

```

PROGRAM OSC
USE FMZM
IMPLICIT NONE

! If the integrand is highly (or infinitely) oscillatory, standard numerical integration methods
! often take too long when used directly.

! In this program, we indirectly integrate 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 we use an extrapolation method 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.

! To compute this integral to 50 significant digits, use 50 for the precision and 70 for the
! number of roots.

TYPE (FM), SAVE :: B, C, D, PI, R1, R2, RESULTS(100), S, TERMS(100), TOL
INTEGER :: J, K, KPRT, KW, N, NAIT, NROOTS
CHARACTER(80) :: ST1
TYPE (FM), EXTERNAL :: F

KPRT = 1
KW = 12
OPEN(12,FILE='OscillateFM.out')
CALL FM_SETVAR(' KW = 12 ')

N = 50
CALL FMSET(N)
NROOTS = 70

!           Integrate between pairs of roots.

WRITE (*,"(A)") ' '
WRITE (*,"(A)") ' Integrals between roots:'
WRITE (12,"(A)") ' '
WRITE (12,"(A)") ' Integrals between roots:'

CALL FM_PI(PI)
TOL = TO_FM(10)**(-N)

DO J = 1, NROOTS
  KPRT = 1
  IF (J == 1) THEN
    R1 = 1/PI
    R2 = 1
  ELSE
    R1 = 1/(J*PI)
    R2 = 1/((J-1)*PI)
  ENDIF
  CALL FM_INTEGRATE(F,1,R1,R2,TOL,RESULTS(J),KPRT,KW)

```

```

    IF (MOD(J,10) == 0) WRITE (*,"(A,I4)") ' J = ',J
ENDDO

```

! Form the sequence of partial sums.

```

WRITE (12,"(A)") ' '
WRITE (12,"(A)") ' Partial sums:'
TERMS(1) = RESULTS(1)
DO J = 2, NROOTS
    TERMS(J) = RESULTS(J) + TERMS(J-1)
    CALL FM_FORM('F56.50',TERMS(J),ST1)
    WRITE (12,"(7X,A)") ST1
ENDDO

```

! Use Aitken extrapolation on the sequence of partial sums.

```

K = NROOTS

```

```

WRITE (12,"(A)") ' '
WRITE (12,"(A)") ' Aitken extrapolation of the partial sums:'
KPRT = 0
R1 = ABS(TERMS(K) - TERMS(K-1))
DO J = 3, NROOTS, 2
    CALL AITKEN(K,TERMS,KPRT,KW)
    K = K - 2
    R2 = ABS(TERMS(K) - TERMS(K-1))
    CALL FM_FORM('ES12.4',R2,ST1)
    WRITE (12,"(I4,A,A)") J/2,' extrapolations. Estimated error =',TRIM(ST1)
    IF (R2 > R1 .OR. J >= NROOTS-1) THEN
        WRITE (12,"(A)") ' '
        WRITE (12,"(A,I4,A)") ' The last two estimates after ',J/2-1,' Aitken extrapolations ='
        WRITE (*,"(A)") ' '
        WRITE (*,"(A,I4,A)") ' The last two estimates after ',J/2-1,' Aitken extrapolations ='
        CALL FM_FORM('F56.50',TERMS(K+1),ST1)
        WRITE (12,"(7X,A)") ST1
        WRITE (*,"(7X,A)") ST1
        CALL FM_FORM('F56.50',TERMS(K+2),ST1)
        WRITE (12,"(7X,A)") ST1
        WRITE (*,"(7X,A)") ST1
        NAIT = K + 2
        EXIT
    ENDIF
R1 = R2
ENDDO

```

! Compare Cohen's alternating series extrapolation method.

! This method applies to alternating series where the first term is positive and the
! sequence of partial sums $a(k)$ is totally monotonic. This means that for each fixed k ,
! the sequence of the k -th forward differences of $a(k)$ consists of all positive values
! or all negative values. Negate the result when the first term is negative.

```

WRITE (*,"(A)") ' '
WRITE (*,"(A)") " Cohen's alternating series extrapolation method:"
WRITE (*,"(A)") ' '
WRITE (12,"(A)") ' '

```

```

WRITE (12,"(A)") " Cohen's alternating series extrapolation method:"
WRITE (12,"(A)") ' '
DO N = NROOTS-1, NROOTS
  D = (3 + SQRT(TO_FM(8)))*N
  D = (D + 1/D)/2
  B = -1
  C = -D
  S = 0
  DO K = 0, N-1
    C = B - C
    S = S + C*ABS(RESULTS(K+1))
    B = (K+N)*(K-N)*B / ((K+TO_FM('0.5'))*(K+1))
  ENDDO
  S = S/D
  WRITE (12,"(1X,A,I2,A)") ' N = ',N,'.      Extrapolated value ='
  IF (RESULTS(1) < 0) S = -S
  CALL FM_FORM('F56.50',S,ST1)
  WRITE (12,"(7X,A)") ST1
  WRITE ( *, "(1X,A,I2,A)") ' N = ',N,'.      Extrapolated value ='
  WRITE ( *, "(7X,A)") ST1
ENDDO

```

! For this example problem, there is a closed-form answer in terms
! of the cosine integral and the sine. Print it as a check.

```

R1 = SIN(TO_FM(1)) - COS_INTEGRAL(TO_FM(1))
WRITE (12,"(A)") ' '
WRITE (12,"(A)") ' For this example problem, there is a closed-form answer: Sin(1) - Ci(1) ='
CALL FM_FORM('F56.50',R1,ST1)
WRITE (12,"(7X,A)") ST1
WRITE ( *, "(A)") ' '
WRITE ( *, "(A)") ' For this example problem, there is a closed-form answer: Sin(1) - Ci(1) ='
WRITE ( *, "(7X,A)") ST1
WRITE ( *, "(A)") ' '
WRITE ( *, "(A)") ' Intermediate results from this calculation are in file Oscillate.out'
WRITE ( *, "(A)") ' '

```

END PROGRAM OSC

```

SUBROUTINE AITKEN(K,RESULTS,KPRT,KW)
USE FMZM
IMPLICIT NONE

```

! Aitken extrapolation.
! Extrapolate RESULTS(1), ..., RESULTS(K). The Aitken values are returned in
! RESULTS(1), ..., RESULTS(K-2)
! KPRT = 1 means write the new values in RESULTS on unit KW
! = 0 means no output is written.

```

TYPE (FM) :: RESULTS(100)
INTEGER :: J, K, KPRT, KW

```

```

CALL FM_ENTER_USER_ROUTINE
IF (KPRT == 1) THEN
  WRITE (KW,"(A)") ' '
  WRITE (KW,"(A)") ' Aitken extrapolation.'

```

ENDIF

DO J = 1, K-2

IF (RESULTS(J+2) - 2*RESULTS(J+1) + RESULTS(J) == 0) THEN

RESULTS(J) = RESULTS(J+2)

ELSE

RESULTS(J) = RESULTS(J+2) - (RESULTS(J+2)-RESULTS(J+1))**2 / &
(RESULTS(J+2) - 2*RESULTS(J+1) + RESULTS(J))

ENDIF

IF (KPRT == 1) THEN

CALL FM_PRINT(RESULTS(J))

ENDIF

ENDDO

CALL FM_EXIT_USER_ROUTINE

END SUBROUTINE AITKEN

FUNCTION F(X,N)

USE FMZM

IMPLICIT NONE

TYPE (FM) :: F, X

INTEGER :: N

CALL FM_ENTER_USER_FUNCTION(F)

F = X

IF (N == 1) THEN

F = SIN(1/X)

ENDIF

CALL FM_EXIT_USER_FUNCTION(F)

END FUNCTION F