

```
! This program finds the eigenvalues and eigenvectors for a random 20x20 real matrix.
! The peculiar algorithm used is not a very good way to do this, but is a test for several
! of the FM sample subroutines.
```

```
! Some trace output is written to the screen as the program runs, and the eigenvalues and
! eigenvectors are written to file EigenSystem.out at the end.
```

```
MODULE EIGEN_VARIABLES
USE FMZM
```

```
! COEFFS holds the coefficients for the characteristic polynomial of a matrix.
! N_COEFFS is the number of coefficients, and NDIG_OF_COEFFS is their precision.
```

```
TYPE (FM), ALLOCATABLE :: COEFFS(:)
INTEGER, SAVE :: N_COEFFS, NDIG_OF_COEFFS
```

```
END MODULE EIGEN_VARIABLES
```

```
PROGRAM ETEST
USE FMZM
IMPLICIT NONE
```

```
TYPE (FM), ALLOCATABLE :: A(:, :)
TYPE (ZM), ALLOCATABLE :: EIGENVECTORS(:, :), EIGENVALUES(:)
TYPE (FM) :: ERR
INTEGER :: SEED(7)
INTEGER :: J, K, KU, KPRT, N
DOUBLE PRECISION :: XVAL
REAL :: TIME1, TIME2
```

```
N = 20
K = 40
KPRT = 1
KU = 6
CALL FM_SET(K)
```

```
ALLOCATE(A(N,N), EIGENVECTORS(N,N), EIGENVALUES(N), STAT=J)
IF (J /= 0) STOP
CALL FM_SETVAR(" KSWIDE = 110 ")
```

```
!           Generate an NxN matrix of random integers between -100 and +100.
```

```
CALL FM_RANDOM_SEED_SIZE(J)
SEED(1:J) = 13
SEED(1) = N
SEED(2) = K
CALL FM_RANDOM_SEED_PUT(SEED(1:J))
DO J = 1, N
  DO K = 1, N
    CALL FM_RANDOM_NUMBER(XVAL)
    A(J,K) = INT(200*XVAL - 100)
  ENDDO
ENDDO
```

! Find the Eigenvalues and Eigenvectors.

```
CALL CPU_TIME(TIME1)
```

```
CALL FM_EIGENSYSTEM(A,N,EIGENVECTORS,EIGENVALUES,KPRT,KU)
```

```
CALL CPU_TIME(TIME2)
```

! Write the solution.

```
OPEN(22,FILE='EigenSystem.out')
```

```
CALL FM_SETVAR(" KW = 22 ")
```

```
DO J = 1, N
```

```
  WRITE (22,*) ' '
```

```
  WRITE (22,"(A,I3,A)") ' Eigenvalue ',J,':'
```

```
  CALL ZM_PRINT(EIGENVALUES(J))
```

```
  WRITE (22,*) ' '
```

```
  WRITE (22,"(A,I3,A)") ' Eigenvector ',J,':'
```

```
  DO K = 1, N
```

```
    CALL ZM_PRINT(EIGENVECTORS(K,J))
```

```
  ENDDO
```

```
ENDDO
```

```
WRITE (22,*) ' '
```

```
WRITE (22,*) ' '
```

! Find the norm of the residual, $\|A X - c X\|$ for each eigenvalue c and eigenvector X .

```
ERR = 0
```

```
DO J = 1, N
```

```
  ERR = ERR + SUM(ABS(MATMUL(TO_ZM(A(1:N,1:N)),EIGENVECTORS(1:N,J)) - &  
                  EIGENVALUES(J)*EIGENVECTORS(1:N,J)))
```

```
ENDDO
```

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

```
WRITE (*,"(A,ES16.7)") ' Norm of the residual = ',TO_DP(ERR/N**2)
```

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

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

```
WRITE (*,"(A,F9.2,A)") ' Time to solve the system: ',TIME2-TIME1,' seconds.'
```

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

```
WRITE (*,*) ' The eigenvalues and eigenvectors are in file EigenSystem.out.'
```

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

```
STOP
```

```
END PROGRAM ETEST
```

```
SUBROUTINE FM_EIGENSYSTEM(A,N,EIGENVECTORS,EIGENVALUES,KPRT,KU)
```

```
USE FMVALS
```

```
USE FMZM
```

```
USE EIGEN_VARIABLES
```

```
IMPLICIT NONE
```

! Find the eigensystem for real $N \times N$ matrix A .

! EIGENVALUES(1:N) is returned with the (complex) eigenvalues.

```

! EIGENVECTORS(1:N,J) is returned with the (complex) eigenvector corresponding to EIGENVALUES(J),
!           for J = 1, 2, ..., N.

! A is a type (fm) multiprecision real array,
! EIGENVALUES and EIGENVECTORS are type (zm) multiprecision complex arrays.

! KPRT controls printing within the routine:
!       KPRT = 0 for no output
!       KPRT = 1 for some trace output.

! KU is the unit number for output.

! This routine does not handle cases with eigenvalues of multiplicity greater than 1, since
! some of those cases do not have a full set of N eigenvectors.

! The algorithm used here is not the best way to compute eigensystems. It is not even a good
! way to do it. But it was easy to code using existing FM routines for finding determinants
! and all the roots of a polynomial, and it has a certain Rube Goldberg charm.

! Algorithm:  1. Generate the N+1 matrices A - lambda*I for lambda = 1, 2, ..., N+1
!            2. Use FM_FACTOR_LU to find the N+1 determinants for these matrices
!            3. Use FM_LIN_SOLVE to find the N+1 coefficients of the Nth degree characteristic
!               polynomial for A, using the points on the curve found in step 2.
!            4. Use ZM_ROOTS to find the N (complex) roots of this polynomial.
!               These roots are the eigenvalues.
!            5. For each eigenvalue, do a few iterations of the inverse power method to find
!               the corresponding eigenvector.

INTEGER :: I, J, JMAX, K, KPRT, KU, KWARN_SAVE, LAMBDA, N, NDSAVE, N_FOUND
TYPE (FM) :: A(N,N)
TYPE (ZM) :: EIGENVALUES(N), EIGENVECTORS(N,N)
TYPE (FM), SAVE :: DET, P, TOL
TYPE (FM), ALLOCATABLE :: A1(:,,:), A2(:,,:), EQN(:,,:), IDENTITY(:,,:), RHS(:)
TYPE (ZM), ALLOCATABLE :: Z1(:,,:), Z2(:,,:), ZX(:), ZT(:), LIST_OF_ROOTS(:)
TYPE (ZM), SAVE :: T, ZDET
INTEGER, ALLOCATABLE :: KSWAP(:)
TYPE (ZM), EXTERNAL :: EIGEN_POLY

CALL FM_ENTER_USER_ROUTINE

!           Raise precision slightly.

NDSAVE = NDIG
NDIG = NDIG + NGRD52
KWARN_SAVE = KWARN
KWARN = 0

ALLOCATE(A1(N,N),A2(N,N),EQN(N+1,N+1),IDENTITY(N,N),Z1(N,N),Z2(N,N),ZX(N),ZT(N),RHS(N+1), &
        COEFFS(N+1),LIST_OF_ROOTS(N),KSWAP(N),STAT=J)
IF (J /= 0) THEN
    WRITE (KU, "(/' Error in FM_EIGENSYSTEM. Unable to allocate arrays with N = ',I8/)" N
    STOP
ENDIF

!           Copy A to A1 with higher precision, and generate an identity matrix.

```

```

110 TOL = TO_FM(MBASE)**(-NDSAVE)
IDENTITY = 0
DO I = 1, N
  DO J = 1, N
    CALL FM_EQU(A(I,J),A1(I,J),NDSAVE,NDIG)
  ENDDO
  IDENTITY(I,I) = 1
ENDDO

!           Generate the N+1 (real) N x N matrices A - lambda*I for lambda = 1, 2, ..., N+1,
!           and find the N+1 determinants for these matrices

DO LAMBDA = 1, N+1
  A2 = A1 - LAMBDA*IDENTITY
  CALL FM_FACTOR_LU(A2,N,DET,KSWAP)
  P = 1
  DO J = N+1, 1, -1
    EQN(LAMBDA,J) = P
    P = LAMBDA*P
  ENDDO
  RHS(LAMBDA) = DET
ENDDO
N_COEFFS = N + 1

!           Find the (real) coefficients of the characteristic polynomial for A.

CALL FM_LIN_SOLVE(EQN,COEFFS,RHS,N_COEFFS,DET)
NDIG_OF_COEFFS = NDIG
IF (DET == 0) THEN
  WRITE (KU,"(/' Error in FM_EIGENSYSTEM. Zero determinant -- no unique ','// &
    "'characteristic polynomial.'/)")
  STOP
ENDIF

!           Use ZM_ROOTS to find all the (complex) roots of this polynomial.
!           These roots (in LIST_OF_ROOTS) are the eigenvalues.

CALL ZM_ROOTS(N,EIGEN_POLY,1,N_FOUND,LIST_OF_ROOTS,KPRT,KU)

IF (N_FOUND /= N) THEN
  IF (KPRT > 0) THEN
    WRITE (KU,"(/' In FM_EIGENSYSTEM, N_FOUND /= N. N = ','// &
      "'I3,'. N_FOUND =',I3/)" N,N_FOUND
    WRITE (KU,"(/' Increase precision and try again.'/)")
  ENDF
  NDIG = 2*NDIG
  GO TO 110
ENDIF

!           Sort the eigenvalues so they have decreasing magnitude, and check that there were
!           no multiple roots.

DO I = 2, N
  JMAX = I-1
  DO J = I, N

```



```

        WRITE (KU,"(/' Increase precision and try again.'/)")
    ENDF
    NDIG = 2*NDIG
    GO TO 110
ENDIF
ENDDO
ENDDO

CALL FM_DEALLOCATE(A1)
CALL FM_DEALLOCATE(A2)
CALL FM_DEALLOCATE(EQN)
CALL FM_DEALLOCATE(IDENTITY)
CALL FM_DEALLOCATE(Z1)
CALL FM_DEALLOCATE(Z2)
CALL FM_DEALLOCATE(ZX)
CALL FM_DEALLOCATE(ZT)
CALL FM_DEALLOCATE(RHS)
CALL FM_DEALLOCATE(COEFFS)
CALL FM_DEALLOCATE(LIST_OF_ROOTS)
DEALLOCATE(A1,A2,EQN,IDENTITY,Z1,Z2,ZX,ZT,RHS,COEFFS,LIST_OF_ROOTS,KSWAP)

NDIG = NDSAVE
KWARN = KWARN_SAVE
CALL FM_EXIT_USER_ROUTINE
END SUBROUTINE FM_EIGENSYSTEM

FUNCTION EIGEN_POLY(X,NF)
USE EIGEN_VARIABLES
USE FMVALS
USE FMZM
IMPLICIT NONE

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

INTEGER :: J, NF
TYPE (ZM) :: EIGEN_POLY, X
TYPE (FM), SAVE :: D

CALL FM_ENTER_USER_FUNCTION(EIGEN_POLY)
IF (NF == 1) THEN
    CALL FM_EQU(COEFFS(1),D,NDIG_OF_COEFFS,NDIG)
    EIGEN_POLY = D
    DO J = 2, N_COEFFS
        CALL FM_EQU(COEFFS(J),D,NDIG_OF_COEFFS,NDIG)
        EIGEN_POLY = EIGEN_POLY*X + D
    ENDDO
ELSE
    EIGEN_POLY = 3*X - 2
ENDIF

CALL FM_EXIT_USER_FUNCTION(EIGEN_POLY)
END FUNCTION EIGEN_POLY

```