```
      PROGRAM HALO96
c     this is the driver program to compute the planar and  halo orbits
c     near L2 (depending upon which subroutines are selected,      i.e.
c     their call statements are uncommented. For the planar family,  we
c     iterate with X1FIX and continue the family with NIC15;   to start
c     the halo branch, X3FIX and NIC315 are used;  NIC135 comes later.)
c     Cartesian position & velocity co-ords are numerically  integrated
c     using simo's edo (rk78).
c     It must be linked with rg, decomp, spline and edo2 (my version).
      REAL*8      BABYX(6), X(42), XINIT(6), XD(6), A(6,6),Z(6),JCONST
      REAL*8      MU, MEARTH, MMOON, EARTHX, MOONX, TCROSS, XMAX
      REAL*8      X1SAVE, X5SAVE, INCR, LAGPT,DL, DELTA, DOLD,EPSLON
      REAL*8      R1, R2, D1F, D3F, D4F, D6F, YDINV, DQ(4,4), COND
      INTEGER*4   I,J,ECOUNT,ITNUMB,L,N
c     ------------------------------------------------variables for edo
      REAL*8     T,TOUT,H,HMIN,HMAX,RELERR, ABSERR, WORK(630)
      INTEGER*4  IOPT,NEQN2, NMAX,IFLAG, IWORK(1)
c     ------------------------------------------------variables for rg
      REAL*8      WR(4),  WI(4),  ZZ(4,4),  FV1(4)
      INTEGER     NM, MATZ, IERR, IV1(4)
c     ------------------------------------------------variables for stabco
      REAL*8      BROUK1, BROUK2
      CHARACTER*3   SYMM
      EXTERNAL      DIFEQ1, DIFEQ2
      COMMON        MEARTH, MMOON, EARTHX, MOONX
      COMMON/CART/ XD
      OPEN( 8, FILE = 'h7.dat', STATUS= 'NEW')
      OPEN( 9, FILE = 'sc.dat', STATUS= 'NEW')
c     -------------value of mu from Szebehely: Theory Of Orbits p. 217
      MU        = 0.0121506683D0
      LAGPT     = 1.1556824834D0
      MMOON     = MU
      MEARTH    = 1.0D0 - MU
      EARTHX    = -MU
      MOONX     = 1.0D0 - MU
      EPSLON    = 1.0D-09
      INCR      = 4.0D-04
c     ---------------------------------------------initialization values
c                                                        small orbits at L2
c      TCROSS    =  0.3373262771787D+01
c      XINIT(1)  =   .1155375630696D+01
c      XINIT(2)  =   .0000000000000D+00
c      XINIT(3)  =   .0000000000000D+00
c      XINIT(4)  =   .2974057844548D-09
c      XINIT(5)  =   .1662874268202D-02
c      XINIT(6)  =   .0000000000000D+00
c      X1SAVE    =   .1155390861102D+01
c      X5SAVE    =   .1580425348516D-02
c     ------------------------------------------just before halo branch
      TCROSS    = .3415093389921411D+01
      XINIT(1)  =  0.11205756306960D+01
      XINIT(2)  =  0.00000000000000D+00
      XINIT(3)  =  0.00000000000000D+00
      XINIT(4)  =  0.15746589336473D-10
      XINIT(5)  =  0.17514264829992D+00
      XINIT(6)  =  0.00000000000000D+00
c     ---------------------------------just a little way on halo branch
c       TCROSS    = .3414947109879041D+01
c       XINIT(1)  =  0.11201154199175D+01
c       XINIT(2)  =  0.00000000000000D+00
c       XINIT(3)  = -0.61234293999200D-02
c       XINIT(4)  =  0.59717457769464D-13
c       XINIT(5)  =  0.17682843077045D+00
c       XINIT(6)  =  0.30156586643431D-12
c     ------------------------------Big Loop: Number of orbits to find
      DO 200 ECOUNT = 1,10
```

```
        IF ( MOD(ECOUNT,3) .EQ. 1 ) THEN
          WRITE(*,'(1X,''Orbit Number '',I4,10X,''Iterations to close '',
     &          ''last orbit '',I6)') ECOUNT, ITNUMB
          CALL MILES( XINIT, LAGPT )
        ENDIF
        CALL JACOBI( XINIT, JCONST, R1, R2)
        ITNUMB      = 0
        DO 9 L      = 1,6
          BABYX(L) = XINIT(L)
    9   CONTINUE
C       ------------------------------ this block is repeated while we
C                                                       refine a fixed orbit
    5   CALL CTIME( BABYX, TCROSS, ECOUNT )
        CALL INIT(   T, XINIT, X  )
C       ---------------------------------------- R.K.F.7-8 integration
        IOPT      = 1
        NEQN2     = 42
C       --------------------i'm guessing on what to set these
        H         = 1.0D-05
        HMIN      = 1.0D-06
        HMAX      = 1.0D0
        NMAX      = 10000
        RELERR    = 1.0D-14
        ABSERR    = 1.0D-14
        IFLAG     = 4
        TOUT      = T
        CALL EDO( DIFEQ2,IOPT, NEQN2, X,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
        TOUT      = TCROSS
        CALL EDO( DIFEQ2,IOPT, NEQN2, X,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
C       ----------------------------------form the matrix A = D_{2}\phi
        DO 55 I     = 1,6
         DO 50 J    = 1,6
            A(I,J) = X(6*J + I)
   50    CONTINUE
   55   CONTINUE
        ITNUMB      = ITNUMB + 1
C       --------------------------------check to see if orbit is closed
        DL       = 0.0D0
        DO 33 L = 1,6
          DL     = DL + (XINIT(L) - X(L))**2
   33   CONTINUE
        DOLD     = DELTA
        DELTA    = DSQRT( DL)
        IF (DELTA .GT. EPSLON) THEN
C       ------------------------------------------------refine ICs block
         IF (ITNUMB .GE. 15) THEN
          WRITE(*, '(1x, '' ORBIT NOT CLOSING'')')
          STOP
         ENDIF
C        ---------------------------------------choose component to hold
C                                                fixed in newton iteration
C         CALL X1FIX( A, XINIT, X, Z, COND)
         CALL X3FIX( A, XINIT, X, Z, COND)
C        ---------------------------------------set new initial conditions
         DO 95 L       = 1,6
            XINIT(L)   = Z(L)
            BABYX(L)   = Z(L)
   95    CONTINUE
         GOTO 5
        ENDIF
C       ---------------------------------------------now orbit is closed
        IF ( MOD( ECOUNT, 5) .EQ. 0 ) THEN
C        ---------------------------------------form the matrix DQ
         CALL JACOBI(XINIT,JCONST, R1, R2)
```

```
      YDINV = 1.0D0 / XINIT(5)
      D1F   = YDINV*( XINIT(1) - (1.0D0-MU)*(XINIT(1)-EARTHX)/R1**3 -
     &                                    MU *(XINIT(1)-MOONX )/R2**3 )
      D3F   = -YDINV * XINIT(3)*( (1.0D0-MU)/R1**3  + MU/R2**3 )
      D4F   = -YDINV * XINIT(4)
      D6F   = -YDINV * XINIT(6)
      DO 160 i = 1,4
       IF (i .EQ. 1)  L=1
       IF (i .EQ. 2)  L=3
       IF (i .EQ. 3)  L=4
       IF (i .EQ. 4)  L=6
       DQ(i,1) = A(L,1) + A(L,5)*D1F
       DQ(i,2) = A(L,3) + A(L,5)*D3F
       DQ(i,3) = A(L,4) + A(L,5)*D4F
       DQ(i,4) = A(L,6) + A(L,5)*D6F
160   CONTINUE
      DQ(3,1) = DQ(3,1) - (XD(4)/XD(2))*( A(2,1) + A(2,5)*D1F )
      DQ(3,2) = DQ(3,2) - (XD(4)/XD(2))*( A(2,3) + A(2,5)*D3F )
      DQ(3,3) = DQ(3,3) - (XD(4)/XD(2))*( A(2,4) + A(2,5)*D4F )
      DQ(3,4) = DQ(3,4) - (XD(4)/XD(2))*( A(2,6) + A(2,5)*D6F )
C
      DQ(4,1) = DQ(4,1) - (XD(6)/XD(2))*( A(2,1) + A(2,5)*D1F )
      DQ(4,2) = DQ(4,2) - (XD(6)/XD(2))*( A(2,3) + A(2,5)*D3F )
      DQ(4,3) = DQ(4,3) - (XD(6)/XD(2))*( A(2,4) + A(2,5)*D4F )
      DQ(4,4) = DQ(4,4) - (XD(6)/XD(2))*( A(2,6) + A(2,5)*D6F )
C     ----------------------------------------compute EWs & EVs of DQ
      N     = 4
      NM    = 4
      MATZ = 0
      CALL RG(NM, N, DQ, WR,WI, MATZ, ZZ, IV1, FV1, IERR)
C     ----------------------write trajectory to data file for graph
C      CALL PLOTTO( XINIT, TCROSS)
C     ------------------------------check if orbit is L246 symmetric
C      CALL SYMMET( XINIT, TCROSS, SYMM, XMAX)
C     ---------------------------------------write output to data file
      WRITE(8,'(1x,''# = '', I2, 1x,''JC = '', D20.14,''  TCROSS= '',
     &            D21.16, '' SYMM='',A3 )') ITNUMB,JCONST, TCROSS,SYMM
      WRITE(8, '(1X, '' true dist to Moon = '',F9.2, '' km    '', 1X,
     &  ''XMAX='',f10.0,''km     cond DG ='',d10.3 )')
     &  3.84551D05*DSQRT( (XINIT(1)-MOONX)**2 + XINIT(3)**2 ),
     &  3.84551D05*(XMAX-MOONX), COND
      DO 110 i = 1,4
       WRITE(8, '(1x, ''XINIT('',I1,'') = '', D21.14, 4X, ''W('', I1,
     &  '') = '', F16.10, F16.10,''i'')') I, XINIT(I), I, WR(I), WI(I)
110   CONTINUE
      WRITE(8, '(1x, ''XINIT(5) = '', d21.14)') XINIT(5)
      WRITE(8, '(1x, ''XINIT(6) = '', d21.14)') XINIT(6)
      WRITE(8, '(1x, '' '')')
C     ------------------------calculate Dr. Brouke's stability coeffs
      CALL STABCO( WR, WI, BROUK1, BROUK2)
      WRITE( 9, '(1x, f10.3, 2x, f10.3)') BROUK1, BROUK2
     ENDIF
C     ---------------------------------------compute guess for new ICs
C      CALL NIC15(  ECOUNT, XINIT, X1SAVE, X5SAVE, INCR )
     CALL NIC315( ECOUNT, XINIT, INCR )
C     CALL NIC135( ECOUNT, XINIT, INCR )
C     ------------------------------------------------------end big loop
200  CONTINUE
     WRITE(*, '(1X, '' STOP -- ECOUNT REACHED MAX'')')
     CLOSE( 8, STATUS = 'KEEP')
     CLOSE( 9, STATUS = 'KEEP')
     END
C    ****************************************************************
     SUBROUTINE MILES(XINIT, LAGPT)
C    write x distance from lagrange point in miles.
     REAL*8  XINIT(6), LAGPT, SPEED
```

```
      SPEED = 655.2D0 * DSQRT( XINIT(4)**2 + XINIT(5)**2 + XINIT(6)**2)
      WRITE(*, '(1x, '' X Dist from L2 (mi)= '',f14.3,3x,''SPEED ='',
     &      f7.3, ''mph'', ''  Ht='', f10.3, ''mi'')') 2.39d05 *
     &      (XINIT(1) - LAGPT), SPEED, 2.39d05*XINIT(3)
      RETURN
      END
C     ******************************************************************
      SUBROUTINE JACOBI( Y, JCONST, R1, R2 )
c     compute jacobi constant (also hamiltonian if desired).
      REAL*8   Y(42), R1, R2, EARTHX, MOONX, MEARTH, MMOON, JCONST
      COMMON    MEARTH, MMOON, EARTHX, MOONX
        R1      = DSQRT( (EARTHX-Y(1))**2 +Y(2)**2 +Y(3)**2)
        R2      = DSQRT( (MOONX -Y(1))**2 +Y(2)**2 +Y(3)**2)
        JCONST  = -( Y(4)**2 + Y(5)**2 + Y(6)**2 ) +Y(1)**2 +Y(2)**2
     &            + 2.0d0*(MEARTH/R1 + MMOON/R2)
c       H0          = 0.5D0*( (Y(4)-Y(2))**2 +(Y(5)+Y(1))**2 + Y(6)**2) +
c     &    Y(2)*(Y(4)-Y(2)) - Y(1)*(Y(5)+Y(1)) - MEARTH/R1 - MMOON/R2
        RETURN
        END
C     ******************************************************************
      SUBROUTINE CTIME( Y, TCROSS, ECOUNT)
c     this finds time when trajectory crosses the Poincare plane { y=0}
c     using newton iteration; this version uses EDO (RK78 integration).
      REAL*8    Y(6), TCROSS, TREF, YREF(6), TFINAL
      INTEGER*4 K, ICOUNT, ecount
      REAL*8    TOLD, TNEW, YOLD, YNEW, TPRINT
c     ------------------------------------------------variables for EDO
      REAL*8    T,TOUT,H,HMIN,HMAX,RELERR, ABSERR,WORK(90)
      INTEGER*4 IOPT,NEQN1, NMAX,IFLAG, IWORK(1)
      EXTERNAL  DIFEQ1
c     ---------------------------------initialize for RKF78 integration
      IOPT      = 1
      NEQN1     = 6
      T         = 0.0D0
c     --------------------i'm guessing on what to set these
      TFINAL    = 5.3D0
      H         = 1.0D-05
      HMIN      = 1.0D-06
      HMAX      = 1.0D0
      NMAX      = 10000
      RELERR    = 1.0D-14
      ABSERR    = 1.0D-14
      IFLAG     = 4
      TOUT      = T
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
c     ---------------------------------------integrate to reference point
      TOUT      = 0.92D0*TCROSS
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
c     ---------------------------------------save reference values
      TREF      = TOUT
      DO 10 K   = 1,6
        YREF(K) = Y(K)
10    CONTINUE
c     -------------------------use spline to get good guess for newton
      TPRINT = 0.01D0
      YOLD   = Y(2)
      TOLD   = T
      TOUT   = T + TPRINT
20    CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
      YNEW   = Y(2)
      TNEW   = TOUT
      IF ( YOLD*YNEW .GT. 0.0D0) THEN
        YOLD   = YNEW
```

```
        TOLD   = TNEW
        TOUT   = T + TPRINT
        GOTO 20
      ENDIF
c     ----------------------------------now refine with newton's method
      ICOUNT     = 0
c     ------------------------------------------------preliminary check
      IF( DABS(Y(2)) .LT. 1.0D-13 ) GOTO 100
c     ---------------------------------------------first newton iterate
      TNEW       = TOUT - Y(2)/Y(5)
      TOUT       = TNEW
c     ---------------------------------------loop using newton's method
c                                        t(n+1) = t(n) - f(t(n))/f'(t(n))
c                                        to find zero of y(2) as fcn of t
 50   ICOUNT     = ICOUNT + 1
c     ------------------------------------------check for bad iterates
      IF( TOUT. LT. TREF ) THEN
        WRITE(*, '(1X, ''TOUT TOO SMALL ---> STOP'')')
        STOP
      ENDIF
      IF (TOUT .GT. TFINAL ) THEN
        WRITE(*, '(1X, ''TOUT TOO BIG   ---> STOP'')')
        STOP
      ENDIF
      IF( ICOUNT .GT. 15 ) THEN
        WRITE(*, '(1X, ''TOO MANY ITERATES IN NEWTON ---> STOP'')')
        STOP
      ENDIF
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
c     ------------------------------------------is y(2) effectively 0?
      IF( DABS(Y(2)) .LT. 1.0D-13 ) GOTO 100
c     ---------------------------------------compute next newton value
      TNEW = TOUT - Y(2)/Y(5)
c     --------------------------------reinitialize to iterate with value
      DO 80 K    = 1,6
        Y(K)     = YREF(K)
 80   CONTINUE
      T          = TREF
      IFLAG      = 1
      TOUT       = T
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
      TOUT       = TNEW
      GOTO 50
c     ----------------------------------------crossing time is now known
 100  TCROSS = TOUT
      RETURN
      END
c     ***********************************************************************
      SUBROUTINE INIT( T, XINIT, X)
c     reinitialize arrays for integration.
      REAL*8     T, XINIT(6), X(42)
      INTEGER*4 K
      T = 0.0D0
      DO 10 K = 1, 6
        X(K)   = XINIT(K)
 10   CONTINUE
c     ----------initialize "variational matrix" "P(i,j)= X(6*j+i)= ID"
      DO 20  K = 7, 42
        X(K) = 0.0D0
 20   CONTINUE
      DO 30 K = 1,6
        X(6*K+K) = 1.0D0
 30   CONTINUE
      RETURN
```

```
      END
C     ***************************************************************
      SUBROUTINE X1FIX(A, XINIT, X, Z, cond)
C     this computes a newton iterate to refine ics to close orbit,
C     iterating with "x(1)" held fixed.  [A,XINIT,X,XD unchanged here]
      REAL*8          A(6,6), XINIT(6), X(42), Z(6), XD(6)
      REAL*8          DR(4,4), DG(4,4), B(4), COND, DEWORK(4)
      INTEGER*4       I, J, IPVT(4), N, NDIM
      COMMON/CART/    XD
      DO 62 I       = 1,4
        DO 60 J     = 1,4
          DR(I,J)   = A(I+2, J+2) - XD(I+2) *A(2,J+2) / XD(2)
 60     CONTINUE
        B(I)        = XINIT(i+2) - X(i+2)
 62   CONTINUE
      DO 70 I       = 1,4
        DO 65 J     = 1,4
          DG(I,J)   = DR(I,J)
 65     CONTINUE
        DG(I,I)     = DG(I,I) - 1.0D0
 70   CONTINUE
      NDIM          = 4
      N             = 4
      CALL DECOMP(NDIM, N, DG, COND, IPVT, DEWORK)
C      write(*,'(1x, ''cond of DG = '', d25.17)') cond
 80   CALL SOLVE( NDIM, N, DG, B,    IPVT)
      Z(1)          = XINIT(1)
      Z(2)          = XINIT(2)
      Z(3)          = XINIT(3) + B(1)
      Z(4)          = XINIT(4) + B(2)
      Z(5)          = XINIT(5) + B(3)
      Z(6)          = XINIT(6) + B(4)
      RETURN
      END
C     ***************************************************************
      SUBROUTINE X3FIX(A, XINIT, X, Z, cond)
C     this computes a newton iterate to refine ics to close orbit,
C     iterating with "x(3)" held fixed.  [A,XINIT,X,XD unchanged here]
      REAL*8          A(6,6), XINIT(6),X(42),Z(6), XD(6)
      REAL*8          DR(4,4), DG(4,4), B(4), COND, DEWORK(4)
      INTEGER*4       I, J, IPVT(4), N, NDIM,L
      COMMON/CART/    XD
      DO 40 I = 1,4
        IF (I .EQ. 1)  L = 1
        IF (I .EQ. 2)  L = 4
        IF (I .EQ. 3)  L = 5
        IF (I .EQ. 4)  L = 6
        DR(I,1)   = A(L, 1) - XD(L) *A(2,1) / XD(2)
        DR(I,2)   = A(L, 4) - XD(L) *A(2,4) / XD(2)
        DR(I,3)   = A(L, 5) - XD(L) *A(2,5) / XD(2)
        DR(I,4)   = A(L, 6) - XD(L) *A(2,6) / XD(2)
        B(I)      = XINIT(L) - X(L)
 40   CONTINUE
      DO 70 I       = 1,4
        DO 65 J     = 1,4
          DG(I,J)   = DR(I,J)
 65     CONTINUE
        DG(I,I)     = DG(I,I) - 1.0D0
 70   CONTINUE
      NDIM          = 4
      N             = 4
      CALL DECOMP(NDIM, N, DG, COND, IPVT, DEWORK)
C      write(*,'(1x, ''cond of DG = '', d25.17)') cond
 80   CALL SOLVE(  NDIM, N, DG, B,    IPVT)
      Z(1)          = XINIT(1) + B(1)
      Z(2)          = XINIT(2)
```

```
      Z(3)          = XINIT(3)
      Z(4)          = XINIT(4) + B(2)
      Z(5)          = XINIT(5) + B(3)
      Z(6)          = XINIT(6) + B(4)
      RETURN
      END
C     ***************************************************************
      SUBROUTINE X5FIX(A, XINIT, X, Z, cond)
C     this computes a newton iterate to refine ics to close orbit,
C     iterating with "x(5)" held fixed.  [A,XINIT,X,XD unchanged here]
      REAL*8        A(6,6), XINIT(6),X(42),Z(6), XD(6)
      REAL*8        DR(4,4), DG(4,4), B(4), COND, DEWORK(4)
      INTEGER*4     I, J, IPVT(4), N, NDIM,L
      COMMON/CART/  XD
      DO 40 I = 1,4
         IF (I .EQ. 1)  L = 1
         IF (I .EQ. 2)  L = 3
         IF (I .EQ. 3)  L = 4
         IF (I .EQ. 4)  L = 6
         DR(I,1)    = A(L, 1) - XD(L) *A(2,1) / XD(2)
         DR(I,2)    = A(L, 3) - XD(L) *A(2,3) / XD(2)
         DR(I,3)    = A(L, 4) - XD(L) *A(2,4) / XD(2)
         DR(I,4)    = A(L, 6) - XD(L) *A(2,6) / XD(2)
         B(I)       = XINIT(L) - X(L)
40    CONTINUE
      DO 70 I       = 1,4
         DO 65 J    = 1,4
           DG(I,J)  = DR(I,J)
65       CONTINUE
       DG(I,I)      = DG(I,I) - 1.0D0
70    CONTINUE
      NDIM          = 4
      N             = 4
      CALL DECOMP(NDIM, N, DG, COND, IPVT, DEWORK)
C      write(*,'(1x, ''cond of DG = '', d25.17)') cond
80    CALL SOLVE(  NDIM, N, DG, B,    IPVT)
      Z(1)          = XINIT(1) + B(1)
      Z(2)          = XINIT(2)
      Z(3)          = XINIT(3) + B(2)
      Z(4)          = XINIT(4) + B(3)
      Z(5)          = XINIT(5)
      Z(6)          = XINIT(6) + B(4)
      RETURN
      END
C     ***************************************************************
      SUBROUTINE PLOTTO( XINIT, TCROSS)
C     this writes points on a trajectory to a data file for plotting.
      REAL*8     XINIT(6), TCROSS, Y(6),MEARTH, MMOON, EARTHX, MOONX
      REAL*8     TFINAL, TPRINT
      REAL*8     T,TOUT,H,HMIN,HMAX,RELERR, ABSERR,WORK(90)
      INTEGER*4  IOPT,NEQN1, NMAX,IFLAG, IWORK(1),I
      COMMON     MEARTH, MMOON, EARTHX, MOONX
      EXTERNAL   DIFEQ1
C      OPEN (4, FILE = 'xy.dat', STATUS = 'new')
C      OPEN (5, FILE = 'yz.dat', STATUS = 'new')
      OPEN (7, FILE = 'xyz.dat', STATUS = 'NEW')
      DO 5 I = 1,6
        Y(I) = XINIT(I)
5     CONTINUE
C     ---------------------------------initialize for RKF78 integration
      IOPT      = 1
      NEQN1     = 6
      T         = 0.0D0
      TOUT      = T
      H         = 1.0D-05
      HMIN      = 1.0D-06
```

```
      HMAX      = 1.0D0
      NMAX      = 10000
      RELERR    = 1.0D-14
      ABSERR    = 1.0D-14
      IFLAG     = 4
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
      TOUT      = T
   10 CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
c        WRITE(4, 111) Y(1), Y(2)
c        WRITE(5, 111) Y(2), Y(3)
  111   FORMAT(1x, f15.7, 3x, f15.7)
        WRITE(7, 113) Y(1),Y(2),Y(3)
  113   FORMAT(1X, F15.7, 3X, F15.7, 3X, F15.7)
        TOUT    = T + TPRINT
        IF (T .LT. TFINAL) GOTO 10
c     CLOSE(  4, status = 'keep')
c     CLOSE(  5, status = 'keep')
      CLOSE(  7, STATUS = 'KEEP')
cc      write(*, '(1x, '' stopping in PLOTTO'' )')
cc      stop
      RETURN
      END
c     ***********************************************************************
      SUBROUTINE SYMMET( XINIT, TCROSS, SYMM,XMAX)
c     check for orthogonal crossing at half period ==> orbit is sym -
c     metric with the symmetry matrix diag(1,-1,1,-1,1,-1) =: L246.
      REAL*8       Y(6), XINIT(6), TCROSS, MEARTH, MMOON, EARTHX, MOONX
      INTEGER*4    I
      CHARACTER*3 SYMM
      REAL*8       T,TOUT,H,HMIN,HMAX,RELERR, ABSERR,WORK(90), XMAX
      INTEGER*4    IOPT,NEQN1, NMAX,IFLAG, IWORK(1)
      COMMON       MEARTH, MMOON, EARTHX, MOONX
      EXTERNAL  DIFEQ1
c     ---------------------------------initialize for RKF78 integration
      IOPT      = 1
      NEQN1     = 6
      T         = 0.0D0
      TOUT      = T
      H         = 1.0D-05
      HMIN      = 1.0D-06
      HMAX      = 1.0D0
      NMAX      = 10000
      RELERR    = 1.0D-14
      ABSERR    = 1.0D-14
      IFLAG     = 4
      DO 5 I    = 1,6
        Y(I)    = XINIT(I)
    5 CONTINUE
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
      TOUT      = TCROSS/2.0D0
      CALL EDO( DIFEQ1,IOPT, NEQN1, Y,T,TOUT,H,HMIN,HMAX,NMAX,RELERR,
     &          ABSERR,IFLAG,WORK,IWORK)
c       do 10 i = 1,3
c       write(*,'(1x,''half period pt: '',d17.10,3x,d17.10)') y(i),y(i+3)
c 10    continue
      IF (  DSQRT( Y(2)**2 + Y(4)**2 + Y(6)**2 ) .LT. 1.0D-08) THEN
        SYMM = '246'
      ELSE
        SYMM = 'NO'
      ENDIF
      XMAX = Y(1)
      RETURN
      END
```

```
c     ***********************************************************************
      SUBROUTINE STABCO(WR, WI, BROUK1, BROUK2)
c     this computes the stability coeff FROM the eigenvalues.
      REAL*8      WR(4), WI(4), BROUK1, BROUK2
      COMPLEX*16 LAMBDA(4), SIND1, SIND2
      LAMBDA(1) = DCMPLX( WR(1), WI(1) )
      LAMBDA(2) = DCMPLX( WR(2), WI(2) )
      LAMBDA(3) = DCMPLX( WR(3), WI(3) )
      LAMBDA(4) = DCMPLX( WR(4), WI(4) )
      SIND1     = LAMBDA(1) + LAMBDA(2)
      SIND2     = LAMBDA(3) + LAMBDA(4)
      BROUK1    = DREAL( -( SIND1 + SIND2 )     )
      BROUK2    = DREAL( 2.0D0 + SIND1 * SIND2 )
      RETURN
      END
c     ***********************************************************************
      SUBROUTINE NIC15( ECOUNT, XINIT, X1SAVE, X5SAVE, INCR )
c     ---extrapolate New Initial Conditions: increment x(1), find x(5).
      REAL*8    XINIT(6), X1SAVE, X5SAVE, INCR, SLOPE, INTCPT
      INTEGER*4 ECOUNT, N,K
      REAL*8    X1(5), X5(5), X1STAR, X5STAR,BX5(5),CX5(5),DX5(5),SEVAL
      N = 5
c     ----------------------------use linear extrapolation to get started
      IF( ECOUNT .LE. 4 ) THEN
        SLOPE  = ( XINIT(5)-X5SAVE ) / ( XINIT(1)-X1SAVE )
        INTCPT = X5SAVE - SLOPE*X1SAVE
c       ------------------------------fill array for spline; must have
c                                              abcissa in ascending order
        X1(N+1 - ECOUNT) = XINIT(1)
        X5(N+1 - ECOUNT) = XINIT(5)
        X1SAVE           = XINIT(1)
        X5SAVE           = XINIT(5)
        XINIT(1)         = XINIT(1) - INCR
        XINIT(5)         = SLOPE*XINIT(1) + INTCPT
        RETURN
      ENDIF
c     ---------------------------------------------now cubic spline ready
      X1(1)     = XINIT(1)
      X5(1)     = XINIT(5)
      X1SAVE    = XINIT(1)
      X5SAVE    = XINIT(5)
      CALL SPLINE( N, X1, X5, BX5, CX5, DX5 )
      X1STAR    = XINIT(1) - INCR
      X5STAR    = SEVAL( N, X1STAR, X1, X5, BX5, CX5, DX5 )
      XINIT(1) = X1STAR
      XINIT(5) = X5STAR
c     ------------------------------------prepare arrays for next time
      DO 10 K = 0,3
        X1(N-K) = X1(N-K-1)
        X5(N-K) = X5(N-K-1)
10    CONTINUE
      RETURN
      END
c     ***********************************************************************
      SUBROUTINE NIC135( ECOUNT, XINIT,  INCR )
c     new ics by incrementing x(1) and splining for x(3), x(5).
      REAL*8    XINIT(6), INCR, BX5(5), CX5(5),DX5(5),SEVAL, X5STAR
      INTEGER*4 ECOUNT, N,K
      REAL*8    X1(5),X3(5),X5(5),BX3(5),CX3(5),DX3(5),X1STAR,X3STAR
      N = 5
c       ----------------------------fill array for spline; must have
c
      IF( ECOUNT .EQ. 1 ) THEN                  abcissa in ascending order
c     ------------------------block for "just a little way on halo"
        X1(5) =  0.11202386386228D+01
        X3(5) = -0.45234293999200D-02
```

```
          X5(5) =  0.17647048488849D+00
c
          X1(4) =  0.11202113181410D+01
          X3(4) = -0.49234293999200D-02
          X5(4) =  0.17654986343764D+00
c
          X1(3) =  0.11201816761746D+01
          X3(3) = -0.53234293999200D-02
          X5(3) =  0.17663597823708D+00
c
          X1(2) =  0.11201497107957D+01
          X3(2) = -0.57234293999200D-02
          X5(2) =  0.17672883275638D+00
        ENDIF
c       ----------------------------------------now cubic spline ready
        X1(1)     = XINIT(1)
        X3(1)     = XINIT(3)
        X5(1)     = XINIT(5)
        CALL SPLINE( N, X1, X3, BX3, CX3, DX3 )
        CALL SPLINE( N, X1, X5, BX5, CX5, DX5 )
        X1STAR    = XINIT(1) - INCR
        X3STAR    = SEVAL( N, X1STAR, X1, X3, BX3, CX3, DX3 )
        X5STAR    = SEVAL( N, X1STAR, X1, X5, BX5, CX5, DX5 )
        XINIT(1) = X1STAR
        XINIT(3) = X3STAR
        XINIT(5) = X5STAR
c       ------------------------------------prepare arrays for next time
        DO 10 K = 0,3
          X1(N-K) = X1(N-K-1)
          X3(N-K) = X3(N-K-1)
          X5(N-K) = X5(N-K-1)
10      CONTINUE
        RETURN
        END
c       ***********************************************************************
        SUBROUTINE NIC315( ECOUNT, XINIT, INCR )
c           this generates new initial conditions for continuing the halo
c        orbits, starting from the bifurcation point. First we take small
c       steps in the direction of the Eigenvector, then we use spline ex-
c         trapolation, with x3 indep variable, x1 and x5 functions of x3.
        INTEGER*4 ECOUNT, N, K
        REAL*8    XINIT(6),INCR,X1STAR,X5STAR,SSTAR,X1(5),X3(5), X5(5)
        REAL*8    BX1(5),CX1(5),DX1(5), BX5(5),CX5(5),DX5(5),SEVAL,SCALE
        N     = 5
        SCALE = 2.0D-06
c       ------------------------------take small steps along eigenvector
c                                                       to fill spline arrays
        IF (ECOUNT .LE. 4) THEN
c         ---------------------------neg to make array in ascending order
          X3( ECOUNT) = -XINIT(3)
          X1( ECOUNT) =  XINIT(1)
          X5( ECOUNT) =  XINIT(5)
          XINIT(3)          =  XINIT(3)  + SCALE*(-15.42867499d0)
          XINIT(6)          =  XINIT(6)  + SCALE*(- 0.26484066d0)
c          WRITE(*,'(1X, ''XINIT 1, 3, 5'', 3D18.11)' ) XINIT(1),XINIT(3),
c     &         XINIT(5)
          RETURN
        ENDIF
c       ----------------------------------------------now ready for spline
        X3(5)     = -XINIT(3)
        X5(5)     =  XINIT(5)
        X1(5)     =  XINIT(1)
        CALL SPLINE(N,X3,X1, BX1, CX1, DX1 )
        CALL SPLINE(N,X3,X5, BX5, CX5, DX5 )
        SSTAR     = -( XINIT(3) - INCR )
        X1STAR    =  SEVAL(N,SSTAR, X3, X1, BX1, CX1, DX1 )
```

```
      X5STAR    =  SEVAL(N,SSTAR, X3, X5, BX5, CX5, DX5 )
      XINIT(3) = -SSTAR
      XINIT(1) =  X1STAR
      XINIT(5) =  X5STAR
C     ------------------------------------prepare arrays for next time
      DO 10 K  = 1,4
         X5(K) = X5(K+1)
         X1(K) = X1(K+1)
         X3(K) = X3(K+1)
 10   CONTINUE
      RETURN
      END
C     ****************************************************************
      SUBROUTINE DIFEQ1( T, X, NEQN1, XDOT)
      REAL*8    T,X(6), XDOT(6), R1, R2, R1N3, R2N3, TEMP
      INTEGER*4 NEQN1
      REAL*8    ME, MM, XE, XM
      COMMON    ME, MM, XE, XM
      R1      = DSQRT( (XE-X(1))**2 +X(2)**2 +X(3)**2)
      R2      = DSQRT( (XM -X(1))**2 +X(2)**2 +X(3)**2)
      R1N3    = 1.0D0 / (R1*R1*R1)
      R2N3    = 1.0D0 / (R2*R2*R2)
      TEMP    = ME*R1N3 + MM*R2N3
      XDOT(1) = X(4)
      XDOT(2) = X(5)
      XDOT(3) = X(6)
      XDOT(4) = 2.0D0*X(5) +X(1) -ME*(X(1)-XE)*R1N3 - MM*(X(1)-XM)*R2N3
      XDOT(5) = -2.0D0*X(4) +X(2) - TEMP * X(2)
      XDOT(6) =                   - TEMP * X(3)
      RETURN
      END
C     ****************************************************************
      SUBROUTINE DIFEQ2( T, X, NEQN2, XDOT)
c     integrates variational equations with more efficient coding
      INTEGER*4 K,J, NEQN2
      REAL*8    T,X(42), XDOT(42), R1, R2, F(6,6), A
      REAL*8    ME, MM, XE, XM, R1N3,R1N5,R2N3,R2N5, XD(6)
      COMMON    ME, MM, XE, XM
      COMMON/CART/ XD
      R1      = DSQRT( ( X(1)-XE )**2 +X(2)**2 +X(3)**2)
      R2      = DSQRT( ( X(1)-XM  )**2 +X(2)**2 +X(3)**2)
      R1N3    = R1**(-3)
      R1N5    = R1**(-5)
      R2N3    = R2**(-3)
      R2N5    = R2**(-5)
      XDOT(1) =  X(4)
      XDOT(2) =  X(5)
      XDOT(3) =  X(6)
      XDOT(4) =  2.0D0*X(5) + X(1) - ME*( X(1) -XE )*R1N3 -
     &                               MM*( X(1) -XM )*R2N3
      XDOT(5) = -2.0D0*X(4) +X(2) - (ME*R1N3 + MM*R2N3)*X(2)
      XDOT(6) = -X(3)*(ME*R1N3 + MM*R2N3)
C     ------------------------------calculate 2nd order partial derivs
      A       = ME*( X(1) - XE )*R1N5 + MM*( X(1) - XM )*R2N5
      F(4,1)  = 3.0D0*( ME*(X(1) -XE)**2*R1N5 + MM*(X(1)-XM)**2*R2N5 )
     &          - ME*R1N3 - MM*R2N3 + 1.0D0
      F(4,2) = 3.0D0*X(2)*A
      F(4,3) = 3.0D0*X(3)*A
      F(4,4) = 0.0D0
      F(4,5) = 2.0D0
      F(4,6) = 0.0D0
      F(5,1) = 3.0D0*X(2)*A
      F(5,2) = 3.D0*X(2)**2*(ME*R1N5 +MM*R2N5) -R2N3*MM -R1N3*ME +1.0D0
      F(5,3) = 3.0D0*X(2)*X(3)*( R2N5*MM + R1N5*ME)
      F(5,4) = -2.0D0
      F(5,5) =  0.0D0
```

```
      F(5,6) =  0.0D0
      F(6,1) = 3.0D0*X(3)*A
      F(6,2) = F(5,3)
      F(6,3) = 3.0D0*X(3)*X(3)*(R2N5*MM + R1N5*ME) -R2N3*MM  -R1N3*ME
      F(6,4) = 0.0D0
      F(6,5) = 0.0D0
      F(6,6) = 0.0D0
C     -------------------------------form variational equations matrix
      DO 20 K = 1,6
        XDOT(6*K+1) = X(6*K+4)
        XDOT(6*K+2) = X(6*K+5)
        XDOT(6*K+3) = X(6*K+6)
        XDOT(6*K+4) = 0.0D0
        XDOT(6*K+5) = 0.0D0
        XDOT(6*K+6) = 0.0D0
        DO 10 J = 1, 6
          XDOT(6*K+4) = XDOT(6*K+4) + F(4,J)*X(6*K+J)
          XDOT(6*K+5) = XDOT(6*K+5) + F(5,J)*X(6*K+J)
          XDOT(6*K+6) = XDOT(6*K+6) + F(6,J)*X(6*K+J)
   10   CONTINUE
   20 CONTINUE
      do 30 k = 1,6
        xd(k) = xdot(k)
   30 continue
      RETURN
      END
C     ***************************************************************
```