APPENDIX B:

FORTRAN FILES

```
c*************************************************************************c
c    Transient Stability assessment using an second order auto-regressive model    c
c*************************************************************************c
c                                                                         c
c The data files are:  3m_pf.dat     9     'pre-fault data'               c
c                      3m_redu.dat   10    'post-fault data'              c
c                      3m_reduf.dat  11    'fault-on data'                c
c                                                                         c
c The result files are:3_mx.dat      12    'angles & time output'         c
c                      3_mxdot.dat   14    'speed & acceleration output'  c
c                      tmp.dat       15    'transient energy'             c
c                      3m_v.dat      17    'C.O.A energy level'           c
c                      angle.dat     20    'estimated angles & time'      c
c                                                                         c
$DEBUG
$INCLUDE:'subrout1.for'
c                                                                         c
c       Main Variables :                                                  c
c                    x  :  angle from the step by step integration (i=1,n-1)   c
c                    x  :  speed from the step by step integration   (i=n,2(n-1))  c
c                    f  :acceleration from the step by step integration (i=n,2(n-1))  c
c                    Vpe  :                                               c
c                          Transient energy calculated with x             c
c                    Vke  :                                               c
c                    gradEp: directinal derivative                        c
c                    Predict:  Estimated values of the angles             c
c                    Predict:Estimated values of the speed                c
c                    tcl  :  clearing time                                c
c                    a    :  number of point used for the estimation      c
c                    h    :  step for the integration of the swing equation  c
c                    Vtotal:  Total transient energy of the system at tcl c
c                    inside: data used to check whether or not the path has crossed  c
c                          the PEBS.                                      c
c                                                                         c
     INTEGER countr, intrat,compt,a,o,swing,l,ierr,iflag,itrmax,e,
    &          inside,inside1,inside2
```

```
c                                                                       c
      PARAMETER (in=20, in2=400)
c                                                                       c
c     Data which define the system(pre-fault,fault-on,post-fault)       c
c                                                                       c
      REAL M(in),P(in),D(in,in),C(in,in),Pcoa,Pcoan,Mt,deltas(in),
     &     dold(in),Pf(in),Df(in,in),Cf(in,in)
c                                                                       c
c     Output from the step by step integration of the swing equation    c
c                                                                       c
      DIMENSION xwrk(4,in2), xend(in2), x(in2), f(in2),fn(in2),Z(in2)
c                                                                       c
c     Input used for the estimation                                     c
c                                                                       c
      REAL Vpeold,gradepold,Vtotal,t,tf,tcl,xe(240,20),v(240,20),
     &         te(in2),h
c                                                                       c
c     Output of the estimation                                          c
c                                                                       c
      REAL gradEpn,shad(in2),kitv(5,9),kita(5,9),tot,predict(240,20)
      REAL  dold1(in),it,Vpe1,Vpe2,fred(in),s(in),jacobi1(in,in),
     &      ddelta(in),deltanew(in),frednew(in),tet,direc
c                                                                       c
c                                                                       c
c*****************data files                                          c
c                                                                       c
      OPEN(9,FILE='3m_pf.dat')
      OPEN(10,FILE='3m_redu.dat')
      OPEN(11,FILE='3m_reduf.dat')
      OPEN(12,FILE='3m_x.dat')
      OPEN(14,FILE='3m_xdot.dat')
      OPEN(15,FILE='tmp.dat')
      OPEN(17,FILE='3m_v.dat')
c                                                                       c
c*****************result files                                       c
c                                                                       c
      OPEN(20,FILE='angle.dat')
      OPEN(22,FILE='vitess.dat')
      OPEN(21,FILE='moi.dat')
      OPEN(23,FILE='essai.dat')
      OPEN(24,FILE='bcu.dat')
c                                                                       c
c****************** Read n (Pre-fault)                                 c
```

```
c                                                                       c
READ(9,*) n
     mm=2*(n-1)
c                                                                       c
c******************* Read  P, and deltas (Pre-fault)                   c
c                                                                       c
      DO  i = 1, n
        READ(9,*) M(i), dold(i)
     END DO
     CLOSE(9)
c                                                                       c
c****************** Get Mt                                           c
c                                                                       c
     Mt=0.
     DO i=1,n
       Mt=Mt+M(i)
     END DO
c                                                                       c
c******************* Read n (post-fault)                               c
c                                                                       c
     READ(10,*) n
     mm=2*(n-1)
c                                                                       c
c******************* Read  P, and deltas (Post-fault)                  c
c                                                                       c
     DO  i = 1, n
        READ(10,*) P(i), deltas(i)
     END DO
c                                                                       c
c******************** Read C and D (Post-fault)                        c
c                                                                       c
      DO 1 i = 1, n-1
        DO 1 j = i+1, n
          READ(10,*) C(i,j), D(i,j)
          C(j,i)=C(i,j)
1         D(j,i)=D(i,j)
c                                                                       c
c******************* Read Pf (Fault)                                   c
c                                                                       c
     READ(11,*) n
     DO  i = 1, n
        READ(11,*)  Pf(i)
     END DO
```

```
c                                                                       c
c******************** Read Cf and Df (Fault)                           c
c                                                                       c
      DO 2 i = 1, n-1
        DO 2 j = i+1, n
          READ(11,*) Cf(i,j), Df(i,j)
          Cf(j,i)=Cf(i,j)
2          Df(j,i)=Df(i,j)
c                                                                       c
c******************** Read the step interval for integration           c
c                                                                       c
      WRITE(*,102)
102   FORMAT(' enter the step interval for integration ---->',\)
      READ(*,*) h
      WRITE(*,103)
103   FORMAT(' enter the printing frequency ---->',\)
      READ(*,*) intrat
      WRITE(*,105)
105   FORMAT(' enter the clearing time ---->',\)
      READ(*,*) tcl
c                                                                       c
c******************** Set the initial conditions                       c
c                                                                       c
      countr=0
      t=0.
      DO i=1,n-1
        x(i)=dold(i)
        x(i+n-1)=0.0
      END DO
c                                                                       c
c********************* Evaluate the derivatives                        c
c                                                                       c
      CALL fis(n,in,M,Mt,Pf,Cf,Df,x,Pcoa,f,gradEp)
c                                                                       c
      CALL Energy(n,in,in2,M,P,x,deltas,C,D,Vpe,Vke,Vp)
c                                                                       c
c********************* Write the initial results                       c
c                                                                       c
      WRITE(12,'(1x,1p,6e13.5)') (x(i),i=1,n-1),t
      WRITE(14,'(1x,1p,6e13.5)') (x(i),f(i),i=n,mm )
      WRITE(15,'(1x,1p,6e13.5)') t, Vke, Vpe, Vke+Vpe, Vp, Vpe-Vp
      d0 = 0.
      w0 = 0.
```

```
      WRITE(17,'(1x,1p,6e13.5)') t, Pcoa, d0,w0,gradEp
c                                                                       c
c********************** Start the iterative process (Fault-on period)  c
c                                                                       c
      o=1
1000  continue
c                                                                       c
      CALL rksyst(n,in,mm,h,x,xend,xwrk,Cf,Df,M,Pf,Mt,f)
c                                                                       c
c     update the state variables and time                               c
c                                                                       c
      DO i=1,mm
        x(i)=xend(i)
      END DO
c                                                                       c
      t=t+h
      te(o)=t
      DO i=1,n-1
        ii=i+n-1
        xe(o,i)=x(i)
        v(o,i)=x(ii)
      END DO
c      update dx/dt                                                     c
c                                                                       c
      w0 = w0 + (Pcoa/Mt)*h
      d0 = d0 + w0 * h
      CALL fis(n,in,M,Mt,Pf,Cf,Df,x,Pcoa,f,gradEp)
      CALL Energy(n,in,in2,M,P,x,deltas,C,D,Vpe,Vke,Vp)
      countr=countr+1
      IF(countr.ge.intrat) THEN
        countr=0
      WRITE(12,'(1x,1p,6e13.5)') (x(i),i=1,n-1),t
      WRITE(14,'(1x,1p,6e13.5)') (x(i),f(i),i=n,mm )
      WRITE(15,'(1x,1p,6e13.5)') t, Vke, Vpe, Vke+Vpe, Vp, Vpe-Vp
      WRITE(17,'(1x,1p,6e13.5)') t, Pcoa, d0, w0,gradEp
      ENDIF
      IF(intrat.eq.0) STOP ' adios'
c                                                                       c
c      when to stop                                                     c
c                                                                       c
      t2=t+1.0e-5
      IF(t2.ge.tcl) GOTO 1001
      o=o+1
```

```
      GOTO 1000
1001  CONTINUE
c                                                                       c
c***************calculate the total transient energy at tcl            c
c                                                                       c
      Vtotal=Vke+Vpe
      WRITE(*,'(10x,A,\)') 'The total energy at tcl is :'
      WRITE(6,*) Vtotal
c                                                                       c
c######################################################## POST-FAULT    c
c                                                                       c
c********************* Read the step interval for integration         c
c                                                                       c
      WRITE(*,102)
      READ(*,*) h
      WRITE(*,103)
      READ(*,*) intrat
      WRITE(*,'(10x,A,\)') 'enter the final time ---> '
      READ(*,*) tf
      WRITE(*,'(10x,A,\)') 'enter the prediction time:t/h=a='
      READ(*,*) a
c                                                                       c
c********************** Write the initial results                      c
c                                                                       c
      WRITE(12,'(1x,1p,6e13.5)') (x(i),i=1,n-1),t
      WRITE(14,'(1x,1p,6e13.5)') (x(i),f(i),i=n,mm )
      WRITE(15,'(1x,1p,6e13.5)') t, Vke, Vpe, Vke+Vpe, Vp, Vpe-Vp
      WRITE(17,'(1x,1p,6e13.5)') t, Pcoa, d0, w0,gradEp
c                                                                       c
c************************ Start the iterative process (Post-Fault)     c
c                                                                       c
      compt=1
      swing=1
c                                                                       c
c****************Begin the first swing                                  c
c                                                                       c
2000  continue
c                                                                       c
       CALL rksyst(n,in,mm,h,x,xend,xwrk,C,D,M,P,Mt,f)
c                                                                       c
      compt=compt+1
c                                                                       c
c          update the state variables and time                          c
```

```
c                                                                       c
      DO i=1,mm
        x(i)=xend(i)
      END DO
c                                                                       c
      t=t+h
      te(o)=t
      DO i=1,n-1
        ii=i+n-1
        xe(o,i)=x(i)
        v(o,i)=x(ii)
      END DO
c   update dx/dt                                                        c
c                                                                       c
      w0 = w0 + (Pcoa/Mt)*h
      d0 = d0 + w0 * h
      Vpeold=Vpe
      gradepold=gradEp
      CALL fis(n,in,M,Mt,P,C,D,x,Pcoa,f,gradEp)
      CALL Energy(n,in,in2,M,P,x,deltas,C,D,Vpe,Vke,Vp)
      CALL sep1(n,in,mm,h,M,Mt,P,C,D,x,deltas,inside)
c                                                                       c
c   write results                                                       c
c                                                                       c
      countr=countr+1
      IF(countr.ge.intrat) THEN
        countr=0
c                                                                       c
c*****************write the results of the step by step integration    c
c                                                                       c
      WRITE(12,'(1x,1p,6e13.5)') (x(i),i=1,n-1),t
      WRITE(14,'(1x,1p,6e13.5)') (x(i),f(i),i=n,mm )
      WRITE(15,'(1x,1p,6e13.5)') t, Vke, Vpe, Vke+Vpe, Vp, Vpe-Vp
      WRITE(17,'(1x,1p,6e13.5)') t, Pcoa, d0, w0,gradEp
      ENDIF
c                                                                       c
      IF(inside.eq.0) GOTO 2001
      IF(compt.ge.a.and.Vpe.gt.Vpeold) THEN
c                                                                       c
c*****************Estimation with a quadratique                        c
c                                                                       c
            CALL estim1(n,a,o,xe,kita)
            CALL estim1(n,a,o,v,kitv)
```

```
c                                                                       c
c***************open 21,20 & 22 and delete the former data             c
c                                                                       c
          OPEN(21,STATUS='OLD',FILE='moi.dat')
          OPEN(20,STATUS='OLD',FILE='angle.dat')
          OPEN(22,STATUS='OLD',FILE='vitess.dat')
c                                                                       c
          gradEpn=1
c                                                                       c
          tot=te(o)
               e=o
          DO i=1,n-1
            ii=i+n-1
                predict(e,i)=xe(e,i)
                predict(e,ii)=v(e,i)
          END DO
          DO WHILE(gradEpn.gt.0)
            tot=tot+h
            DO i=1,n-1
              ii=i+n-1
              predict(e+1,i)=kita(1,i)+kita(2,i)*predict(e,i)
     &             +kita(3,i)*predict(e-1,i)
              predict(e+1,ii)=kitv(1,i)+kitv(2,i)*predict(e,ii)
     &             +kitv(3,i)*predict(e-1,ii)
              Z(i)= predict(e+1,i)
                   Z(ii)=predict(e+1,ii)
            END DO
c                                                                       c
            CALL fis(n,in,M,Mt,P,C,D,Z,Pcoan,fn,gradEpn)
            CALL sep1(n,in,mm,h,M,Mt,P,C,D,Z,deltas,inside1)
c                                                                       c
c*****************Write in the result files                           c
c                                                                       c
            WRITE(21,'(1x,1p,6e13.5)')gradEpn
            WRITE(20,'(1x,1p,6e13.5)')(Z(i),i=1,n-1),tot
            WRITE(22,'(1x,1p,6e13.5)')(Z(i),i=n,mm),tot
c                                                                       c
            IF(inside1.eq.0) GOTO 3000
                 e=e+1
          END DO
3000       CONTINUE
c                                                                       c
          CLOSE(20)
```

```
            CLOSE(21)
            CLOSE(22)
c                                                                       c
c****************check if the stab. boundary has been reached          c
c  if not take pot.energy local max on the ray from sep and the pt of the prediction  c
c                                                                       c
            tet=0
            direc=1
            DO WHILE(direc.gt.0.0)
              tet=tet+h*2
              DO i=1,n-1
                ii=i+n-1
                shad(i)=((Z(i)-xe(1,i))/tot)*tet+Z(i)
                shad(ii)=Z(i)-xe(1,i)/tot
              END DO
              CALL fis(n,in,M,Mt,P,C,D,shad,Pcoa,fred,direc)
              CALL sep1(n,in,mm,h,M,Mt,P,C,D,shad,deltas,inside2)
              IF(inside2.eq.0) GOTO 3001
            END DO
c                                                                       c
3001          CONTINUE
c                                                                       c
c**************determine the min of norm(grad) starting with shad(i)  c
c                                                                       c
            CALL start(n,in,mm,h,M,Mt,P,C,D,shad,dold1,it)
c                                                                       c
c**************dold1 is the initial point to find the cuep(N-R method) c
c                                                                       c
            CALL pot(n,in,p,dold1,deltas,C,D,Vpe1)
            CALL fredis(n,in,M,Mt,P,C,D,dold1,Pcoa,fred)
c                                                                       c
c**************find the cuep                                          c
c                                                                       c
            CALL epfndr(n,in,M,P,D,C,fred,s,Mt,dold1,jacobi1,
     &                  ddelta,l,ierr,deltanew,frednew,iflag,itrmax)
                  OPEN(9,FILE='3m_pf.dat')
                  READ(9,*) n
                  DO i=1,n
                    READ(9,*) M(i),dold(i)
                  END DO
                  CLOSE(9)
c                                                                       c
c*************dold1 is the point found by N-R                         c
```

```
c                                                                       c
           CALL pot(n,in,P,dold1,deltas,C,D,Vpe2)
c                                                                       c
c********** is dold1 the cuep?                                         c
c                                                                       c
      IF (ierr.eq.0.and.iflag.eq.0.and.Vpe2.lt.Vtotal) THEN
         WRITE(*,'(10x,A)')'unstable system & found cuep'
      ENDIF
      IF (ierr.eq.1.or.iflag.eq.1) THEN
         WRITE(*,*) 'computational problem'
      ENDIF
      IF (Vpe2.gt.Vtotal) THEN
         WRITE(*,'(10x,A)')'stable system & found cuep'
      ENDIF
      WRITE(*,*) 'Vpe at CUEP is ',Vpe2
      WRITE(*,*) 'Energy difference of ',(Vtotal-Vpe2)
      WRITE(*,*) 'cuep is',(dold1(i),i=1,n)
c                                                                       c
c**********When to stop                                               c
c                                                                       c
         IF(Vpe2.lt.0.0) GOTO 2000
         IF(Vtotal.gt.Vpe2) GOTO 2001
    END IF
c                                                                       c
    IF(t.ge.tf) GOTO 2001
    o=o+1
    GOTO 2000
2001  CONTINUE
c                                                                       c
    STOP
    END
c                                                                       c
c***********************************************************************c
c                      END OF THE MAIN PROGRAM                          c
c***********************************************************************c


c***********************************************************************c
c***********************************************************************c
c               SUBROUTINES USED IN AUTO-REGRESSVE FILE                 c
c***********************************************************************c
c***********************************************************************c
c                                                                       c
```

```
c*********************************************************************c
c     Subroutine to find the local SEP of a point on a trajectory     c
c*********************************************************************c
c                                                                     c
      SUBROUTINE sep1(n,in,mm,h,M,Mt,P,C,D,x,deltas,inside)
c                                                                     c
      INTEGER inside,i,j
      DIMENSION x(20),xprime(20)
      REAL M(n),Mt,P(in),C(in),D(in),h,deltas(n)
      REAL Pcoa,gradep,norm,norm0,f(20),xend(20),xwrk(4,20)
c                                                                     c
      norm0 = 0.0
      DO i=1,n-1
         norm0 = norm0 + (x(i)-deltas(i))**2.0
      END DO
      norm0 = norm0**0.5
      DO i=1,n-1
         j = i + n - 1
         xprime(i) = x(i)
         xprime(j) = 0.0
      END DO
c                                                                     c
      CALL fis(n,in,M,Mt,P,C,D,xprime,Pcoa,f,gradEp)
c                                                                     c
      DO i=1,2
       CALL rksyst(n,in,mm,h,xprime,xend,xwrk,C,D,M,P,Mt,f)
         DO j=1,mm
           xprime(j) = xend(j)
         END DO
       CALL fis(n,in,M,Mt,P,C,D,xprime,Pcoa,f,gradEp)
      END DO
c                                                                     c
      norm = 0.0
      DO i=1,n-1
         norm = norm + (xprime(i)-deltas(i))**2.0
      END DO
      norm = norm**0.5
c                                                                     c
      IF (norm0.lt.norm) THEN
         inside =0
      ELSE
         inside = 1
      END IF
```

```
c                                                                       c
      RETURN
      END

c***********************************************************************c
c       Subroutine which determine the starting point to                c
c         find the controlling uep.                                     c
c***********************************************************************c
c                                                                       c
      SUBROUTINE start(n,in,mm,h,M,Mt,P,C,D,shad,dold1,it)
c                                                                       c
      INTEGER n,in,i,mm
c                                                                       c
c*************input data                                              c
c                                                                       c
      REAL    h,M(in),Mt,P(in),C(in,in),D(in,in),shad(n)
c                                                                       c
c*************output data                                             c
c                                                                       c
      REAL    dold1(mm),it
      REAL    dnew(10),fnold,fnnew,fred(10)
c                                                                       c
c*************begin the iterative process to find the local           c
c             min of the norm of the gradient.                          c
c                                                                       c
      it=0
c*************starting point Xnew(e,i)                                 c
      dold1(n)=0.0
      DO i=1,n-1
        dold1(i)=shad(i)
        dold1(n)=dold1(n)-M(i)*dold1(i)/M(n)
c         WRITE(6,*) dold1(i)
      END DO
c                                                                       c
201   CONTINUE
c                                                                       c
c*************evaluate the derivatives and the gradient               c
c                                                                       c
      CALL fredis(n,in,M,Mt,P,C,D,dold1,Pcoa,fred)
c                                                                       c
c*************get the norm of vector f                                c
c                                                                       c
      fnold=fnorm(n,fred)
```

```
c                                                                       c
c**************get an updated delta                                    c
c                                                                       c
      dnew(n)=0
      DO i=1,n-1
        dnew(i)=dold1(i)+h*fred(i)
        dnew(n)=dnew(n)-M(i)*dnew(i)/M(n)
      END DO
c**************get fred for dnew                                       c
      CALL fredis(n,in,M,Mt,P,C,D,dnew,Pcoa,fred)
      fnnew=fnorm(n,fred)
      IF(fnnew.lt.fnold) THEN
        DO i=1,n
          dold1(i)=dnew(i)
        END DO
        it=it+h
        GOTO 201
      END IF
c                                                                       c
      RETURN
      END
c                                                                       c
c***********************************************************************c
c     Subroutine to determine the cuep usin a N-R method                c
c***********************************************************************c
c                                                                       c
      SUBROUTINE epfndr(n,in,M,P,D,C,fred,s,Mt,deltao,jacobi,ddelta,
     &                  l,erflag,deltanew,frednew,iflag,itrmax)
c                                                                       c
      INTEGER erflag
      PARAMETER(pi=3.141593,Rmin=1.0e-3)
c                                                                       c
      REAL M(in), P(in), D(in,in), C(in,in), fred(in), Pcoa, s(in),
     & Mt,deltao(in),jacobi(in,in),ddelta(in),deltanew(in),frednew(in)
c                                                                       c
      INTEGER l(in)
c                                                                       c
c******************** Read fnormtol and itrmax                         c
c                                                                       c
      fnormtol=0.01
c                                                                       c
      itrmax=1000
c                                                                       c
```

```
c******************** Set iteration countr = 0                                c
      itctr=0
c                                                                            c
c******************** Start iterative process                                c
2000  CONTINUE
c      WRITE(*,'(1x,I2)') itctr
c                                                                            c
c******************** Evaluate fis                                           c
      deltao(n)=0.
      DO i=1,n-1
        deltao(n)=deltao(n)-M(i)*deltao(i)/M(n)
      END DO
c                                                                            c
      CALL fredis(n,in,M,Mt,P,C,D,deltao,Pcoa,fred)
c                                                                            c
c*************** If the solution is within the specified tolerance end      c
      F2=fnorm(n,fred)
c      WRITE(*,*) ' F2 = ', F2
c                                                                            c
      IF(F2.le.fnormtol) then
c       WRITE(*,'(/,2x,A)') ' SUCCESFUL SOLUTION REACHED'
       erflag=0
       iflag=0
       GOTO 1000
      ENDIF
c                                                                            c
       itctr=itctr+1
c                                                                            c
c******************** If maximum number of iterations is exceeded end        c
      IF(itctr.GT.itrmax) then
c       WRITE(*,'(/,A,I3,A)') ' MAXIMUM NUMBER OF ITERATIONS:  ',itrmax,
c     &                  ' EXCEEDED'
        erflag=1
        GOTO 1000
      ENDIF
c******************** Get the Jacobian                                       c
      CALL       getjacobi1(n,in,M,Mt,C,D,deltao,jacobi)
c                                                                            c
      nm1=n-1
c                                                                            c
      CALL gauss(nm1,in,jacobi,l,s)
      CALL solve(nm1,in,jacobi,l,fred,ddelta)
c                                                                            c
```

```
c******************** deltan=deltao-(jacobi**-1)*fred=deltao-ddelta                c
c       ddelta=-ddelta                                                           c
      DO i=1,n-1
       ddelta(i)=-ddelta(i)
      END DO
c                                                                              c
c          linser                                                              c
      g=f2
      icntr=0
100   CONTINUE
      r=1./2.**icntr
      icntr=icntr+1
c                                                                              c
      deltanew(n)=0.
      DO i=1,n-1
        deltanew(i)=deltao(i)+ddelta(i)*r
        deltanew(n)=deltanew(n)-M(i)*deltanew(i)/M(n)
      END DO
c                                                                              c
      CALL fredis(n,in,M,Mt,P,C,D,deltanew,Pcoa,frednew)
      gnew=fnorm(n,frednew)
      IF(gnew.ge.g.and.r.ge.rmin) GO TO 100
      IF(r.lt.rmin) THEN
        iflag=1
       GOTO 1000
      ELSE
        iflag=0
      END IF
      DO i=1,n
        deltao(i)=deltanew(i)
      END DO
c                                                                              c
c******************** Goto 2000                                               c
      GOTO 2000
c                                                                              c
1000  CONTINUE
c                                                                              c
      RETURN
      END
c                                                                              c

c*******************************************************************************c
c                                                                              c
```

```
      SUBROUTINE getjacobi1(n,in,M,Mt,C,D,deltao,jacobi)
c                                                                       c
      REAL M(n), C(in,n), D(in,n), deltao(n), jacobi(in,n), Mt
c                                                                       c
      DO 1 i=1,n-1
       term1=0.0
       term2=0.0
       term4=0.0
       DO 2 k=1,n
c                                                                       c
        IF(k.NE.n) THEN
         dnk=deltao(n)-deltao(k)
         term4=term4+D(n,k)*sin(dnk)
        ENDIF
c                                                                       c
        IF(k.NE.i) THEN
         dik=deltao(i)-deltao(k)
         term1=term1+C(i,k)*cos(dik)
         term2=term2+D(i,k)*sin(dik)
        ENDIF
c                                                                       c
2       CONTINUE
        din=deltao(i)-deltao(n)
       jacobi(i,i)=-term1+(Mt-2.0*M(i))*term2/Mt
     &            -(M(i)/M(n))*(C(i,n)*cos(din)-D(i,n)*sin(din))
     &            +( 2.0*M(i)*M(i)/(Mt*M(n)) )*term4
       DO 3 j=1,n-1
c                                                                       c
        IF(j.NE.i) THEN
         term3=0.0
         DO 4 k=1,n
c                                                                       c
          IF(k.NE.j) THEN
           djk=deltao(j)-deltao(k)
           term3=term3+D(j,k)*sin(djk)
          ENDIF
c                                                                       c
4         CONTINUE
         dij=deltao(i)-deltao(j)
        term5=c(i,j)*cos(dij)-D(i,j)*sin(dij)
        term6=c(i,n)*cos(din)-D(i,n)*sin(din)
         jacobi(i,j)=term5 - 2.0 * M(i)/Mt * term3
     &             - M(j)/M(n) * term6
```

```
     &           +2.0*M(i)*M(j)/Mt/M(n) * term4
        ENDIF
c                                                                       c
3     CONTINUE
1    CONTINUE
c                                                                       c
     RETURN
     END
c                                                                       c
c***********************************************************************c
c      GAUSS AND SOLVE ARE TAKEN FROM NUMERICAL METHODS                 c
c***********************************************************************c
c                                                                       c
     SUBROUTINE gauss(n,in,a,l,s)
c                                                                       c
     DIMENSION a(in,n),l(n),s(n)
c                                                                       c
     DO 3 i=1,n
        l(i)=i
        smax=0.0
        DO 2 j=1,n
           smax=AMAX1(smax,ABS(a(i,j)))
2       CONTINUE
        s(i)=smax
3    CONTINUE
c                                                                       c
     DO 7 k=1,n-1
        rmax=0.0
        DO 4 i=k,n
           r=ABS(a(l(i),k))/s(l(i))
           if(r.LE.rmax) GO TO 4
           j=i
           rmax=r
4       CONTINUE
        lk=l(j)
        l(j)=l(k)
        l(k)=lk
        DO 6 i=k+1,n
           xmult=a(l(i),k)/a(lk,k)
           DO 5 j=k+1,n
              a(l(i),j)=a(l(i),j)-xmult*a(lk,j)
5          CONTINUE
           a(l(i),k)=xmult
```

```
6        CONTINUE
7     CONTINUE
      RETURN
      END
c                                                                         c
c     SOLVE ************************************************************c
c                                                                         c
      SUBROUTINE solve(n,in,a,l,b,x)
c                                                                         c
      DIMENSION a(in,n),l(n),b(n),x(n)
c                                                                         c
      DO 3 k=1,n-1
        DO 2 i=k+1,n
          b(l(i))=b(l(i))-a(l(i),k)*b(l(k))
2        CONTINUE
3     CONTINUE
      x(n)=b(l(n))/a(l(n),n)
      DO 5 i=n-1,1,-1
        sum=b(l(i))
        DO 4 j=i+1,n
          sum=sum-a(l(i),j)*x(j)
4        CONTINUE
        x(i)=sum/a(l(i),i)
5     CONTINUE
      RETURN
      END
c                                                                         c
c!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!c
c                                                                         c
      FUNCTION fnorm(n,fred)
c                                                                         c
      INTEGER n
      REAL fred(n),fnorm
c                                                                         c
      fnorm=0.0
      DO i=1,n-1
      fnorm=fnorm+fred(i)*fred(i)
      END DO
      fnorm=SQRT(fnorm)
      RETURN
      END
c                                                                         c
c*******************************************************************c
```

```
c    subroutine used to calculate the gradient vector                  c
c*********************************************************************c
c                                                                      c
      SUBROUTINE fredis(n,in,M,Mt,P,C,D,delta,Pcoa,fred)
c                                                                      c
      REAL M(in),P(in),D(in,in),C(in,in),delta(n),fred(n),Pcoa,Mt
c                                                                      c
      Pcoa=P(n)
      DO 1 i=1,n-1
        Pcoa=Pcoa +P(i)
        DO 1 j=i+1,n
          Pcoa=Pcoa-2.0*D(i,j)*cos(delta(i)-delta(j))
1     CONTINUE
c                                                                      c
      DO 3 i=1,n
        Pei=0.0
        DO 2 j=1,n
          IF(j.ne.i) THEN
            deltaij=delta(i)-delta(j)
            Pei=Pei+C(i,j)*sin(deltaij)+D(i,j)*cos(deltaij)
          END IF
2       CONTINUE
        fred(i)=P(i)-Pei-M(i)/Mt*Pcoa
3     CONTINUE
c                                                                      c
      RETURN
      END
c                                                                      c
c*********************************************************************c
c    subroutine which determine the potentiel energy                   c
c*********************************************************************c
c                                                                      c
      SUBROUTINE pot(n,in,P,delta,deltas,C,D,Vpe)
c                                                                      c
REAL P(n),delta(n),deltas(n),C(in,n),D(in,n),Vpe
c                                                                      c
      Vp1=-P(n)*(delta(n)-deltas(n))
      Vp2=0.0
      Vd=0.0
c                                                                      c
      DO 1 i=1,n-1
      Vp1=Vp1-P(i)*(delta(i)-deltas(i))
      DO 1 j=i+1,n
```

```
c                                                                       c
        deltaij=delta(i)-delta(j)
        deltaijs=deltas(i)-deltas(j)
        Vp2=Vp2-C(i,j)*( cos(deltaij)-cos(deltaijs) )
c                                                                       c
      IF(deltaij.ne.deltaijs) THEN
        Dijp = ( delta(i) + delta(j) - (deltas(i) + deltas(j)) ) /
     &             ( deltaij-deltaijs ) * D(i,j)
      ELSE
        Dijp=D(i,j)
      ENDIF
        Vd = Vd+ Dijp * ( sin(deltaij) - sin(deltaijs) )
        Vpe = Vp1+Vp2+ Vd
1     CONTINUE
      RETURN
      END
c                                                                       c
c***********************************************************************c
c      Subroutine which estimates the angles whith a quadratique.       c
c        We use the least-squares estimator((x'x)-1x'=B)                c
c***********************************************************************c
c                                                                       c
      SUBROUTINE estim1(n,a,j,xe,kita)
c                                                                       c
      INTEGER i,j,k,n,r,col1,a,u
c                                                                       c
      REAL  mtu(240,5),mtutr(5,240),xe(240,20),ID(5,5),
     +       Cm(5,5),CI(5,5),b(5),dm(5)
      REAL  Y(240),kita(5,9)
c                                                                       c
c***********************************************************************c
c          loop for each delta (n-1 values)                             c
c***********************************************************************c
c                                                                       c
      DO k=1,3
        B(k)=0
        Dm(k)=0
      END DO
      DO 300  i=1,n-1
c                                                                       c
        DO k=1,a-2
          mtu(k,1)=1
          mtu(k,2)=xe(j-a+k+1,i)
```

```
          mtu(k,3)=xe(j-a+k,i)
          Y(k)=xe(j-a+k+2,i)
        END DO
c                                                                       c
c***************Get the transpose matrix of mtu                        c
c                                                                       c
        DO k=1,a-2
          mtutr(1,k)=1
          mtutr(2,k)=xe(j-a+k+1,i)
          mtutr(3,k)=xe(j-a+k,i)
        END DO
        col1=a-2
        do u=1,3
          do r=1,3
            Cm(u,r)=0.0
            do k=1,col1
              Cm(u,r)=Cm(u,r)+mtutr(u,k)*mtu(k,r)
            end do
          end do
        end do
        CALL  frc28 (Cm,CI,3)
        CALL multi(3,col1,1,mtutr,Y,B)
        col1=3
        CALL multi(3,col1,1,CI,B,Dm)
        col1=a-2
        DO  k=1,3
           kita(k,i)=Dm(k)
        END DO
300   CONTINUE
      RETURN
      END
c                                                                       c
c***********************************************************************c
c       Subroutine which multiply two matrix                            c
c***********************************************************************c
c                                                                       c
      SUBROUTINE multi(lig,col13,col23,A,B,C)
c                                                                       c
INTEGER i,j,k,lig,col13,col23
      REAL  A(5,80),B(80,5),C(5,5)
c                                                                       c
      DO 1 i=1,lig
       DO 2 j=1,col23
```

```
        C(i,j)=0
        DO 3 k=1,col13
          C(i,j)=C(i,j)+A(i,k)*B(k,j)
3       CONTINUE
2     CONTINUE
1    CONTINUE
c                                                                       c
     RETURN
     END
c                                                                       c
c***********************************************************************
c     Subroutine which invert a matrix                                  c
c***********************************************************************
c******************Numerical method
c                                                                       c
      SUBROUTINE FRC28 (a,ai,n)
c                                                                       c
      PARAMETER (np=5,mp=1,NMAX=100)
c                                                                       c
     INTEGER m,n
     INTEGER i,icol,irow,j,k,l,ll,indxc,indxr,ipiv
c                                                                       c
     REAL  a,ai,c,b
c                                                                       c
     REAL  big,dum,pivinv
c                                                                       c
     DIMENSION indxc(NMAX),indxr(NMAX),ipiv(NMAX)
     DIMENSION a(np,np),ai(np,np),c(np,np),b(np,mp)
c                                                                       c
     m=1
     DO j=1,n
       b(j,m)=1.0
       ipiv(j)=0.0
     END DO
c                                                                       c
     DO i=1,n
       DO j=1,n
         c(i,j)=a(i,j)
       END DO
     END DO
c                                                                       c
     DO 22 i=1,n
      big=0.0
```

```
      DO 13 j=1,n
        IF (ipiv(j).ne.1.0) THEN
          DO 12 k=1,n
            IF (ipiv(k).eq.0.0) THEN
              IF (abs(a(j,k)).ge.big) THEN
                big=abs(a(j,k))
                irow=j
                icol=k
              ENDIF
            ELSE IF (ipiv(k).gt.1.0) then
              PAUSE 'singular matrix in gaussj'
            ENDIF
12          CONTINUE
        ENDIF
13      CONTINUE
      ipiv(icol)=ipiv(icol)+1.0
      IF (irow.ne.icol) THEN
        DO 14 l=1,n
          dum=a(irow,l)
          a(irow,l)=a(icol,l)
          a(icol,l)=dum
14        CONTINUE
        DO 15 l=1,m
          dum=b(irow,l)
          b(irow,l)=b(icol,l)
          b(icol,l)=dum
15        CONTINUE
      ENDIF
      indxr(i)=irow
      indxc(i)=icol
      IF (a(icol,icol).eq.0.0) pause 'singular matrix in gaussj'
      pivinv=1.0/a(icol,icol)
      a(icol,icol)=1.0
      DO 16 l=1,n
        a(icol,l)=a(icol,l)*pivinv
16      CONTINUE
      DO 17 l=1,m
        b(icol,l)=b(icol,l)*pivinv
17      CONTINUE
      DO 21 ll=1,n
        IF(ll.ne.icol)THEN
          dum=a(ll,icol)
          a(ll,icol)=0.0
```

```
          DO 18 l=1,n
            a(ll,l)=a(ll,l)-a(icol,l)*dum
18          CONTINUE
          DO 19 l=1,m
            b(ll,l)=b(ll,l)-b(icol,l)*dum
19          CONTINUE
        ENDIF
21     CONTINUE
22   CONTINUE
c                                                                       c
    DO 24 l=n,1,-1
      IF(indxr(l).ne.indxc(l))THEN
        DO 23 k=1,n
          dum=a(k,indxr(l))
          a(k,indxr(l))=a(k,indxc(l))
          a(k,indxc(l))=dum
23        CONTINUE
      ENDIF
24   CONTINUE
c                                                                       c
    DO i=1,n
      DO j=1,n
        ai(i,j)=a(i,j)
        a(i,j)=c(i,j)
      END DO
    END DO
c                                                                       c
    RETURN
    END
c                                                                       c
c***********************************************************************c
c     Subroutine to solve a system of differential equations            c
c***********************************************************************c
c                                                                       c
c                                                                       c
    SUBROUTINE rksyst(n,in,mm,h,x,xend,xwrk,C,D,M,P,Mt,f)
c                                                                       c
    REAL M(n), P(n), D(in,n), C(in,n), Pcoa, Mt
c                                                                       c
    DIMENSION xend(mm),xwrk(4,mm), x(mm), f(mm)
c                                                                       c
    CALL fis(n,in,M,Mt,P,C,D,x,Pcoa,f,gradEp)
c                                                                       c
```

```
      DO 10 i=1,mm
          xwrk(1,i)=h*f(i)
          xend(i)=x(i)+xwrk(1,i)/2.0
10    CONTINUE
c                                                                       c
      CALL fis(n,in,M,Mt,P,C,D,xend,Pcoa,f,gradEp)
c                                                                       c
      DO 20 jj=1,mm
          xwrk(2,jj)=h*f(jj)
          xend(jj)=x(jj)+xwrk(2,jj)/2.0
20    CONTINUE
c                                                                       c
      CALL fis(n,in,M,Mt,P,C,D,xend,Pcoa,f,gradEp)
c                                                                       c
      DO 30 k=1,mm
          xwrk(3,k)=h*f(k)
          xend(k)=x(k)+xwrk(3,k)
30    CONTINUE
c                                                                       c
      CALL fis(n,in,M,Mt,P,C,D,xend,Pcoa,f,gradEp)
c                                                                       c
      DO 40 l=1,mm
          xwrk(4,l)=h*f(l)
40    CONTINUE
c                                                                       c
      DO 50 i=1,mm
        xend(i)=x(i)+(xwrk(1,i)+2.0*xwrk(2,i)
     &          +2.0*xwrk(3,i)+xwrk(4,i))/6.
50    CONTINUE
c                                                                       c
      RETURN
      END
c                                                                       c
c***********************************************************************c
c       Subroutine which gives the state equation(d(delta)/dt)          c
c            & the directional derivative(gradEp.v)                     c
c***********************************************************************c
c                                                                       c
      SUBROUTINE fis(n,in,M,Mt,P,C,D,x,Pcoa,f,gradEp)
c                                                                       c
      DIMENSION f(240), x(240)
      REAL M(n),P(n),D(in,n),C(in,n),Pcoa,Mt,gradEp,dn
c                                                                       c
```

```
      Pcoa = P(n)
      dn=0.0
      gradEp=0.0
c                                                                       c
      DO i=1,n-1
        Pcoa = Pcoa + P(i)
        dn=dn-M(i)*x(i)/M(n)
      END DO
c                                                                       c
      DO 1 i=1,n-2
          DO 1 j=i+1,n-1
              Pcoa = Pcoa - 2.0*D(i,j)*cos(x(i)-x(j))
1     CONTINUE
c                                                                       c
      DO i=1,n-1
        Pcoa=Pcoa-2.0*D(i,n)*cos(x(i)-dn)
      END DO
c                                                                       c
      DO 3 i=1,n-1
          ii=i+n-1
          Pei=0.0
          DO 2 j=1,n-1
              IF(j.ne.i) THEN
                Pei=Pei+C(i,j)*sin(x(i)-x(j))+D(i,j)*cos(x(i)-x(j))
              END IF
2         CONTINUE
          Pei=Pei+C(i,n)*sin(x(i)-dn)+D(i,n)*cos(x(i)-dn)
          f(i)=x(ii)
          f(ii)=(P(i)-Pei)/M(i)-Pcoa/Mt
          gradEp=gradEp-M(i)*f(ii)*(f(i))
3     CONTINUE
c                                                                       c
      wn=0
      Do i=1,n-1
        wn=wn-M(i)*f(i)/M(n)
      END DO
c                                                                       c
      Pen=0
      DO  j=1,n-1
         Pen=Pen+C(n,j)*sin(dn-x(j))+D(n,j)*cos(dn-x(j))
      END DO
      Pen=Pen+D(n,n)
c                                                                       c
```

```
      gradEp=gradEp-(P(n)-Pen-M(n)*Pcoa/Mt)*(wn)
10    CONTINUE
      RETURN
      END
c                                                                       c
c***********************************************************************c
c          Subroutine which gives the potential,kinetic and total       c
c               energy of a system                                      c
c***********************************************************************c
c                                                                       c
      SUBROUTINE Energy(n,in,in2,M,P,x,deltas,C,D,Vpe,Vke,Vp)
c                                                                       c
c                                                                       c
      REAL P(n),x(in2),deltas(n),C(in,n),D(in,n),Vpe,Vke,M(n)
c***************** x2n => wn                                           c
       x2n = 0.
       Vke = 0.
c                                                                       c
       DO jj =1,n-1
          Vke = Vke + M(jj) * x(jj+n-1)**2
          x2n = x2n - m(jj) * x(jj+n-1)
       END DO
c                                                                       c
       x2n = x2n / M(n)
       Vke = Vke + M(n) * x2n**2
       Vke = Vke * 0.5
c************************ xn => delta(n)                                 c
      xn = 0.
      DO jj=1,n-1
         xn = xn - M(jj) * x(jj)
      END DO
      xn = xn / M(n)
c************************potential energy                               c
c                                                                       c
      Vp1=-P(n)*(xn-deltas(n))
      Vp2=0.0
      Vd=0.0
c                                                                       c
      DO  i=1,n-1
         Vp1=Vp1-P(i)*(x(i)-deltas(i))
         deltain = x(i) - xn
         deltains = deltas(i)-deltas(n)
         Vp2=Vp2 - C(i,n) * ( cos(deltain) - cos(deltains) )
```

```
c                                                                       c
        IF(deltain.ne.deltains) THEN
          Dinp = D(i,n) * ( x(i) + xn - (deltas(i) + deltas(n)) ) /
     &                ( deltain - deltains )
                            ELSE
                            Dinp = D(i,n)
        END IF
c                                                                       c
        Vd = Vd  + Dinp  * ( sin(deltain) - sin(deltains) )
      END DO
c                                                                       c
      DO i = 1,n-2
        DO j=i+1,n-1
          deltaij=x(i)-x(j)
          deltaijs=deltas(i)-deltas(j)
          Vp2=Vp2-C(i,j)*( cos(deltaij)-cos(deltaijs) )
c                                                                       c
          IF(deltaij.ne.deltaijs) THEN
            Dijp=(x(i)+x(j)-(deltas(i)+deltas(j)))/
     &                    (deltaij-deltaijs)*D(i,j)
                             ELSE
                             Dijp=D(i,j)
          END IF
c                                                                       c
          Vd = Vd+ Dijp * ( sin(deltaij) - sin(deltaijs) )
          Vp = Vp1 + Vp2
          Vpe = Vp + Vd
        END DO
      END DO
c                                                                       c
      RETURN
      END
c                                                                       c
c***********************************************************************c
c                                    END                                c
c***********************************************************************c
```