF
*
*     Quick return if possible
*
      IF( N.EQ.0 )
     $   RETURN
*
*     Form inv(U).  If INFO > 0 from DTRTRI, then U is singular,
*     and the inverse is not computed.
*
      CALL DTRTRI( 'Upper', 'Non-unit', N, A, LDA, INFO )
      IF( INFO.GT.0 )
     $   RETURN
*
      NBMIN = 2
      LDWORK = N
      IF( NB.GT.1 .AND. NB.LT.N ) THEN
         IWS = MAX( LDWORK*NB, 1 )
```

```
         IF( LWORK.LT.IWS ) THEN
            NB = LWORK / LDWORK
            NBMIN = MAX( 2, ILAENV( 2, 'DGETRI', ' ', N, -1, -1, -1 ) )
         END IF
      ELSE
         IWS = N
      END IF
*
*     Solve the equation inv(A)*L = inv(U) for inv(A).
*
      IF( NB.LT.NBMIN .OR. NB.GE.N ) THEN
*
*        Use unblocked code.
*
         DO 20 J = N, 1, -1
*
*           Copy current column of L to WORK and replace with zeros.
*
            DO 10 I = J + 1, N
               WORK( I ) = A( I, J )
               A( I, J ) = ZERO
   10       CONTINUE
*
*           Compute current column of inv(A).
*
            IF( J.LT.N )
     $         CALL DGEMV( 'No transpose', N, N-J, -ONE, A( 1, J+1 ),
     $                     LDA, WORK( J+1 ), 1, ONE, A( 1, J ), 1 )
   20    CONTINUE
      ELSE
*
*        Use blocked code.
*
         NN = ( ( N-1 ) / NB )*NB + 1
         DO 50 J = NN, 1, -NB
            JB = MIN( NB, N-J+1 )
*
*           Copy current block column of L to WORK and replace with
*           zeros.
*
            DO 40 JJ = J, J + JB - 1
               DO 30 I = JJ + 1, N
                  WORK( I+( JJ-J )*LDWORK ) = A( I, JJ )
                  A( I, JJ ) = ZERO
   30          CONTINUE
   40       CONTINUE
*
*           Compute current block column of inv(A).
*
            IF( J+JB.LE.N )
     $         CALL DGEMM( 'No transpose', 'No transpose', N, JB,
     $                     N-J-JB+1, -ONE, A( 1, J+JB ), LDA,
     $                     WORK( J+JB ), LDWORK, ONE, A( 1, J ), LDA )
            CALL DTRSM( 'Right', 'Lower', 'No transpose', 'Unit', N, JB,
     $                  ONE, WORK( J ), LDWORK, A( 1, J ), LDA )
   50    CONTINUE
      END IF
*
*     Apply column interchanges.
*
      DO 60 J = N - 1, 1, -1
```

```
         JP = IPIV( J )
         IF( JP.NE.J )
     $      CALL DSWAP( N, A( 1, J ), 1, A( 1, JP ), 1 )
   60 CONTINUE
*
      WORK( 1 ) = IWS
      RETURN
*
*     End of DGETRI
*
      END
```