```
      program test
      use fmzm
      implicit none

!  Least squares fit for the coefficients in the asymptotic series for the j-th harmonic number.

!  H(j) = 1 + 1/2 + 1/3 + ... + 1/j  defines the j-th harmonic number.

!  Find an approximation to H(j) of the form:

!  ln(j) + c(1) + c(2)/j + ... + c(k)/j**(k-1)

!  Integrating 1/x from 1 to j gives ln(j) as a first approximation, and we generate n data
!  points (x(i), y(i)) where x(i) is j and y(i) is H(j) for various j values.  Then we do a
!  least squares fit of the model function c(1) + c(2)/j + ... + c(k)/j**(k-1) to the data
!  (x(i), y(i)-ln(i)).

!  Since this is a sample problem, we can compare the results of the fit to the "true"
!  asymptotic formula, where c(1) = 0.57721566..., Euler's constant, and for i > 1,
!  c(i) = -B(i-1)/(i-1).  The B values are Bernoulli numbers, and the first few are:
!  B(1) = -1/2, B(2) = 1/6, B(4) = -1/30, B(6) = 1/42, ..., with the others being zero:
!  B(3) = B(5) = B(7) = ... = 0.

!  The first c's in the list of fitted coefficients give the most agreement with the
!  theoretical values, and the last ones the least.  The linear system is ill-conditioned,
!  but by using high precision we can get good accuracy for several coefficients.
!  For example, using 400 digit precision, 60 data points at intervals of 100 (i.e.,
!  x(i) = 100, 200, 300, ..., 6000), and fitting 60 coefficients, we get at least 50
!  decimal agreement between the fitted c's and the theoretical ones for c(1), ..., c(29).
!  c(41) agrees to 16 decimals, and because the number is large this is 31 significant
!  digit agreement.

      integer :: j, k, n, ngap
      type (fm) :: h_n, one, det
      type (fm), allocatable :: a(:,:), b(:), c(:), x(:), y(:)
      type (fm), external :: f

!             This is not a good way to compute Euler's constant, but with 150 digit precision,
!             n = 40 data points at intervals of ngap = 10, fitting k = 40 coefficients we get
!             c(1) = .57721566490153286060651209008240243104215933593992,
!             correct to 50 places.

!             Set FM precision.

      call fm_set(150)

!             n is the number of harmonic data points.

      n = 40

!             ngap is the gap between harmonic data points.

      ngap = 10

!             k is the number of coefficients to fit.

      k = 40
```

```fortran
      allocate(a(k, k), b(k), c(k), x(n), y(n), stat=j)
      if (j /= 0) then
          write (*, "(/' Error in hfit.  Unable to allocate arrays with k, n = ', 2i8/)") k, n
          stop
      endif

!             Generate the harmonic data points.
!             Since the coefficient of the first term in the model, ln(x), is assumed
!             to be 1 and is not being fitted, subtract that from the y data points.

      h_n = 0
      one = 1
      write (*,*) ' '
      write (*,*) ' Data points:'
      write (*,*) ' '
      do j = 1, n*ngap
         h_n = h_n + one/j
         if (mod(j, ngap) == 0) then
             x(j/ngap) = j
             y(j/ngap) = h_n - log(x(j/ngap))
             write (*, "(a, i4, a, i6, a, a)") ' i = ', j/ngap, '   x = ', j, '   y = ',  &
                                            trim(fm_format('f40.35', y(j/ngap)))
         endif
      enddo

!             Generate the linear system for the normal equations.

      call fm_geneq(f, a, b, k, x, y, n)

!             Solve the linear system for the normal equations.

      call fm_lin_solve(a, c, b, n, det)

!             Print the solution.
!             When using f format, FM doesn't like to print 0.00000...0 showing no
!             significant digits when the actual number is too small for that format.
!             FM will shift to e format when possible, to avoid showing all zeroes.
!             In this example, all the even-numbered coefficients are zero in the
!             asymptotic series for the harmonic numbers, so any non-zero digits
!             found in the fit are not interesting.  Therefore the if statement
!             below prints exactly zero when c(j) is too small, making the output
!             look neater.

      write (*,*) ' '
      write (*,*) ' Fitted coefficients:'
      do j = 1, k
         if (abs(c(j)) > 1.0d-50) then
             write (*, "(a, i3, a, a)") ' j = ', j, ' c(j) = ', trim(fm_format('f60.50', c(j)))
         else
             write (*, "(a, i3, a, a)") ' j = ', j, ' c(j) = ', trim(fm_format('f60.50', to_fm(0)))
         endif
      enddo

      end program test

      function f(j, x)     result (return_value)
      use fmzm
```

```
      implicit none

!  This defines the model function being fitted to the data points.
!  For the harmonic number case, the model function is:

!  f(j, x) = 1/x**(j-1)

!  This will fit the terms  c1 + c2/n + c3/n**2 + ... to the harmonic model function
!  ln(x) + c1 + c2/n + c3/n**2 + ....

      integer :: j
      type (fm) :: return_value, x
      intent (in) :: j, x

      return_value = 1/x**(j-1)

      end function f
```