```
      RECURSIVE SUBROUTINE FM_INTEGRATE(F,N,A,B,TOL,RESULT,KPRT,NW)
      USE FMVALS
      USE FMZM
      IMPLICIT NONE

!  High-precision numerical integration.

!  Integrate F(X,N) from A to B.  N is passed on to function F to indicate which function to use in
!  cases where several different functions may be defined there.


!  WARNING:  If the function F being integrated or one of its derivatives does not exist at one or
!            both of the endpoints (A,B), the endpoints should be exactly representable in FM's
!            number system.  For non-exact numbers like 1/3, sqrt(2), or pi/2, when FM_INTEGRATE
!            raises precision to evaluate the integration formula the endpoints are not accurate
!            enough at the higher precision.

!  Example:  Integrate  sqrt( tan( x ) )  from  0  to  pi/2.
!            First, pi/2 is not exact as an FM number.  At some precisions it may have rounded up,
!            making tan(x) negative and causing an error in sqrt.  Using sqrt( abs( tan( x ) ) )
!            is safer.
!            Second, b = pi/2 has been computed at the user's precision, so when FM_INTEGRATE
!            raises precision, the value of b is just zero-padded on the end, which does not give
!            enough information about how f(x) behaves near the singularity at pi/2.
!
!            Make the endpoints exact by changing variables.  Change the interval to [ 0 , 1 ]:
!
!                                    u = ( 2/pi ) * x   =>   du = ( 2/pi ) * dx
!            so
!                                    x = ( pi/2 ) * u   =>   dx = ( pi/2 ) * du
!            new limits
!                                    x = 0  =>  u = 0  and  x = pi/2  =>  u = 1
!
!            New integral:   Integrate  (pi/2) * sqrt( abs( tan( pi*u/2 ) ) )  from  0  to  1.
!
!            Now the function F should declare a local saved type(fm) variable PI, and then use
!            CALL FM_PI(PI) each time F is called to make sure the value of PI is correct at the
!            higher precision being used by FM_INTEGRATE when F is called.

      TYPE (FM) :: A,B,RESULT,TOL
      TYPE (FM), EXTERNAL :: F
      INTEGER :: N,KPRT,NW
      INTENT (IN) :: N,A,B,TOL,KPRT,NW
      INTENT (INOUT) :: RESULT

!  A,B,TOL, and RESULT are all type (FM) variables, and function F returns a type (FM) result.

!  RESULT is returned as the value of the integral, with ABS((RESULT-true)/true) less than TOL
!         (i.e., TOL is a relative error tolerance).
!         For example, to get 30 significant digits correct for the integral, set TOL = 1.0E-30.

!  FM precision must be set high enough in the calling program (using FM_SET) so that 1+TOL > 1
!  at that precision.  Using TOL = EPSILON(TO_FM(1)) will usually get a full precision result,
!  but for some functions this might fail.  A better strategy is to set precision higher than
!  the accuracy required for the integral.  For example, to get the integral to 50 significant
```

```
!  digits, CALL FMSET(60) and then set TOL = TO_FM(' 1.0E-50 ') before the call to FM_INTEGRATE.

!  KPRT can be used to show intermediate results on unit NW.
!  KPRT = 0 for no output
!       = 1 prints a summary for each call to FM_INTEGRATE
!       = 2 prints a trace of all iterations.

!  NW is the unit number used for KPRT output and any error or warning messages.

!  No method for numerical integration is foolproof.  Since it samples only a finite number of
!  function values, any numerical integration algorithm can be made to fail by choosing a
!  sufficiently badly-behaved function.  Such functions often vary by many orders of magnitude
!  over relatively small fractions of the interval (A,B).

!  F should be well-behaved in the interior of the interval (A,B).
!  The routine tries to handle any singularities of F or F' at A and/or B, so cases with interior
!  singularities should be done as separate calls having the singularities as endpoints.
!  The routine will try to handle cases where F or F' has singularities inside (A,B), but then
!  the time will be much slower and the routine might fail.

!  For a function with a removable singularity in the interior of the interval, such as
!  f(x) = 1/ln(x) - 1/(x-1), define F(X,N) to check for X = 1 and return the correct limiting
!  value, 0.5 in this case, when X is 1.

!  Among functions with no singularities, examples of badly behaved functions are those with one
!  or more extremely narrow tall spikes in their graphs. If possible, identify the peaks of any
!  such spikes first, then make separate calls with the peaks as endpoints of the intervals of
!  integration.

!  If the value of the integral is zero or very close to zero, relative error may be undefined, so
!  this routine may fail to converge and then return unknown.  For these cases, try breaking the
!  integral into two pieces and calling twice to get two non-zero results.  These two results can
!  then be added, often giving the original integral with sufficiently small absolute error even
!  though small relative error could not be attained.

!  If the function values are too extreme, it can cause problems.  For example, if an exponential
!  in F underflows and then is multiplied by something bigger than one, then F will return unknown.
!  If the result of the integral is much larger than the underflow threshold (TINY(TO_FM(1))), then
!  it is safe to set the underflowed results in F to zero to avoid getting unknown.

!  If the function is nearly divergent FM_INTEGRATE may fail.  1/x from 0 to b is divergent.
!  1/x**0.99 converges, but so slowly that FM_INTEGRATE may run a long time and then might fail.
!  1/x**0.9999 converges even more slowly and FM_INTEGRATE may fail by declaring that the integral
!  seems divergent.

!  When the integrand is highly (or infinitely) oscillatory, FM_INTEGRATE may fail.
!  If F has more than about 100 oscillations on the interval (A,B), it may be necessary to break
!  the interval into smaller intervals and call FM_INTEGRATE several times.
!  For infinitely many oscillations, like sin(1/x) from 0 to 1, first turn the integral into an
!  infinite series by calling FM_INTEGRATE to integrate each separate loop between roots of
!  sin(1/x).  The function is well-behaved for each call, so FM_INTEGRATE can get high precision
!  quickly for each.  Next form a sequence of k partial sums for this series.  The series converges
!  slowly, with 50 or 100 terms giving only 3 or 4 significant digits of the sum, so an
!  extrapolation method can be used to get a more accurate value of the sum of this series from
!  its first k terms. For an alternating series like this, the extrapolation method of Cohen,
!  Villegas, and Zagier often works very well.
```

```
!  Repeated Aitken extrapolation could be used instead -- it is a more widely known method.
!  Sample program Oscillate.f95 computes this integral.


!             M  is the maximum level for the integration algorithm.  The number of function
!                evaluations roughly doubles for each successive level until the tolerance is met.
!                Using M = 12 allows up to about 5,000 digits for most integrals, but the upper
!                limit for a given M depends on the function.
!                Raising M further will approximately double the maximum precision for each
!                extra level, but will also double the memory usage for each extra level.

      INTEGER, PARAMETER :: M = 12
      INTEGER, PARAMETER :: NT = 20*2**M
      TYPE (FM), SAVE :: ST_SAVE(0:NT)
      INTEGER, SAVE :: PRECISION_OF(0:NT) = 0
      INTEGER :: ABSIGN,I,ISTEP,K,KWSAVE,NDS,NDSAVE,NRETRY,NUMBER_USED_SAVE
      INTEGER, SAVE :: R_LEVEL = 0, NUM_F = 0

!             Local TYPE (FM) variables can be tricky in recursive routines.  If they are declared
!             with the SAVE attribute and initialized as usual, they will be shared among
!             all levels of the recursive calls.  Here we need them to be different for each level
!             of recursion, so we initialize them with index -2 and do not make them saved.
!             To avoid leaking memory, we also save and restore NUMBER_USED in the same way as the
!             low-level routines in FM.

      TYPE (FM) :: A1,AB2,B1,C1,C2,CI,CT,D,EPS,ERR1,ERR2,FMAX,FMAX2,H,HF,LAST_H,PI,PRIOR_HS,  &
                   S,S1,S2,SI,ST,T,TOL1,TOL2,X,XF,XMAX,V,W

      CHARACTER(80) :: ST1,ST2
      REAL :: TIME1,TIME2
      LOGICAL :: SPIKE_FOUND,ST_IS_SAVED

      CALL FM_ENTER_USER_ROUTINE

!             Initialize the index values for the local FM variables.
!             This cannot be done in the TYPE statement above, or with a DATA statement, because
!             those forms of initialization force the SAVE attribute to be applied to the variables.

      CALL FM_INT_INIT(A1,AB2,B1,C1,C2,CI,CT,D,EPS,ERR1,ERR2,FMAX,FMAX2,H,HF,LAST_H,PI,PRIOR_HS,  &
                       S,S1,S2,SI,ST,T,TOL1,TOL2,X,XF,XMAX,V,W)

!             Iterative tanh-sinh integration is used, increasing the order until convergence
!             is obtained, or M levels have been done.

      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      RESULT = 0
      NUMBER_USED_SAVE = NUMBER_USED

      NCALL = NCALL + 1
      NAMEST(NCALL) = 'INTEGRATE'
      KWSAVE = KW
      KW = NW
      R_LEVEL = R_LEVEL + 1
      NRETRY = 0
      IF (KPRT >= 2) THEN
          WRITE (NW,"(A)") ' '
```

```
          WRITE (NW,"(A,I9,A)") ' Input to FM_INTEGRATE.    Function N = ',N,'.    A, B ='
          CALL FM_PRINT(A)
          CALL FM_PRINT(B)
          CALL FM_FORM('ES20.8',TOL,ST1)
          WRITE (NW,"(A,A)") ' TOL =',TRIM(ST1)
       ENDIF
       CALL CPU_TIME(TIME1)

!             Check for special cases.

       IF (A == B) THEN
          A1 = F(A,N)
          IF (R_LEVEL <= 1) THEN
             NUM_F = 1
          ELSE
             NUM_F = NUM_F + 1
          ENDIF
          IF (MWK(START(A1%MFM)+2) == MUNKNO) THEN
             CALL FM_F_FAIL(A,N,NW)
             PRIOR_HS = A1
             GO TO 120
          ELSE
             PRIOR_HS = 0
             GO TO 120
          ENDIF
       ENDIF

!             Check to make sure the user has set precision high enough for the value of TOL chosen.

       TOL1 = TOL
       IF (TOL < ABS(EPSILON(A))) THEN
          WRITE (NW,"(A)") ' '
          WRITE (NW,"(A)") ' Error in FM_INTEGRATE.  TOL is '
          CALL FM_PRINT(TOL)
          WRITE (NW,"(A)") ' This is too small for the current precision level.  Current epsilon ='
          TOL1 = ABS(EPSILON(A))
          CALL FM_PRINT(TOL1)
          WRITE (NW,"(A)") ' This larger value will be used.  TOL ='
          CALL FM_PRINT(TOL1)
          WRITE (NW,"(A)") ' Use FM_SET to set a higher precision before the call to'
          WRITE (NW,"(A)") ' FM_INTEGRATE if the smaller TOL is needed.'
       ENDIF

!             Raise the precision.
!             Check for an integrable singularity at either endpoint, and increase precision
!             if it seems that a retry would be needed at the first precision.

       NDSAVE = NDIG
       A1 = LOG10(F(A+(B-A)*TO_FM(' 1.0E-10 '),N)/F(A+(B-A)*TO_FM(' 1.0E-20 '),N))/10
       B1 = LOG10(F(B-(B-A)*TO_FM(' 1.0E-10 '),N)/F(B-(B-A)*TO_FM(' 1.0E-20 '),N))/10
       IF (R_LEVEL <= 1) THEN
          NUM_F = 4
       ELSE
          NUM_F = NUM_F + 4
       ENDIF
       IF (A1 < -0.999 .OR. B1 < -0.999) THEN
```

```
          PRIOR_HS = TO_FM(' UNKNOWN ')
          WRITE (NW,"(A)") ' '
          WRITE (NW,"(A,I9,A)") ' FM_INTEGRATE failed -- F(X,N) for N = ',N,  &
                                ' seems to have a non-integrable singularity'
          WRITE (NW,"(A)")      '                          at an endpoint.  A,B ='
          CALL FM_PRINT(A)
          CALL FM_PRINT(B)
          WRITE (NW,"(A)") ' Check the limits of integration, function number (N), and' // &
                           ' function definition.'
          WRITE (NW,"(A)") ' '
          GO TO 120
       ENDIF
       NDIG = NDIG+INT(30/ALOGMT)
       CALL FM_EQU_R1(A1,NDSAVE,NDIG)
       CALL FM_EQU_R1(B1,NDSAVE,NDIG)
       IF (A1 < -0.2 .OR. B1 < -0.2) NDIG = 2*NDIG

       CALL CPU_TIME(TIME1)

!             Start here when doing a retry.

  110  NRETRY = NRETRY + 1
       CALL FM_EQU(A,A1,NDSAVE,NDIG)
       CALL FM_EQU(B,B1,NDSAVE,NDIG)
       ABSIGN = 1
       IF (A1 > B1) THEN
          CALL FM_EQU(B,A1,NDSAVE,NDIG)
          CALL FM_EQU(A,B1,NDSAVE,NDIG)
          ABSIGN = -1
       ELSE IF (A1 == B1) THEN
          PRIOR_HS = 0
          GO TO 120
       ENDIF
       CALL FM_EQU(TOL1,TOL2,NDSAVE,NDIG)

       IF (KPRT >= 2) THEN
          WRITE (NW,"(A)") ' '
          WRITE (NW,"(A,I9,A,I5)") ' Begin FM_INTEGRATE.  NDIG = ',NDIG,'   Recursion level = ',  &
                                   R_LEVEL
       ENDIF

       S = 0
       PRIOR_HS = 0
       ERR2 = 1
       EPS = EPSILON(TOL2)
       D = ABS(B1-A1)/100
       FMAX = 0
       FMAX2 = 0
       XMAX = A1
       H = 1
       LAST_H = H/2**M
       CALL FM_PI(PI)

       HF = (B1-A1)/2
       AB2 = A1 + HF
       DO K = 1, M
```

```
          H = H/2
          ISTEP = 2**(M-K)
          IF (K > 1) THEN
              T = ISTEP*LAST_H
              CALL FM_CHSH(T,C1,S1)
              T = 2*S1*C1
              C2 = C1**2 + S1**2
              S2 = T
          ENDIF
          DO I = 0, NT, ISTEP
             IF (MOD(I,2*ISTEP) /= 0 .OR. K == 1) THEN

!                The + or -X values are the abscissas for interval (-1,1).
!                XF translates these to the interval (A,B).

                IF (I == 0) THEN
                    X = 0
                    W = PI/2
                    XF = HF*X + AB2
                    T = F(XF,N)
                    NUM_F = NUM_F + 1
                    IF (MWK(START(T%MFM)+2) == MUNKNO) THEN
                        CALL FM_F_FAIL(XF,N,NW)
                        PRIOR_HS = T
                        GO TO 120
                    ENDIF
                    FMAX2 = MAX(FMAX2,ABS(T))
                    IF (ABS(T) > FMAX .AND. XF > A1+D .AND. XF < B1-D) THEN
                        FMAX = ABS(T)
                        XMAX = XF
                    ENDIF
                    S = S + W*HF*T
                ELSE
                    IF (K == 1) THEN
                        T = I*LAST_H
                        CALL FM_CHSH(T,CI,SI)
                    ELSE

!                       Use the hyperbolic addition formulas to get the next cosh and sinh
!                       quickly when evaluated at I*LAST_H.

                        IF (I == ISTEP) THEN
                            CI = C1
                            SI = S1
                        ELSE
                            T  = SI*C2 + CI*S2
                            CI = CI*C2 + SI*S2
                            SI = T
                            C1 = CI
                            S1 = SI
                        ENDIF
                    ENDIF
                    ST_IS_SAVED = .FALSE.
                    IF (ST_SAVE(I)%MFM > 0) THEN
                        IF (PRECISION_OF(I) >= NDIG) ST_IS_SAVED = .TRUE.
                    ENDIF
```

```
            IF (ST_IS_SAVED) THEN
                ST = ST_SAVE(I)
                CT = SQRT(1+ST**2)
            ELSE
                T = PI*SI/2
                CALL FM_CHSH(T,CT,ST)
                ST_SAVE(I) = ST
                PRECISION_OF(I) = NDIG
            ENDIF
            W = (PI/2)*CI/CT**2
            IF (W < EPS) EXIT
            X = ST/CT
            XF = HF*(-X) + AB2
            IF (XF > A1) THEN
                T = F(XF,N)
                NUM_F = NUM_F + 1
                IF (MWK(START(T%MFM)+2) == MUNKNO) THEN
                    CALL FM_F_FAIL(XF,N,NW)
                    PRIOR_HS = T
                    GO TO 120
                ENDIF
                FMAX2 = MAX(FMAX2,ABS(T))
                IF (ABS(T) > FMAX .AND. XF > A1+D .AND. XF < B1-D) THEN
                    FMAX = ABS(T)
                    XMAX = XF
                ENDIF
                S = S + W*HF*T
            ENDIF
            XF = HF*(X) + AB2
            IF (XF < B1) THEN
                T = F(XF,N)
                NUM_F = NUM_F + 1
                IF (MWK(START(T%MFM)+2) == MUNKNO) THEN
                    CALL FM_F_FAIL(XF,N,NW)
                    PRIOR_HS = T
                    GO TO 120
                ENDIF
                FMAX2 = MAX(FMAX2,ABS(T))
                IF (ABS(T) > FMAX .AND. XF > A1+D .AND. XF < B1-D) THEN
                    FMAX = ABS(T)
                    XMAX = XF
                ENDIF
                S = S + W*HF*T
            ENDIF
        ENDIF
    ENDIF
ENDDO
IF (KPRT >= 2) THEN
    WRITE (NW,"(A)") ' '
    WRITE (NW,"(A,I9,A,I9,A)") ' K = ',K,'    ',NUM_F,  &
                               ' function calls so far.   Integral approximation ='
    V = H*S
    CALL FM_PRINT(V)
ENDIF
IF (K > 1) THEN
    ERR1 = ERR2
```

```
            IF (S /= 0) THEN
                ERR2 = ABS( (PRIOR_HS - H*S)/(H*S) )
            ELSE
                ERR2 = ABS( (PRIOR_HS - H*S) )
            ENDIF
            IF (KPRT >= 2) THEN
                CALL FM_FORM('ES15.3',ERR2,ST1)
                WRITE (NW,"(A,A)") '      relative error of the last two approximations = ',  &
                                   TRIM(ST1)
            ENDIF

!            Check for convergence.

            IF (K > 3 .AND. ERR2 > 0 .AND. ERR2 < TOL2/10.0) EXIT
            IF (K > 5 .AND. ERR2 == 0) EXIT

!            If the errors do not decrease fast enough, raise precision and try again.

            IF (K > 3*NRETRY .AND. ERR1 > 0 .AND. ERR2 > 0) THEN
                IF (LOG(ERR2)/LOG(ERR1) < 1.2 .AND. ERR1 < 1.0D-6) THEN
                    NDIG = 2*NDIG
                    IF (KPRT >= 2) THEN
                        WRITE (NW,"(A,I9,A,I9)") ' FM_INTEGRATE Retry.  So far, NUM_F = ',NUM_F,  &
                                                 '   New NDIG = ',NDIG
                    ENDIF
                    IF (NRETRY <= 3) GO TO 110
                    NDIG = NDIG/2
                ENDIF
            ENDIF
        ENDIF
        PRIOR_HS = H*S

!            No convergence in M iterations.
!            Before giving up, look for an interior singularity or tall spike.  If one is found,
!            split (A,B) into two intervals with the interior singularity as an endpoint, and try
!            again as two integrals.

        IF (K == M .OR. (K >= 9 .AND. ERR2 > 1.0D-7 .AND. ABS(TOL2) < 1.0D-16)) THEN
            IF (KPRT >= 2) THEN
                WRITE (NW,"(A)") ' '
                WRITE (NW,"(A,I6,A)") ' No convergence in ',M,  &
                              ' iterations.  Look for an interior singularity.'
                CALL FM_FORM('ES25.6',XMAX,ST1)
                CALL FM_FORM('ES25.6',FMAX,ST2)
                WRITE (NW,"(I9,A,A,A)") NUM_F,' function calls so far.   XMAX, FMAX =',  &
                                        TRIM(ST1),TRIM(ST2)
            ENDIF
            CALL FM_SPIKE(F,N,A1,B1,XMAX,FMAX,NUM_F,SPIKE_FOUND,KPRT,NW)
            CALL FMEQU_R1(A1%MFM,NDIG,NDSAVE)
            CALL FMEQU_R1(B1%MFM,NDIG,NDSAVE)
            CALL FMEQU_R1(XMAX%MFM,NDIG,NDSAVE)
            NDS = NDIG
            NDIG = NDSAVE
            IF (SPIKE_FOUND) THEN
                IF (MIN(ABS(A-XMAX),ABS(B-XMAX)) < 1.01*D) THEN
                    NDIG = 2*NDS
```

```
                    IF (NRETRY <= 5) GO TO 110
                    NDIG = NDSAVE
                ENDIF
                IF (KPRT >= 2) THEN
                    WRITE (NW,"(A)") ' '
                    WRITE (NW,"(A)") ' Split the integral.  First half:  A,B = '
                    CALL FM_PRINT(A1)
                    CALL FM_PRINT(XMAX)
                ENDIF
                CALL FM_INTEGRATE(F,N,A1,XMAX,TOL,C1,KPRT,NW)
                IF (MWK(START(C1%MFM)+2) == MUNKNO) THEN
                    PRIOR_HS = C1
                    GO TO 120
                ENDIF
                IF (KPRT >= 2) THEN
                    WRITE (NW,"(A)") ' '
                    WRITE (NW,"(A)") ' Split the integral.  Second half:  A,B = '
                    CALL FM_PRINT(XMAX)
                    CALL FM_PRINT(B1)
                ENDIF
                CALL FM_INTEGRATE(F,N,XMAX,B1,TOL,C2,KPRT,NW)
                PRIOR_HS = C1 + C2
                GO TO 120
            ENDIF
            CALL FM_INT_FAIL(N,A,B,TOL,M,ERR2,PRIOR_HS,NW)
            PRIOR_HS = TO_FM(' UNKNOWN ')
            GO TO 120
        ENDIF
    ENDDO

    PRIOR_HS = ABSIGN*H*S

!             Round the result and return.

  120 CALL FM_EQU(PRIOR_HS,RESULT,NDIG,NDSAVE)

    NDIG = NDSAVE
    NCALL = NCALL - 1
    CALL CPU_TIME(TIME2)

    IF (KPRT >= 2 .OR. ( R_LEVEL <= 1 .AND. KPRT == 1 ) ) THEN
        WRITE (NW,"(A)") ' '
        WRITE (NW,"(A,I9,A)") ' Return from FM_INTEGRATE.    Function N = ',N,'.    A, B ='
        CALL FM_PRINT(A)
        CALL FM_PRINT(B)
        CALL FM_FORM('ES20.8',TOL,ST1)
        WRITE (NW,"(A,A)") ' TOL =',TRIM(ST1)
        IF (ABS(TIME2-TIME1) > 0.0001 .AND. ABS(TIME2-TIME1) < 1000.0) THEN
            WRITE (NW,"(1X,I9,A,F9.5,A)") NUM_F,' function calls were made in ',TIME2-TIME1,  &
                                          ' seconds.'
            WRITE (NW,"(A)") ' RESULT ='
        ELSE
            WRITE (NW,"(1X,I9,A,ES14.5,A)") NUM_F,' function calls were made in ',TIME2-TIME1,  &
                                            ' seconds.'
            WRITE (NW,"(A)") ' RESULT ='
        ENDIF
```

```
    CALL FM_PRINT(RESULT)
ENDIF
KW = KWSAVE
R_LEVEL = R_LEVEL - 1
TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
NUMBER_USED = NUMBER_USED_SAVE

CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_INTEGRATE

SUBROUTINE FM_INT_INIT(A1,AB2,B1,C1,C2,CI,CT,D,EPS,ERR1,ERR2,FMAX,FMAX2,H,HF,LAST_H,PI,  &
                       PRIOR_HS,S,S1,S2,SI,ST,T,TOL1,TOL2,X,XF,XMAX,V,W)
USE FMZM
IMPLICIT NONE
TYPE (FM) :: A1,AB2,B1,C1,C2,CI,CT,D,EPS,ERR1,ERR2,FMAX,FMAX2,H,HF,LAST_H,PI,PRIOR_HS,  &
             S,S1,S2,SI,ST,T,TOL1,TOL2,X,XF,XMAX,V,W
CALL FM_ENTER_USER_ROUTINE
A1%MFM = -2
AB2%MFM = -2
B1%MFM = -2
C1%MFM = -2
C2%MFM = -2
CI%MFM = -2
CT%MFM = -2
D%MFM = -2
EPS%MFM = -2
ERR1%MFM = -2
ERR2%MFM = -2
FMAX%MFM = -2
FMAX2%MFM = -2
H%MFM = -2
HF%MFM = -2
LAST_H%MFM = -2
PI%MFM = -2
PRIOR_HS%MFM = -2
S%MFM = -2
S1%MFM = -2
S2%MFM = -2
SI%MFM = -2
ST%MFM = -2
T%MFM = -2
TOL1%MFM = -2
TOL2%MFM = -2
X%MFM = -2
XF%MFM = -2
XMAX%MFM = -2
V%MFM = -2
W%MFM = -2
CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_INT_INIT

SUBROUTINE FM_INT_FAIL(N,A,B,TOL,M,ERR,VAL,NW)
USE FMZM
IMPLICIT NONE
INTEGER :: N,M,NW
TYPE (FM) :: A,B,TOL,ERR,VAL
```

```
CALL FM_ENTER_USER_ROUTINE
WRITE (NW,*) ' '
WRITE (NW,*) ' FM_INTEGRATE failed -- no convergence in ',M,' iterations.'
WRITE (NW,*) ' UNKNOWN has been returned in RESULT.'
WRITE (NW,*) ' Possible causes:  (1) highly oscillatory integrand'
WRITE (NW,*) '                   (2) non-convergent integral'
WRITE (NW,*) '                   (3) integrable singularity in the interior of interval (A,B)'
WRITE (NW,*) '                   (4) narrow spike in the interior of interval (A,B)'
WRITE (NW,*) '                   (5) integral too close to zero'
WRITE (NW,*) ' A possible remedy for the last 3 is to split the integral into two pieces,'
WRITE (NW,*) ' making two calls to FM_INTEGRATE and then adding the two results.'
WRITE (NW,*) ' Put singularities or spikes at the endpoints of the intervals of integration.'
WRITE (NW,*) ' '
WRITE (NW,*) ' Function N = ',N,'.    A, B ='
CALL FM_PRINT(A)
CALL FM_PRINT(B)
WRITE (NW,*) ' TOL ='
CALL FM_PRINT(TOL)
WRITE (NW,*) ' The last integral approximation ='
CALL FM_PRINT(VAL)
WRITE (NW,*) ' The estimated relative error in the last integral approximation ='
CALL FM_PRINT(ERR)
WRITE (NW,*) ' '

CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_INT_FAIL

SUBROUTINE FM_F_FAIL(X,N,NW)
USE FMZM
IMPLICIT NONE
INTEGER :: N,NW
TYPE (FM) :: X

CALL FM_ENTER_USER_ROUTINE
WRITE (NW,*) ' '
WRITE (NW,*) ' FM_INTEGRATE failed -- F(X,N) gave UNKNOWN for N = ',N,'   and X ='
CALL FM_PRINT(X)
WRITE (NW,*) ' Check the limits of integration, function number (N), and' // &
             ' function definition.'
WRITE (NW,*) ' Be careful of rounding at an irrational endpoint producing an' // &
             ' illegal function argument.'
WRITE (NW,*) ' Example:  Integrate  log( cos( t ) ) from 0 to pi/2'
WRITE (NW,*) ' At some precisions the computed value of pi/2 rounds up, giving' // &
             ' a small negative cos(t),'
WRITE (NW,*) ' which then causes the log function to return UNKNOWN.'
WRITE (NW,*) ' Possible fixes:'
WRITE (NW,*) '     Change variables to get an integral from 0 to 1'
WRITE (NW,*) '     Change the function to log( abs( cos( t ) ) )'
WRITE (NW,*) '     Compute pi/2 with rounding toward -infinity'
WRITE (NW,*) ' '

CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_F_FAIL

SUBROUTINE FM_SPIKE(F,N,A,B,XMAX,FMAX,NUM_F,SPIKE_FOUND,KPRT,NW)
```

```
      USE FMVALS
      USE FMZM
      IMPLICIT NONE

!  look for an interior singularity or tall spike in F(X,N).
!  After getting no convergence in M iterations in FM_INTEGRATE, FMAX = ABS(F(XMAX,N)) was
!  the largest magnitude found for F on the interval (A+D,B-D), where D = ABS(B-A)/100.

      TYPE (FM) :: A,B,XMAX,FMAX
      TYPE (FM), EXTERNAL :: F
      INTEGER :: N,NUM_F,KPRT,NW
      LOGICAL :: SPIKE_FOUND
      TYPE (FM), SAVE :: AB,AVERAGE,D,DX,FPMAX,EPS,F1,F2,H,T,X1,XPMAX
      INTEGER :: J

      CALL FM_ENTER_USER_ROUTINE
      SPIKE_FOUND = .FALSE.
      IF (KPRT >= 2) THEN
          WRITE (NW,"(A)") ' Enter FM_SPIKE'
      ENDIF

      DX = ABS(B-A)/100
      H = MIN(XMAX-A,B-XMAX)/2

      AVERAGE = 0
      DO J = 1, 99
         X1 = A + (J*(B-A))/100
         F1 = F(X1,N)
         NUM_F = NUM_F + 1
         IF (MWK(START(F1%MFM)+2) == MUNKNO) THEN
             XMAX = X1
             FMAX = ABS(F1)
             SPIKE_FOUND = .TRUE.
             CALL FM_EXIT_USER_ROUTINE
             RETURN
         ELSE
             AVERAGE = AVERAGE + ABS(F1)
         ENDIF
      ENDDO
      AVERAGE = AVERAGE/99

!             Search for a singularity in F.

      DO
         X1 = XMAX + H
         IF (X1 > A + DX .AND. X1 < B - DX) THEN
             F1 = F(X1,N)
             NUM_F = NUM_F + 1
             IF (MWK(START(F1%MFM)+2) == MUNKNO) THEN
                 XMAX = X1
                 FMAX = ABS(F1)
                 SPIKE_FOUND = .TRUE.
                 CALL FM_EXIT_USER_ROUTINE
                 RETURN
             ENDIF
             IF (ABS(F1) > FMAX) THEN
```

```
                XMAX = X1
                FMAX = ABS(F1)
                H = -(H*14)/10
            ENDIF
        ENDIF
        H = -(H*10)/14
        AB = MAX(ABS(A),ABS(B))
        CALL FM_ULP(AB,EPS)
        IF (ABS(H) < ABS(EPS)) THEN
            IF (FMAX > 50*AVERAGE) THEN
                SPIKE_FOUND = .TRUE.
                CALL FM_EXIT_USER_ROUTINE
                RETURN
            ELSE
                EXIT
            ENDIF
        ENDIF
    ENDDO

!             Search for a singularity in F'.

    FPMAX = -1
    AVERAGE = 0
    AB = MAX(ABS(A),ABS(B))
    CALL FM_ULP(AB,EPS)
    EPS = SQRT(EPS)
    DO J = 1, 99
        X1 = A + (J*(B-A))/100
        IF (X1-EPS <= A) CYCLE
        F1 = F(X1-EPS,N)
        IF (X1+EPS >= B) CYCLE
        F2 = F(X1+EPS,N)
        NUM_F = NUM_F + 2
        IF (MWK(START(F1%MFM)+2) == MUNKNO .OR. MWK(START(F2%MFM)+2) == MUNKNO) THEN
            IF (MWK(START(F1%MFM)+2) == MUNKNO) THEN
                XMAX = X1 - EPS
                FMAX = ABS(F1)
            ELSE
                XMAX = X1 + EPS
                FMAX = ABS(F2)
            ENDIF
            SPIKE_FOUND = .TRUE.
            CALL FM_EXIT_USER_ROUTINE
            RETURN
        ELSE
            D = ABS((F2-F1)/(2*EPS))
            AVERAGE = AVERAGE + D
            IF (D > FPMAX) THEN
                XPMAX = X1
                FPMAX = D
            ENDIF
        ENDIF
    ENDDO
    AVERAGE = AVERAGE/99

    H = MIN(XPMAX-A,B-XPMAX)/2
```

```
DO
   X1 = XPMAX + H
   IF (X1 > A + DX .AND. X1 < B - DX) THEN
       F1 = F(X1-EPS,N)
       NUM_F = NUM_F + 1
       IF (MWK(START(F1%MFM)+2) == MUNKNO) THEN
           XPMAX = X1
           FPMAX = ABS(F1)
           SPIKE_FOUND = .TRUE.
           CALL FM_EXIT_USER_ROUTINE
           RETURN
       ENDIF
       F2 = F(X1+EPS,N)
       NUM_F = NUM_F + 1
       IF (MWK(START(F2%MFM)+2) == MUNKNO) THEN
           XPMAX = X1+EPS
           FPMAX = ABS(F2)
           SPIKE_FOUND = .TRUE.
           CALL FM_EXIT_USER_ROUTINE
           RETURN
       ENDIF
       D = ABS((F2-F1)/(2*EPS))
       IF (D > FPMAX) THEN
           XPMAX = X1
           FPMAX = D
           H = -(H*14)/10
       ENDIF
   ENDIF
   H = -(H*10)/14
   AB = MAX(ABS(A),ABS(B))
   CALL FM_ULP(AB,T)
   IF (ABS(H) < ABS(T)) THEN
       IF (FPMAX > 50*AVERAGE) THEN
           SPIKE_FOUND = .TRUE.
           XMAX = XPMAX
           FMAX = FPMAX
           CALL FM_EXIT_USER_ROUTINE
           RETURN
       ELSE
           EXIT
       ENDIF
   ENDIF
ENDDO

CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_SPIKE
```