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

!  This routine finds a root of F(X,NF) = 0 using AX and BX as starting points.
!  AX and BX do not have to bracket a root in the sense that F(AX,NF) and F(BX,NF) have opposite
!  signs on input.  This means the search can fail if AX and BX are not close enough to any roots
!  or if the function has no real roots or is badly behaved.

!  When a root is found, FM_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 = 0 for no output
!        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 (FM)           :: AX, BX, ROOT
      TYPE (FM), EXTERNAL :: F, FM_FPRIME
      CHARACTER (80) :: STR
      INTEGER :: J, KU, KPRT, KWARN_SAVE, MAXIT, NDSAVE, NF
      LOGICAL :: USE_F_OVER_FP
      TYPE (FM), SAVE :: ERR, ERR1, F1, F2, TOL, X1, X1OLD, X2, X3

!             Subroutines may create temporary multiple precision variables to hold the argument
!             values in cases where an argument might be A+B or TO_FM('1.7').
!             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,*) ' FM_SECANT.  Begin trace of all iterations.'
      ENDIF

!             Raise precision slightly.

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

      MAXIT = 1000
```

```
      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(F1) .OR. IS_OVERFLOW(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(F1) .OR. IS_OVERFLOW(F1))) THEN
                 X1 = X3
                 EXIT
             ENDIF
          ENDDO
          IF (IS_UNKNOWN(F1) .OR. IS_OVERFLOW(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(F1) .OR. IS_OVERFLOW(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(F1) .OR. IS_OVERFLOW(F1))) THEN
                     X1 = X3
                     EXIT
                 ENDIF
              ENDDO
          ENDIF
      ENDIF
      IF (IS_UNKNOWN(F1) .OR. IS_OVERFLOW(F1)) THEN
          WRITE (KU,*) ' '
          WRITE (KU,*) ' Invalid input for FM_SECANT.  ',  &
                       'Unknown or overflowed function value for AX ='
          CALL FM_PRINT(X1)
          WRITE (KU,*) ' '
          J = 0
          X2 = TO_FM(' UNKNOWN ')
          ERR = TO_FM(' UNKNOWN ')
          GO TO 110
      ENDIF

      IF (IS_UNKNOWN(F2) .OR. IS_OVERFLOW(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(F2) .OR. IS_OVERFLOW(F2))) THEN
                 X2 = X3
                 EXIT
             ENDIF
          ENDDO
          IF (IS_UNKNOWN(F2) .OR. IS_OVERFLOW(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(F2) .OR. IS_OVERFLOW(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(F2) .OR. IS_OVERFLOW(F2))) THEN
                   X2 = X3
                   EXIT
               ENDIF
            ENDDO
        ENDIF
      ENDIF
      IF (IS_UNKNOWN(F2) .OR. IS_OVERFLOW(F2)) THEN
          WRITE (KU,*) ' '
          WRITE (KU,*) ' Invalid input for FM_SECANT.  ',  &
                       'Unknown or overflowed function value for BX ='
          CALL FM_PRINT(X2)
          WRITE (KU,*) ' '
          J = 0
          X2 = TO_FM(' UNKNOWN ')
          ERR = TO_FM(' UNKNOWN ')
          GO TO 110
      ENDIF

      IF (KPRT == 2) THEN
          STR = FM_FORMAT('ES20.10',F1)
          WRITE (KU,"('  J =',I3,3X,'f(AX) = ',A,'    x:')") 0,STR(1:25)
          CALL FM_PRINT(X1)
          STR = FM_FORMAT('ES20.10',F2)
          WRITE (KU,"('  J =',I3,3X,'f(BX) = ',A,'    x:')") 0,STR(1:25)
          CALL FM_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
         IF ( (ABS((F2-F1)/(X2-X1)) < 1.0D-2 .AND. ABS(F2) < 1.0D-4) .OR. USE_F_OVER_FP) THEN
             USE_F_OVER_FP = .TRUE.
             X1 = X2
             X2 = X3
```

```
            F1 = F2
            F2 = FM_FPRIME(0,X3,F,NF) / FM_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(F2) .OR. IS_OVERFLOW(F2)) THEN
            F2 = -2*F1
            X3 = X1
            X1 = X2
            X2 = X3
            X3 = F1
            F1 = F2
            F2 = X3
        ENDIF

        IF (KPRT == 2) THEN
            STR = FM_FORMAT('ES20.10',F2)
            WRITE (KU,"('  J =',I3,4X,'f(x) = ' ,A,'    x:')") J,STR(1:25)
            CALL FM_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
                WRITE (KU,"(/' Possible false convergence in FM_SECANT after',I5,"//  &
                          "' iterations.  ','Last two approximations =')") J
                CALL FM_PRINT(X1)
                CALL FM_PRINT(X2)
                WRITE (KU,"(/' These agree to the convergence tolerance, but the previous"//  &
                          " iteration was suspiciously far away:')")
                CALL FM_PRINT(X1OLD)
                WRITE (KU,"(/' and the function value of the last iteration was"//  &
                          " suspiciously far from zero:')")
                CALL FM_PRINT(F2)
            ENDIF
            GO TO 110
        ENDIF
```

```
      ENDDO

!             No convergence after maxit iterations.

      WRITE (KU,"(/' No convergence in FM_SECANT after',I5,' iterations.  ',"//  &
                "'Last two approximations =')") MAXIT
      CALL FM_PRINT(X1)
      CALL FM_PRINT(X2)
      X2 = TO_FM(' UNKNOWN ')
      WRITE (KU,"(/' Unknown has been returned.')")

!             The root was found.

  110 CALL FM_EQU(X2,ROOT,NDIG,NDSAVE)
      NDIG = NDSAVE
      IF (KPRT >= 1) THEN
          CALL FM_ULP(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,"('  FM_SECANT.   Function ',I3,I7,' iterations.'/17X"// &
                    "'Estimated relative error =',A,',    Root:')") NF,J,TRIM(STR)
          CALL FM_PRINT(ROOT)
          WRITE (KU,*) ' '
      ENDIF

      KWARN = KWARN_SAVE
      CALL FM_EXIT_USER_ROUTINE
      END SUBROUTINE FM_SECANT
```