```
      SUBROUTINE ZM_SECANT(AX,BX,F,NF,ROOT,KPRT,KU)
      USE FMVALS
      USE FMZM
      IMPLICIT NONE

!  This routine searches for a root of F(X,NF) = 0 using AX and BX as starting points.
!  AX and BX are complex, and the search can fail if AX and BX are not close enough to any roots
!  or if the function is badly behaved.

!  When a root is found, ZM_SECANT tries to return full accuracy even in the case of multiple
!  or closely-spaced roots, by raising precision above the user's level.

!  ROOT  is the value returned as the approximate root of the equation.

!  KPRT  controls printing within the routine:
!        KPRT = -1 for no output
!        KPRT =  0 for no output except warning and error messages.
!        KPRT =  1 for the approximation to the root and the function
!                  value to be printed once at the end of the routine.
!        KPRT =  2 for the approximation to the root and the function
!                  value to be printed each iteration.

!  KU    is the unit number for output.

      TYPE (ZM)           :: AX, BX, ROOT
      TYPE (ZM), EXTERNAL :: F, ZM_FPRIME
      CHARACTER (80) :: STR
      DOUBLE PRECISION :: VALUE
      INTEGER :: J, JSET, KPRT, KU, KWARN_SAVE, MAXIT, NDSAVE, NF
      LOGICAL :: USE_F_OVER_FP
      TYPE (ZM), SAVE :: F1, F1OLD, F2, X1, X1OLD, X2, X3
      TYPE (FM), SAVE :: ERR, ERR1, TOL

!             Subroutines may create temporary multiple precision variables to hold the argument
!             values in cases where an argument might be A+B or TO_ZM('1 + 7 i').
!             To avoid deleting these temporaries before we are finished using them, any subroutine
!             that has multiple precision arguments must call FM_ENTER_USER_ROUTINE upon entry and
!             FM_EXIT_USER_ROUTINE when returning.  Then the FM memory manager will know when it is
!             safe to delete these temporary variables.

      CALL FM_ENTER_USER_ROUTINE
      IF (KPRT == 2) THEN
          WRITE (KU,*) ' '
          WRITE (KU,*) ' ZM_SECANT.  Begin trace of all iterations.'
      ENDIF

!             Raise precision slightly.

      NDSAVE = NDIG
      NDIG = NDIG + NGRD52
      CALL ZM_EQU(AX,X1,NDSAVE,NDIG)
      CALL ZM_EQU(BX,X2,NDSAVE,NDIG)
      KWARN_SAVE = KWARN
      KWARN = 0
```

```
      MAXIT = 1000
      JSET = 50
      ERR = 1
      TOL = 100*EPSILON(ERR)
      USE_F_OVER_FP = .FALSE.
      F1 = F(X1,NF)
      F2 = F(X2,NF)

!             Check for legal function values.

      IF (IS_UNKNOWN(ABS(F1)) .OR. IS_OVERFLOW(ABS(F1))) THEN
          DO J = 1, 3
             X3 = ((TO_FM(4) - J)/4)*X1 + (1-(TO_FM(4) - J)/4)*X2
             F1 = F(X3,NF)
             IF (.NOT. (IS_UNKNOWN(ABS(F1)) .OR. IS_OVERFLOW(ABS(F1)))) THEN
                 X1 = X3
                 EXIT
             ENDIF
          ENDDO
          IF (IS_UNKNOWN(ABS(F1)) .OR. IS_OVERFLOW(ABS(F1))) THEN
              DO J = 1, 3
                 X3 = ((TO_FM(4) + J)/4)*X1 + (1-(TO_FM(4) + J)/4)*X2
                 F1 = F(X3,NF)
                 IF (.NOT. (IS_UNKNOWN(ABS(F1)) .OR. IS_OVERFLOW(ABS(F1)))) THEN
                     X1 = X3
                     EXIT
                 ENDIF
                 X3 = (1-(TO_FM(4) + J)/4)*X1 + ((TO_FM(4) + J)/4)*X2
                 F1 = F(X3,NF)
                 IF (.NOT. (IS_UNKNOWN(ABS(F1)) .OR. IS_OVERFLOW(ABS(F1)))) THEN
                     X1 = X3
                     EXIT
                 ENDIF
              ENDDO
          ENDIF
      ENDIF
      IF (IS_UNKNOWN(ABS(F1)) .OR. IS_OVERFLOW(ABS(F1))) THEN
          IF (KPRT >= 0) THEN
              WRITE (KU,*) ' '
              WRITE (KU,*) ' Invalid input for ZM_SECANT. ',  &
                           ' Unknown or overflowed function value for AX ='
              CALL ZM_PRINT(X1)
              WRITE (KU,*) ' '
          ENDIF
          J = 0
          X2 = TO_ZM(' UNKNOWN + UNKNOWN i ')
          ERR = TO_ZM(' UNKNOWN + UNKNOWN i ')
          GO TO 110
      ENDIF

      IF (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2))) THEN
          DO J = 1, 3
             X3 = ((TO_FM(4) - J)/4)*X1 + (1-(TO_FM(4) - J)/4)*X2
             F2 = F(X3,NF)
             IF (.NOT. (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2)))) THEN
                 X2 = X3
```

```
                EXIT
            ENDIF
        ENDDO
        IF (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2))) THEN
            DO J = 1, 3
               X3 = ((TO_FM(4) + J)/4)*X1 + (1-(TO_FM(4) + J)/4)*X2
               F2 = F(X3,NF)
               IF (.NOT. (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2)))) THEN
                   X2 = X3
                   EXIT
               ENDIF
               X3 = (1-(TO_FM(4) + J)/4)*X1 + ((TO_FM(4) + J)/4)*X2
               F2 = F(X3,NF)
               IF (.NOT. (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2)))) THEN
                   X2 = X3
                   EXIT
               ENDIF
            ENDDO
        ENDIF
    ENDIF
    IF (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2))) THEN
        IF (KPRT >= 0) THEN
            WRITE (KU,*) ' '
            WRITE (KU,*) ' Invalid input for ZM_SECANT. ',  &
                         ' Unknown or overflowed function value for BX ='
            CALL ZM_PRINT(X2)
            WRITE (KU,*) ' '
        ENDIF
        J = 0
        X2 = TO_ZM(' UNKNOWN + UNKNOWN i ')
        ERR = TO_ZM(' UNKNOWN + UNKNOWN i ')
        GO TO 110
    ENDIF

!           Secant does not do well if the magnitude of the two starting function values differ
!           by too much.  Adjust if necessary.

    DO J = 1, 10
       IF (ABS(F2/F1) > 10) THEN
           X2 = (X1 + X2)/2
           F2 = F(X2,NF)
       ELSE IF (ABS(F1/F2) > 10) THEN
           X1 = (X1 + X2)/2
           F1 = F(X1,NF)
       ELSE
           EXIT
       ENDIF
    ENDDO

    IF (KPRT == 2) THEN
        STR = ZM_FORMAT('ES20.10','ES20.10',F1)
        WRITE (KU,"('  J =',I3,3X,'f(AX) = ',A,'    x:')") 0,TRIM(STR)
        CALL ZM_PRINT(X1)
        STR = ZM_FORMAT('ES20.10','ES20.10',F2)
        WRITE (KU,"('  J =',I3,3X,'f(BX) = ',A,'    x:')") 0,TRIM(STR)
        CALL ZM_PRINT(X2)
```

```
      ENDIF

!             This loop does the iteration.

      DO J = 1, MAXIT

         IF (F2-F1 /= 0.0) THEN
             X3 = X2 - F2*(X2-X1)/(F2-F1)
         ELSE
             X3 = X2 + 1
         ENDIF

!             Multiple roots cause very slow convergence and loss of accuracy.
!             If the slope is very small, try to improve convergence and accuracy by using
!             the (slower) function  f(x)/f'(x)  which has no multiple roots.

         X1OLD = X1
         F1OLD = F1
         IF ( (ABS((F2-F1)/(X2-X1)) < 1.0D-2 .AND. ABS(F2) < 1.0D-4 .AND.  &
               ABS(F2*(X2-X1)/(F2-F1)) < 1.0D-4*ABS(X2)) .OR. USE_F_OVER_FP) THEN
             USE_F_OVER_FP = .TRUE.
             X1 = X2
             X2 = X3
             F1 = F2
             F2 = ZM_FPRIME(0,X3,F,NF) / ZM_FPRIME(1,X3,F,NF)
         ELSE
             X1 = X2
             X2 = X3
             F1 = F2
             F2 = F(X3,NF)
         ENDIF

!             If F2 is one of the FM non-numbers, +-underflow, +-overflow, unknown,
!             then replace it by something representable, so that the next x3 will be
!             closer to x1.  Also swap x1 and x2, making the bad x go away first.

         IF (IS_UNKNOWN(ABS(F2)) .OR. IS_OVERFLOW(ABS(F2))) THEN
             F2 = -2*F1
             X3 = X1
             X1 = X2
             X2 = X3
             X3 = F1
             F1 = F2
             F2 = X3
         ENDIF

!             A common failure mode for secant is to get into a pattern that repeats x1 and x2
!             close together with nearly equal function values and x3 farther away with much
!             larger function value.  Check for this, and re-start the iteration by choosing
!             a different x3.

         IF (ABS(F2) > 100*MAX(ABS(F1OLD),ABS(F1)) .AND. J >= JSET) THEN
             IF (JSET >= 200) THEN
                 MAXIT = J
                 EXIT
             ENDIF
```

```
            JSET = JSET + 50
            CALL FM_RANDOM_NUMBER(VALUE)
            VALUE = 9*VALUE - 4
            X2 = VALUE*X1 + (1-VALUE)*X2
            F2 = F(X2,NF)
        ENDIF

        IF (KPRT == 2) THEN
            STR = ZM_FORMAT('ES20.10','ES20.10',F2)
            WRITE (KU,"('  J =',I3,4X,'f(x) = ' ,A,'    x:')") J,TRIM(STR)
            CALL ZM_PRINT(X2)
        ENDIF

        ERR1 = ERR
        IF (X2 /= 0.0) THEN
            ERR = ABS((X2-X1)/X2)
        ELSE
            ERR = ABS(X2-X1)
        ENDIF

!            If the error is less than the tolerance, double check to make sure the previous
!            error was small along with the current function value.  Some divergent iterations
!            can get err < tol without being close to a root.

        IF (ERR < TOL .OR. F2 == 0) THEN
            IF (ERR1 > SQRT(SQRT(TOL)) .AND. ABS(F2) > SQRT(EPSILON(TO_FM(1)))) THEN
                IF (KPRT >= 0) THEN
                    WRITE (KU,"(/' Possible false convergence in ZM_SECANT after',I5,"//  &
                              "' iterations.  ','Last two approximations =')") J
                    CALL ZM_PRINT(X1)
                    CALL ZM_PRINT(X2)
                    WRITE (KU,"(/' These agree to the convergence tolerance, but the previous"//  &
                              " iteration was suspiciously far away:')")
                    CALL ZM_PRINT(X1OLD)
                    WRITE (KU,"(/' and the function value of the last iteration was"//  &
                              " suspiciously far from zero:')")
                    CALL ZM_PRINT(F2)
                    WRITE (KU,"(/' Unknown has been returned.')")
                ENDIF
                X2 = TO_ZM(' UNKNOWN + UNKNOWN i ')
            ENDIF
            GO TO 110
        ENDIF

    ENDDO

!            No convergence after maxit iterations.

    IF (KPRT >= 0) THEN
        WRITE (KU,"(/' No convergence in ZM_SECANT after',I5,' iterations.  ',"//  &
                  "'Last two approximations =')") MAXIT
        CALL ZM_PRINT(X1)
        CALL ZM_PRINT(X2)
        WRITE (KU,"(/' Unknown has been returned.')")
    ENDIF
    X2 = TO_ZM(' UNKNOWN + UNKNOWN i ')
```

```
!             The root was found.

  110 CALL ZM_EQU(X2,ROOT,NDIG,NDSAVE)
      NDIG = NDSAVE
      IF (KPRT >= 1) THEN
          CALL FM_ULP(ABS(X2),ERR1)
          IF (.NOT.( IS_UNKNOWN(ERR1) .OR. IS_UNDERFLOW(ERR1) )) THEN
              ERR1 = ABS(ERR1/X2)/2
              IF (ERR < ERR1) ERR = ERR1
          ENDIF
          STR = FM_FORMAT('ES16.6',ERR)
          WRITE (KU,*) ' '
          WRITE (KU,"('  ZM_SECANT.   Function ',I3,I7,' iterations.'/17X"// &
                    "'Estimated relative error =',A,',    Root:')") NF,J,TRIM(STR)
          CALL ZM_PRINT(ROOT)
          WRITE (KU,*) ' '
      ENDIF

      KWARN = KWARN_SAVE
      CALL FM_EXIT_USER_ROUTINE
      END SUBROUTINE ZM_SECANT
```