

```

MODULE C2FM_READS
  REAL (KIND(1.0D0)), DIMENSION(:), ALLOCATABLE :: C2FM_TEMP
  REAL (KIND(1.0D0)), DIMENSION(:,:), ALLOCATABLE :: C2FM_TEMP2
END MODULE C2FM_READS

```

```

PROGRAM MINF
USE FMZM
USE C2FM_READS

```

```
! Sample root-finding program.
```

```
! FM_FIND_MIN is a function minimization routine that uses Brent's method.
```

```
! The equation to be solved is  $F(X,NF) = 0$ .
```

```
! X is the argument to the function.
```

```
! NF is the function number in case roots to several functions are needed.
```

```
IMPLICIT NONE
```

```
! CHARACTER(80) :: ST1,ST2
```

```
! Declare the multiple precision variables.
```

```
TYPE (FM) :: A,B,TOL,XVAL,FVAL
```

```
TYPE (FM) , EXTERNAL :: F
```

```
! Set the FM precision to 50 significant digits (plus a few more "guard digits")
```

```
! CALL FM_SET(50)
```

```
! Find a minimum of the first function,  $X**3 - 9*X + 17$ .
```

```
! A, B are two endpoints of an interval in which the search takes place.
```

```
CALL FM_SET(50)
```

```
A = 1
```

```
B = 2
```

```
! TOL is the error tolerance. For most functions, the best accuracy we can obtain
```

```
! corresponds to about half the digits being used for the arithmetic.
```

```
! EPSILON(A) gives the relative accuracy of full precision, so SQRT(EPSILON(A))
```

```
! gives the relative accuracy of half precision.
```

```
TOL = SQRT(EPSILON(A))
```

```
! For this call no trace output will be done (KPRT = 0).
```

```
! KW = 6 is used, so any error messages will go to the screen.
```

```
WRITE (*,*) ' '
```

```
WRITE (*,*) ' '
```

```
WRITE (*,*) ' Case 1. Call FM_FIND_MIN to find a relative minimum between 1 and 2'
```

```
WRITE (*,*) ' for  $f(x) = X**3 - 9*X + 17$ .'
```

```
WRITE (*,*) ' Use KPRT = 0, so no output will be done in the routine, then'
```

```
WRITE (*,*) ' write the results from the main program.'
```

```

CALL FM_FIND_MIN(1,A,B,TOL,XVAL,FVAL,F,1,0,6)

!           Write the result.

!   CALL FM_FORM('F32.30',XVAL,ST1)
!   CALL FM_FORM('F32.30',FVAL,ST2)
!   WRITE (*,"/' A minimum for function 1 is'/'      x = ',A/'   f(x) = ',A)") &
!           TRIM(ST1),TRIM(ST2)
WRITE (*,"/' A minimum for function 1 is'/5X," // &
      "' x = ',ES25.16,',   f(x) = ',ES25.16)") TO_DP(XVAL),TO_DP(FVAL)

!           Find a maximum of the first function,  X**3 - 9*X + 17.

!           This time we use FM_FIND_MIN's built-in trace (KPRT = 1) to print the final
!           approximation to the root.  The output will appear on more than one line, to
!           allow for the possibility that precision could be hundreds or thousands of digits,
!           so the number might not fit on one line.

WRITE (*,*) ' '
WRITE (*,*) ' '
WRITE (*,*) ' Case 2.  Find a relative maximum between -5 and 1.'
WRITE (*,*) '           Use KPRT = 1, so FM_FIND_MIN will print the results.'

CALL FM_FIND_MIN(2,-TO_FM('5.0D0'),TO_FM('1.0D0'),TOL,XVAL,FVAL,F,1,1,6)

!           Find a maximum of the first function,  X**3 - 9*X + 17.

!           See what happens when the maximum value is at an endpoint of the search interval.
!           The algorithm still finds a relative maximum in the interior of the interval,
!           not the absolute maximum at x=5.

WRITE (*,*) ' '
WRITE (*,*) ' '
WRITE (*,*) ' Case 3.  Find a relative maximum between -5 and 5.'
WRITE (*,*) '           Use KPRT = 2, so FM_FIND_MIN will print all iterations,'
WRITE (*,*) '           as well as the final results.'

CALL FM_FIND_MIN(2,-TO_FM('5.0D0'),TO_FM('5.0D0'),TOL,XVAL,FVAL,F,1,2,6)

WRITE (*,*) ' '

END PROGRAM MINF

!   http://www.netlib.no/netlib/go/FVAL.f

!   function FVAL(ax,bx,f,tol)

SUBROUTINE FM_FIND_MIN(MIN_OR_MAX,AX,BX,TOL,XVAL,FVAL,F,NF,KPRT,KW)
USE FMZM
USE C2FM_READS

!   MIN_OR_MAX having value 1 means minimize the function, otherwise maximize.
!   AX, BX      define the endpoints of an interval in which the search takes place.

```

```

! TOL      is the tolerance for the minimum.  Usually TOL should be no less than
!          sqrt(epsilon(ax)), meaning the x-coordinate XVAL of the extreme point will
!          be accurate to only about half the digits carried.  The y-coordinate FVAL
!          should be accurate to nearly full precision.
!          This happens because the typical graph is nearly parabolic near the minimum,
!          and within sqrt(epsilon(ax)) of the minimum all the function values are
!          essentially identical at the current precision.
! XVAL     is returned as the value of X that minimizes (or maximizes) function F(X,NF).
!          It is a relative extreme point, and may not be the global extreme point if the
!          function has more than one extremum on the interval.
! FVAL     is returned as the function value at XVAL.
! F(X,NF)  is the function to be minimized.  X is the argument and NF is the function
!          number, in case several functions are defined within F.
! 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.
! KW       is the unit number for output.

! The method used is a combination of golden section search and successive parabolic interpolation.
! Convergence is never much slower than that for a fibonacci search.  If f has a continuous second
! derivative which is positive at the minimum (which is not at ax or bx), then convergence is
! superlinear, and usually of the order of about 1.324....

! The function f is never evaluated at two points closer together than eps*abs(FVAL)+(tol/3),
! where eps is approximately the square root of the relative machine precision.  If f is a
! unimodal function and the computed values of f are always unimodal when separated by at least
! eps*abs(x)+(tol/3), then FVAL approximates the abscissa of the global minimum of f on the
! interval ax,bx with an error less than 3*eps*abs(FVAL)+tol.  If f is not unimodal, then FVAL
! may approximate a local, but perhaps non-global, minimum to the same accuracy.

! This routine is a slightly modified translation of function FVAL from netlib, which was adapted
! from the algol 60 procedure localmin given by Richard Brent in Algorithms For Minimization
! Without Derivatives, Prentice-Hall (1973).

      IMPLICIT NONE
!      CHARACTER(80)  :: ST1,ST2
      INTEGER  :: J,MINV,MIN_OR_MAX,NF,KPRT,KW
      TYPE (FM)  :: AX,BX,TOL,XVAL,FVAL
      TYPE (FM) , EXTERNAL  :: F
      TYPE (FM)  :: A,B,C,D,E,EPS,XM,P,Q,R,TOL1,T2,U,V,W,FU,FV,FW,FX,X,TOL3

      SAVE  :: A, B, C, D, E, EPS, XM, P, Q, R, TOL1, T2, U, V, W, FU, FV, FW, FX, X, TOL3
      CALL FM_ENTER_USER_ROUTINE

      MINV = 1
      IF (MIN_OR_MAX /= 1) MINV = -1

!          C is the squared inverse of the golden ratio.

      C = (3-SQRT(TO_FM('5.0D0')))/2

!          EPS is approximately the square root of the relative machine precision.

```

```

EPS = EPSILON(AX)
TOL1 = EPS + 1
EPS = SQRT(EPS)

```

```

A = MIN(AX,BX)
B = MAX(AX,BX)
V = A + C*(B-A)
W = V
X = V
E = 0
FX = F(X,NF)*MINV
FV = FX
FW = FX
TOL3 = TOL/3
J = 1

```

```

IF (KPRT == 2) THEN
  WRITE (KW,*) ' '
  IF (MIN_OR_MAX == 1) THEN
    WRITE (KW,*) ' FM_FIND_MIN. Begin trace of all iterations.'
    WRITE (KW,*) ' Search for a relative minimum on the interval'
    WRITE (KW,"(13X,ES20.10,' to ',ES20.10/)" TO_DP(AX),TO_DP(BX)
  ELSE
    WRITE (KW,*) ' FM_FIND_MIN. Begin trace of all iterations.'
    WRITE (KW,*) ' Search for a relative maximum on the interval'
    WRITE (KW,"(13X,ES20.10,' to ',ES20.10/)" TO_DP(AX),TO_DP(BX)
  ENDIF
  ! ST1 = FM_FORMAT('ES35.25',X)
  ! ST2 = FM_FORMAT('ES35.25',FX*MINV)
  ! WRITE (KW,"(' J =',I3,4X,' x = ',A)") J,TRIM(ST1)
  ! WRITE (KW,"(' ',3X,4X,' f(x) = ',A/)" TRIM(ST2)
  WRITE (KW,"(' J =',I3,4X,' X =',ES25.16,' F(X) =',ES25.16)" J,TO_DP(X), &
    TO_DP(FX)*MINV
ENDIF

```

```

! The main loop starts here.

```

```

110 XM = (A+B)/2
TOL1 = EPS*ABS(X) + TOL3
T2 = 2*TOL1

```

```

! Check the stopping criterion.

```

```

IF (ABS(X-XM) <= (T2-(B-A)/2)) GO TO 160
P = 0
Q = 0
R = 0
IF (ABS(E) > TOL1) THEN
  R = (X-W)*(FX-FV) ! Fit a parabola.
  Q = (X-V)*(FX-FW)
  P = (X-V)*Q-(X-W)*R
  Q = 2*(Q-R)
  IF (Q > 0) THEN
    P = -P
  ELSE
    Q = -Q
  ENDIF
ENDIF

```

```

ENDIF
R = E
E = D
ENDIF

IF ((ABS(P) >= ABS(Q*R/2)) .OR. (P <= Q*(A-X)) .OR. (P >= Q*(B-X))) GO TO 120

!           Make a parabolic-interpolation step.

D = P/Q
U = X + D

!           f must not be evaluated too close to ax or bx.

IF (((U-A) >= T2) .AND. ((B-U) >= T2)) GO TO 130
D = TOL1
IF (X >= XM) D = -D
GO TO 130

!           Make a golden-section step.

120 IF (X < XM) THEN
    E = B - X
ELSE
    E = A - X
ENDIF
D = C*E

!           f must not be evaluated too close to x.

130 IF (ABS(D) >= TOL1) THEN
    U = X + D
ELSE
    IF (D > 0) THEN
        U = X + TOL1
    ELSE
        U = X - TOL1
    ENDIF
ENDIF
FU = F(U,NF)*MINV

J = J + 1
IF (KPRT == 2) THEN
!           ST1 = FM_FORMAT('ES35.25',U)
!           ST2 = FM_FORMAT('ES35.25',FU*MINV)
!           WRITE (KW,"('      J =',I3,4X,' x = ',A)") J,TRIM(ST1)
!           WRITE (KW,"('      ',3X,4X,' f(x) = ',A/)") TRIM(ST2)
WRITE (KW,"('      J =',I3,4X,' X =',ES25.16,'      F(X) =',ES25.16)") J,TO_DP(U), &
    TO_DP(FU)*MINV
ENDIF

!           update a, b, v, w, and x.

IF (FX <= FU) THEN
    IF (U < X) THEN
        A = U

```

```

ELSE
    B = U
ENDIF
ENDIF
IF (FU > FX) GO TO 140
IF (U < X) THEN
    B = X
ELSE
    A = X
ENDIF
V = W
FV = FW
W = X
FW = FX
X = U
FX = FU
GO TO 110

140 IF ((FU > FW) .AND. (W /= X)) GO TO 150
    V = W
    FV = FW
    W = U
    FW = FU
    GO TO 110

150 IF ((FU > FV) .AND. (V /= X) .AND. (V /= W)) GO TO 110
    V = U
    FV = FU
    GO TO 110

! end of main loop

160 XVAL = X
    FVAL = FX*MINV

IF (KPRT >= 1) THEN
    WRITE (KW,*) ' '
    IF (MIN_OR_MAX == 1) THEN
        WRITE (KW, &
            "(' FM_FIND_MIN. Function ',I3,I6,' iterations. A relative minimum"// &
            " on interval'/13X,ES20.10,' to ',ES20.10,' is')")NF,J,TO_DP(AX),TO_DP(BX)
!         ST1 = FM_FORMAT('ES35.25',XVAL)
!         ST2 = FM_FORMAT('ES35.25',FVAL)
!         WRITE (KW,"(15X,' x = ',A)") TRIM(ST1)
!         WRITE (KW,"(15X,' f(x) = ',A)") TRIM(ST2)
        WRITE (KW,"(15X,' x =',ES24.16,', f(x) =',ES24.16)") TO_DP(XVAL),TO_DP(FVAL)
    ELSE
        WRITE (KW, &
            "(' FM_FIND_MIN. Function ',I3,I6,' iterations. A relative maximum"// &
            " on interval'/13X,ES20.10,' to ',ES20.10,' is')")NF,J,TO_DP(AX),TO_DP(BX)
!         ST1 = FM_FORMAT('ES35.25',XVAL)
!         ST2 = FM_FORMAT('ES35.25',FVAL)
!         WRITE (KW,"(15X,' x = ',A)") TRIM(ST1)
!         WRITE (KW,"(15X,' f(x) = ',A)") TRIM(ST2)
        WRITE (KW,"(15X,' x =',ES24.16,', f(x) =',ES24.16)") TO_DP(XVAL),TO_DP(FVAL)
    ENDIF
ENDIF

```

```
WRITE (KW,*) ' '
ENDIF
```

```
CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_FIND_MIN
```

```
FUNCTION F(X,NF)
USE FMZM
USE C2FM_READS
```

```
! X is the argument to the function.
! NF is the function number.
```

```
IMPLICIT NONE
INTEGER :: NF
TYPE (FM) :: F,X
```

```
CALL FM_ENTER_USER_FUNCTION(F)
```

```
IF (NF == 1) THEN
  F = X**3 - 9*X + 17
ELSE IF (NF == 2) THEN
  F = X*TAN(X) - 1
ELSE
  F = 3*X**2 + X - 2
ENDIF
```

```
CALL FM_EXIT_USER_FUNCTION(F)
END FUNCTION F
```