```fortran
      SUBROUTINE ZM_LIN_SOLVE(A,X,B,N,DET)
      USE FMVALS
      USE FMZM
      IMPLICIT NONE

!  Gauss elimination to solve the linear system  A X = B, where:

!  A   is the matrix of the system, containing the  N x N coefficient matrix.

!  B   is the  N x 1  right-hand-side vector.

!  X   is the returned  N x 1  solution vector.

!  DET is returned as the determinant of A.
!      Nonzero DET means a solution was found.
!      DET = 0 is returned if the system is singular.

!  A,X,B,DET are all type (zm) complex multiprecision variables.

      INTEGER :: N
      TYPE (ZM) :: A(N,N), B(N), X(N), DET
      TYPE (FM), SAVE :: TOL
      TYPE (ZM), ALLOCATABLE :: A1(:,:), A2(:,:), B1(:), R1(:), X1(:)
      INTEGER, ALLOCATABLE :: KSWAP(:)
      INTEGER :: I, J, NDSAVE

      CALL FM_ENTER_USER_ROUTINE
      ALLOCATE(A1(N,N),A2(N,N),B1(N),R1(N),X1(N),KSWAP(N),STAT=J)
      IF (J /= 0) THEN
          WRITE (*,"(/' Error in ZM_LIN_SOLVE.  Unable to allocate arrays with N = ',I8/)") N
          STOP
      ENDIF

      TOL = EPSILON(TO_FM(1))/MBASE/TO_FM(10)**10

      NDSAVE = NDIG
      NDIG = 2*NDIG

!             Copy A and B to A1 and B1 with higher precision.

  110 CALL FM_EQU_R1(TOL,NDSAVE,NDIG)
      DO I = 1, N
         DO J = 1, N
            CALL ZM_EQU(A(I,J),A1(I,J),NDSAVE,NDIG)
            CALL ZM_EQ(A1(I,J),A2(I,J))
         ENDDO
         CALL ZM_EQU(B(I),B1(I),NDSAVE,NDIG)
      ENDDO

!             Solve the system.

      CALL ZM_FACTOR_LU(A1,N,DET,KSWAP)
      IF (DET == 0 .OR. IS_UNKNOWN(DET)) THEN
          IF (KWARN > 0) THEN
              WRITE (KW,"(/' Error in ZM_LIN_SOLVE.  The matrix is singular.'/)")
```

```
            ENDIF
            IF (KWARN >= 2) STOP
            X1 = TO_ZM(' UNKNOWN ')
            GO TO 120
        ENDIF
        CALL ZM_SOLVE_LU(A1,N,B1,X1,KSWAP)

!              Do an iterative refinement.

        R1 = MATMUL(A2,X1) - B1

        CALL ZM_SOLVE_LU(A1,N,R1,B1,KSWAP)
        X1 = X1 - B1

!              Check for accuracy at the user's precision.

        IF (SQRT( ABS(DOT_PRODUCT( B1 , B1 )) ) > TOL) THEN
            NDIG = 2*NDIG
            GO TO 110
        ENDIF

!              Round and return X and DET.

    120 DO I = 1, N
           CALL ZM_EQU(X1(I),X(I),NDIG,NDSAVE)
        ENDDO
        CALL ZMEQU_R1(DET%MZM,NDIG,NDSAVE)

        NDIG = NDSAVE

        CALL FM_DEALLOCATE(A1)
        CALL FM_DEALLOCATE(A2)
        CALL FM_DEALLOCATE(B1)
        CALL FM_DEALLOCATE(R1)
        CALL FM_DEALLOCATE(X1)
        DEALLOCATE(A1,A2,B1,R1,X1,KSWAP)

        CALL FM_EXIT_USER_ROUTINE
        END SUBROUTINE ZM_LIN_SOLVE

        SUBROUTINE ZM_FACTOR_LU(A,N,DET,KSWAP)
        USE FMZM
        IMPLICIT NONE

!  Gauss elimination to factor the NxN matrix A (LU decomposition).

!  The time is proportional to  N**3.

!  Once this factorization has been done, a linear system  A x = b
!  with the same coefficient matrix A and Nx1 vector b can be solved
!  for x using routine ZM_SOLVE_LU in time proportional to  N**2.

!  DET is returned as the determinant of A.
!      Nonzero DET means there is a unique solution.
!      DET = 0 is returned if the system is singular.
```

```fortran
!  KSWAP is a list of row interchanges made by the partial pivoting strategy during the
!        elimination phase.

!  After returning, the values in matrix A have been replaced by the multipliers
!  used during elimination.  This is equivalent to factoring the A matrix into
!  a lower triangular matrix L times an upper triangular matrix U.

      INTEGER :: N
      INTEGER :: JCOL, JDIAG, JMAX, JROW, KSWAP(N)
      TYPE (ZM) :: A(N,N), DET
      TYPE (ZM), SAVE :: AMAX, AMULT, TEMP

      CALL FM_ENTER_USER_ROUTINE
      DET = 1
      KSWAP(1:N) = 1
      IF (N <= 0) THEN
          DET = 0
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF
      IF (N == 1) THEN
          KSWAP(1) = 1
          DET = A(1,1)
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF

!             Do the elimination phase.
!             JDIAG is the current diagonal element below which the elimination proceeds.

      DO JDIAG = 1, N-1

!             Pivot to put the element with the largest absolute value on the diagonal.

         AMAX = ABS(A(JDIAG,JDIAG))
         JMAX = JDIAG
         DO JROW = JDIAG+1, N
            IF (ABS(A(JROW,JDIAG)) > ABS(AMAX)) THEN
                AMAX = ABS(A(JROW,JDIAG))
                JMAX = JROW
            ENDIF
         ENDDO

!             If AMAX is zero here then the system is singular.

         IF (AMAX == 0.0) THEN
             DET = 0
             CALL FM_EXIT_USER_ROUTINE
             RETURN
         ENDIF

!             Swap rows JDIAG and JMAX unless they are the same row.

         KSWAP(JDIAG) = JMAX
         IF (JMAX /= JDIAG) THEN
             DET = -DET
```

```fortran
              DO JCOL = JDIAG, N
                 TEMP = A(JDIAG,JCOL)
                 A(JDIAG,JCOL) = A(JMAX,JCOL)
                 A(JMAX,JCOL) = TEMP
              ENDDO
          ENDIF
          DET = DET * A(JDIAG,JDIAG)

!              For JROW = JDIAG+1, ..., N, eliminate A(JROW,JDIAG) by replacing row JROW by
!                  row JROW - A(JROW,JDIAG) * row JDIAG / A(JDIAG,JDIAG)

          DO JROW = JDIAG+1, N
             IF (A(JROW,JDIAG) == 0) CYCLE
             AMULT = A(JROW,JDIAG)/A(JDIAG,JDIAG)

!              Save the multiplier for use later by ZM_SOLVE_LU.

             A(JROW,JDIAG) = AMULT
             DO JCOL = JDIAG+1, N
                A(JROW,JCOL) = A(JROW,JCOL) - AMULT*A(JDIAG,JCOL)
             ENDDO
          ENDDO
      ENDDO
      DET = DET * A(N,N)

      CALL FM_EXIT_USER_ROUTINE
      END SUBROUTINE ZM_FACTOR_LU

      SUBROUTINE ZM_SOLVE_LU(A,N,B,X,KSWAP)
      USE FMZM
      IMPLICIT NONE

!  Solve a linear system  A x = b.
!  A is the NxN coefficient matrix, after having been factored by ZM_FACTOR_LU.
!  B is the Nx1 right-hand-side vector.
!  X is returned with the solution of the linear system.
!  KSWAP is a list of row interchanges made by the partial pivoting strategy during the
!        elimination phase in ZM_FACTOR_LU.
!  Time for this call is proportional to  N**2.

      INTEGER :: N, KSWAP(N)
      TYPE (ZM) :: A(N,N), B(N), X(N)
      INTEGER :: J, JDIAG, JMAX
      TYPE (ZM), SAVE :: TEMP

      CALL FM_ENTER_USER_ROUTINE
      IF (N <= 0) THEN
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF
      IF (N == 1) THEN
          X(1) = B(1) / A(1,1)
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF
      DO J = 1, N
```

```
         X(J) = B(J)
      ENDDO

!             Do the elimination phase operations only on X.
!             JDIAG is the current diagonal element below which the elimination proceeds.

      DO JDIAG = 1, N-1

!             Pivot to put the element with the largest absolute value on the diagonal.

         JMAX = KSWAP(JDIAG)

!             Swap rows JDIAG and JMAX unless they are the same row.

         IF (JMAX /= JDIAG) THEN
             TEMP = X(JDIAG)
             X(JDIAG) = X(JMAX)
             X(JMAX) = TEMP
         ENDIF

!             For JROW = JDIAG+1, ..., N, eliminate A(JROW,JDIAG) by replacing row JROW by
!                 row JROW - A(JROW,JDIAG) * row JDIAG / A(JDIAG,JDIAG)
!             After factoring, A(JROW,JDIAG) is the original A(JROW,JDIAG) / A(JDIAG,JDIAG).

         DO J = JDIAG+1, N
            X(J) = X(J) - A(J,JDIAG) * X(JDIAG)
         ENDDO
      ENDDO

!             Do the back substitution.

      DO JDIAG = N, 1, -1

!             Divide row JDIAG by the diagonal element.

         X(JDIAG) = X(JDIAG) / A(JDIAG,JDIAG)

!             Zero above the diagonal in column JDIAG by replacing row JROW by
!                 row JROW - A(JROW,JDIAG) * row JDIAG
!             For JROW = 1, ..., JDIAG-1.

         IF (JDIAG == 1) EXIT
         DO J = 1, JDIAG-1
            X(J) = X(J) - A(J,JDIAG) * X(JDIAG)
         ENDDO
      ENDDO

      CALL FM_EXIT_USER_ROUTINE
      END SUBROUTINE ZM_SOLVE_LU
```