Supplementary material to the work: "Brownian Bridge for Coherent State Path Integral Monte Carlo"

Riccardo Fantoni*
Università di Trieste, Dipartimento di Fisica, strada Costiera 11, 34151 Grignano (Trieste), Italy
(Dated: April 27, 2026)

Supplementary material to the article "Brownian Bridge for Coherent State Path Integral Monte Carlo".

Keywords: Quantum Many Body; Coherent States; Path Integral; Quantum Monte Carlo; Brownian Bridge; Helium; Thermodynamics

I. INTRODUCTION

We give in the Appendix the code listing for the CSPIMC simulations carried out in the work: "Brownian Bridge for Coherent State Path Integral Monte Carlo". We will denote this paper with [*].

Subroutine `cs()` calculates the high temperature reduced density matrix of Eq. (3.7) of paper [*]. Subroutine `bridge()` performs the Brownian bridge of Eqs. (4.1)-(4.2) of the paper [*] and subroutine `acccs()` calculates the "a priori" sampling distribution of Eq. (4.6) of the paper [*].

Appendix A: The code

This is the code used for the CSPIMC computer experiment. We list here the main FORTRAN code `cspimc.f` with its included `mc-cs.par` parameters file. And the input data file `data-cs-new.in` that is read at the beginning of the run.

```
------------------------------------------------
       cspimc.f
------------------------------------------------
       PROGRAM CoStPIMC
       IMPLICIT NONE
C     ****************************************************************
C     ** MONTE CARLO SIMULATION PATH INTEGRAL WITH COHERENT STATES    **
C     **                                                             **
C     ** space dimensions:    DIM=1,2,3                              **
C     ** number of particles: NP                                     **
C     ** number of timesteps: FTN0                                   **
C     ** units:               hbar=k_B=1                             **
C     ** BETA=1/TEMP=FTN0*LS  (LS imaginary time spacing)            **
C     ** Harmonic Oscillator  (HO) for Coherent States  (CS):        **
C     ** MOHO = m_ho*omega                                           **
C     **                                                             **
C     ** OBSERVABLES:                                               **
C     ** kinetic energy   KE                                        **
C     ** potential energy V                                         **
C     ****************************************************************
       INCLUDE       'mc-cs.par'                          ! parameters

       INTEGER*4     SEED
       INTEGER*8     IDIM, STEP, I, J, K
       INTEGER*8     II(MNP), JJ, KK, PREV(MNP)
       INTEGER*8     C1, CP, IP, KP, NIP, NKP, LL, OUT1, OUT2, PIP
       INTEGER*8     INIT, NSTEP, IPRINT, ISAVE, IRATIO, IEQUI
       INTEGER*8     MBMM, MCM

       REAL*8        DENS, TEMP, BETA, DENSLJ
       REAL*8        DRMAX, ALPHA, CDIM
       REAL*8        RANF, DUMMY, SR9, SR3
       REAL*8        VLRC, VLRC6, VLRC12, WLRC, WLRC6, WLRC12
       REAL*8        ACM, ACATMA, ACATMAS, ACATMAB
       REAL*8        V, VNEW, VOLD, VEND, DELTV, VN, VS
       REAL*8        W, WNEW, WOLD, WEND, DELTW, WN, WS
       REAL*8        KE, KENEW, KEOLD, DELTKE, KEEND
       REAL*8        AVV, ACV, ACVSQ, FLV
       REAL*8        AVKE, ACKE, ACKESQ, FLKE
       REAL*8        ACT, ACTNEW, ACTOLD, DELTACT, DELTACTB
       REAL*8        RXIOLD(MDIM), RXINEW(MDIM)
       REAL*8        RAOLD(MDIM), PAOLD(MDIM)
       REAL*8        RBOLD(MDIM), PBOLD(MDIM)
       REAL*8        RANEW(MDIM), PANEW(MDIM)
       REAL*8        RBNEW(MDIM), PBNEW(MDIM)
       REAL*8        RXP(MDIM,0:N), RXPP(MDIM,0:N)
       REAL*8        RAP(MDIM,0:N), RAPP(MDIM,0:N)
       REAL*8        RBP(MDIM,0:N), RBPP(MDIM,0:N)
       REAL*8        STD, PS, KKK, VVV, WWW
       REAL*8        RATIO, RATIOS, RATIOB

       COMPLEX*16    ZNEW, ZOLD, ZN, ZO

       CHARACTER     CNFILE*30, POTK*10

       LOGICAL       OVRLAP, IFB, IFDISP, IFBRIDGE, IFZERO

C      PI = ACOS(-1.d0)

C     ****************************************************************
C     ** READ INPUT DATA                                            **
C     ****************************************************************

       WRITE(*,'('' ********** PROGRAM PIMC ************      ''/)')
       WRITE(*,'('' COHERENT STATES                          ''/)')
       WRITE(*,'('' PATH INTEGRAL MONTE CARLO PROGRAM        ''/)')
       OPEN (UNIT=10, FILE='data-cs-new.in', STATUS='UNKNOWN')
       READ (10,*) I
       READ (10,*) DIM
       READ (10,*) I
       READ (10,*) SEED
       READ (10,*) I
       READ (10,*) IFB
       READ (10,*) I
       READ (10,*) NP
       READ (10,*) I
       READ (10,'(A)') POTK
       READ (10,*) I
       READ (10,*) FTN0
       READ (10,*) I
       READ (10,*) FTM
       READ (10,*) I
       READ (10,*) NSTEP
       READ (10,*) I
       READ (10,*) IPRINT
       READ (10,*) I
       READ (10,*) ISAVE
       READ (10,*) I
       READ (10,*) IEQUI
       READ (10,*) I
       READ (10,*) IRATIO
       READ (10,*) I
       READ (10,'(A)') CNFILE
       READ (10,*) I
       READ (10,*) INIT
       READ (10,*) I
       READ (10,*) DENS
       READ (10,*) I
       READ (10,*) TEMP
       READ (10,*) I
       READ (10,*) ALPHA
       READ (10,*) I
       READ (10,*) MBMM
       READ (10,*) I
       READ (10,*) RCUT
       READ (10,*) I
       READ (10,*) SIG
       READ (10,*) I
       READ (10,*) EPSR
       READ (10,*) I
       READ (10,*) EPSA
       READ (10,*) I
       READ (10,*) EPS
       READ (10,*) I
       READ (10,*) MOHO
       READ (10,*) I
       READ (10,*) IFDISP
       READ (10,*) I
       READ (10,*) IFBRIDGE
       READ (10,*) I
       READ (10,*) IFZERO
       CLOSE (UNIT=10)
       WRITE(*,'('' IN DIMENSION (1,2,3,...)''I12/)') DIM
       WRITE(*,'('' ********************************      ''/)')

       GOTO 1111                  ! comment if input from keyboard
```

* riccardo.fantoni@scuola.istruzione.it

```
      WRITE(*,'('' ********** PROGRAM PIMC ************      ''/)')
      WRITE(*,'('' COHERENT STATES                          '')')
      WRITE(*,'('' PATH INTEGRAL MONTE CARLO PROGRAM        '')')

      WRITE(*,'('' ENTER THE SPATIAL DIMENSIONS             '')')
      READ (*,*) DIM
      WRITE(*,'('' ENTER SEED FOR RANDOM SEQUENCE           '')')
      READ (*,*) SEED
      WRITE(*,'('' IF BOSE (T/F)                            '')')
      READ (*,*) IFB
      WRITE(*,'('' ENTER THE NUMBER OF PARTICLES < MNP      '')')
      READ (*,*) NP
      WRITE(*,'('' ENTER TYPE OF PAIR POTENTIAL (Many Body) '')')
      READ (*,'(A)') POTK
      WRITE(*,'('' ENTER THE NUMBER OF DISCRETIZATIONS < N  '')')
      READ (*,*) FTNO
      WRITE(*,'('' ENTER THE BARE MASS                      '')')
      READ (*,*) FTM
      WRITE(*,'('' ENTER NUMBER OF CYCLES                   '')')
      READ (*,*) NSTEP
      WRITE(*,'('' ENTER NUMBER OF STEPS BETWEEN OUTPUT LINES '')')
      READ (*,*) IPRINT
      WRITE(*,'('' ENTER NUMBER OF STEPS BETWEEN DATA SAVES '')')
      READ (*,*) ISAVE
      WRITE(*,'('' ENTER NUMBER OF STEPS FOR EQUILIBRATION  '')')
      READ (*,*) IEQUI
      WRITE(*,'('' ENTER INTERVAL FOR UPDATE OF MAX. DISPL. '')')
      READ (*,*) IRATIO
      WRITE(*,'('' ENTER THE CONFIGURATION FILE NAME        '')')
      READ (*,'(A)') CNFILE
      WRITE(*,'('' ENTER 0 IF INITIALIZATION NEEDED         '')')
      READ (*,*) INIT
      WRITE(*,'('' ENTER THE DENSITY                        '')')
      READ (*,*) DENS
      WRITE(*,'('' ENTER THE TEMPERATURE                    '')')
      READ (*,*) TEMP
      WRITE(*,'('' ENTER MAX. DISPLACEMENT DRMAX/SIGMA      '')')
      READ (*,*) ALPHA
      WRITE(*,'('' ENTER MAXIMUM NUMBER OF BRIDGE TIMESLICES '')')
      READ (*,*) MBMM
      WRITE(*,'('' ENTER THE POTENTIAL CUTOFF DISTANCE (~ 2.5)'')')
      READ (*,*) RCUT
      WRITE(*,'('' ENTER SIG                                '')')
      READ (*,*) SIG
      WRITE(*,'('' ENTER EPSR                               '')')
      READ (*,*) EPSR
      WRITE(*,'('' ENTER EPSA                               '')')
      READ (*,*) EPSA
      WRITE(*,'('' ENTER EPS                                '')')
      READ (*,*) EPS
      WRITE(*,'('' ENTER HO MASS*OMEGA=SQRT(MASS*k)         '')')
      READ (*,*) MOHO
      WRITE(*,'('' IF DISPLACEMENT MOVE (T/F)               '')')
      READ (*,*) IFDISP
      WRITE(*,'('' IF BRIDGE MOVE (T/F)                     '')')
      READ (*,*) IFBRIDGE
      WRITE(*,'('' IF ZERO PATH INITIALLY (T/F)             '')')
      READ (*,*) IFZERO
      WRITE(*,'('' IN DIMENSION (1,2,3,...)''I12/)') DIM
      WRITE(*,'('' ************************************      ''/)')

 1111 CONTINUE

      IF (MBMM .GT. FTNO) THEN
         WRITE(*,*) "number of timeslices in bridge > ",FTNO
         STOP
      ENDIF
      RX=0.                  ! real paths on the origin
      RA=0.                  ! ghost paths
      RB=0.                  ! ghost paths
      PA=0.                  ! ghost paths
      PB=0.                  ! ghost paths

C     ** INITIALIZE RANDOM NUMBER GENERATOR **

      CALL SRAND ( SEED )
      IF (SEED.EQ.1) THEN
         CALL SRAND ( 0 )
      ENDIF

C     ** INVERSE TEMPERATURE BETA **

      BETA = (1.d0/TEMP)

C     ** IMAGINARY TIMESTEP **

      LS  = BETA/FTNO      ! time step
      STD = (LS/FTM)**0.5  ! standatd deviation of free-particle rho

C    ** WRITE INPUT DATA **

      WRITE(*,'('' SEED                     '',I10   )') SEED
      WRITE(*,'('' POTENTIAL                '',A     )') POTK
      WRITE(*,'('' DIMENSIONS               '',I10   )') DIM
      WRITE(*,'('' NUMBER OF PARTICLES      '',I10   )') NP
      WRITE(*,'('' NUMBER OF CYCLES         '',I10   )') NSTEP
      WRITE(*,'('' NUMBER OF EQUIL. STEPS   '',I10   )') IEQUI
      WRITE(*,'('' OUTPUT FREQUENCY         '',I10   )') IPRINT
      WRITE(*,'('' SAVE FREQUENCY           '',I10   )') ISAVE
      WRITE(*,'('' RATIO UPDATE FREQUENCY   '',I10   )') IRATIO
      WRITE(*,'('' CONFIGURATION FILE  NAME '',A     )') CNFILE
      WRITE(*,'('' TEMPERATURE              '',E10.4 )') TEMP
      WRITE(*,'('' DENSITY                  '',E10.4 )') DENS
      WRITE(*,'('' PARTICLE MASS            '',E10.4 )') FTM
      WRITE(*,'('' HO MASS*OMEGA            '',E10.4 )') MOHO
      WRITE(*,'('' # TIME SLICES            '',I10   )') FTNO
      WRITE(*,'('' TIME STEP                '',E10.4 )') LS

C    ** CONVERT INPUT DATA TO PROGRAM UNITS **

      SIGMA  = ( DBLE ( NP ) / DENS ) ** ( 1.0 / DIM )
      DRMAX = ALPHA * SIGMA

      RATIO  = 0.0
      RATIOS = 0.0
      OUT1   = 0
      OUT2   = 0

      IF (POTK .EQ. 'LJ') THEN
         DENSLJ = DENS  * SIG ** DBLE( DIM )
C         RCUT = SIGMA/2.d0/SIG
      ENDIF

C     ** INITIALIZE CONFIGURATION **

      DO I = 1, NP
         NEXT(I) = I
         LEXT(I) = I
         DO J = 1, FTNO
            MEXT(J,I)  = I
         ENDDO
      ENDDO

      IF ( INIT .EQ. 0 ) THEN
         PRINT*,"PATHS INITIALIZATION ..............."
         CALL INITCN ( CNFILE )          ! random configuration
         CALL READCN ( CNFILE )          ! random configuration

         IF (IFZERO) THEN
            RX=0.             ! real paths on the origin
            RA=0.             ! ghost paths
            RB=0.             ! ghost paths
            PA=0.             ! ghost paths
            PB=0.             ! ghost paths
         ENDIF
      ENDIF

C     ** READ INITIAL CONFIGURATION **

      IF ( INIT .NE. 0 ) THEN
         CALL READCN ( CNFILE )
      ENDIF

C     ** ZERO ACCUMULATORS **

      ACV     = 0.0
      ACVSQ   = 0.0
      FLV     = 0.0
      ACKE    = 0.0
      ACKESQ  = 0.0
      FLKE    = 0.0
      ACM     = 0.0
      ACATMA  = 0.0
      ACATMAB = 0.0
      ACATMAS = 0.0

C     ** CALCULATE LONG RANGE CORRECTIONS    **
C     ** SPECIFIC TO THE LENNARD JONES FLUID **
      IF (DIM .EQ. 1) CDIM = 1
      IF (DIM .EQ. 2) CDIM = PI
      IF (DIM .EQ. 3) CDIM = 2*PI

      SR3 = - RCUT ** (- 6. + DIM)/(-6. + DIM)
      SR9 = - RCUT ** (- 12. + DIM)/(-12. + DIM)

      VLRC12 =   4 * EPS * CDIM * DENSLJ * NP * SR9
      VLRC6  = - 4 * EPS * CDIM * DENSLJ * NP * SR3
      VLRC   =   VLRC12 + VLRC6
      WLRC12 =   4.0  * VLRC12
      WLRC6  =   2.0  * VLRC6
      WLRC   =   WLRC12 + WLRC6

C    ** WRITE OUT SOME USEFUL INFORMATION **

      WRITE(*,'('' SIGMA                    '',E10.4)') SIGMA
      WRITE(*,'('' MAXIMUM DISPLACEMENT     '',E10.4)') DRMAX

C    ** CALCULATE INITIAL ENERGY AND CHECK FOR OVERLAPS **

      CALL CSSUMUP (POTK, OVRLAP, KE, V)

      IF (POTK .EQ. 'LJ') THEN
         VS = ( V + VLRC )
         WS = ( W + WLRC )
      ELSE
         VS = V
      ENDIF

      WRITE(*,'('' INITIAL V                '',E10.4)') VS
      WRITE(*,'('' INITIAL KE               '',E10.4)') KE

      WRITE(*,'(//'' START OF MARKOV CHAIN              ''//)')
      WRITE(*,'(''  NMOVE     RATIO       ACTION          ''/)')

C     ***************************************************************
C     ** LOOPS OVER ALL CYCLES AND ALL TIME SLICES                  **
C     ***************************************************************

      DO 100 STEP = 1, NSTEP

         CP = INT(FTNO*RANF(DUMMY))+1    ! select a random timeslice
         MBM = INT((MBMM-1)*RANF(DUMMY))+2  ! # timeslices in bridge
         IP = INT(NP*RANF(DUMMY))+1      ! select a particle
         KP = INT(NP*RANF(DUMMY))+1      ! select another particle
         IF (.NOT.IFB) KP = IP           ! for boltzmann statistics
         IF (IFDISP) GOTO 77

C     ***************************************************************
C     ** BRIDGE&SWAP MOVE (BOSE STATISTICS)                         **
C     ***************************************************************
         mcm = cp+mbm-floor((cp+mbm-.1)/ftn0)*ftn0
```

```
      nip=next(ip)
      nkp=next(kp)

      vold=0.d0
      if(cp+mbm.le.ftn0)then
         do i=cp+1,mcm-1
            call ppnergy ( potk, rx(:,i,ip), ip,
     :           rx(:,i,kp), kp, i,
     :           vvv, www )
            vold=vold+vvv
            wold=wold+www
         enddo
      else
         do i=cp+1,ftn0
            call ppnergy ( potk, rx(:,i,ip), ip,
     :           rx(:,i,kp), kp, i,
     :           vvv, www )
            vold=vold+vvv
            wold=wold+www
         enddo
         do i=1,mcm-1
            call ppnergy ( potk, rx(:,i,nip), nip,
     :           rx(:,i,nkp), nkp, i,
     :           vvv, www )
            vold=vold+vvv
            wold=wold+www
         enddo
      endif

      keold=0.d0
      do k=1,np
         if(k.eq.ip.or.k.eq.kp.or.k.eq.nip.or.k.eq.nkp)then
            do i=1,ftn0
               call cskkknergy ( k, i, kkk )
               keold=keold+kkk
            enddo
         endif
      enddo

      if(cp+mbm.le.ftn0)then
         call bridge(rx(:,cp,ip),rx(:,mcm,kp),std,
     :        rxp,out1)
         call bridge(ra(:,cp,ip),ra(:,mcm,kp),std,
     :        rap,out1)
         call bridge(rb(:,cp,ip),rb(:,mcm,kp),std,
     :        rbp,out1)
         if (ip.ne.kp) then
            call bridge(rx(:,cp,kp),rx(:,mcm,ip),std,
     :           rxpp,out2)
            call bridge(ra(:,cp,kp),ra(:,mcm,ip),std,
     :           rapp,out2)
            call bridge(rb(:,cp,kp),rb(:,mcm,ip),std,
     :           rbpp,out2)
         endif
      else
         call bridge(rx(:,cp,ip),rx(:,mcm,nkp),std,
     :        rxp,out1)
         call bridge(ra(:,cp,ip),ra(:,mcm,nkp),std,
     :        rap,out1)
         call bridge(rb(:,cp,ip),rb(:,mcm,nkp),std,
     :        rbp,out1)
         if (ip.ne.kp) then
            call bridge(rx(:,cp,kp),rx(:,mcm,nip),std,
     :           rxpp,out2)
            call bridge(ra(:,cp,kp),ra(:,mcm,nip),std,
     :           rapp,out2)
            call bridge(rb(:,cp,kp),rb(:,mcm,nip),std,
     :           rbpp,out2)
         endif
      endif

      vnew=0.d0
      wnew=0.d0
      ll=0
      if(cp+mbm.le.ftn0)then
         do i=cp+1,mcm-1
            ll=ll+1
            if (ip.eq.kp) then
               call ppnergy ( potk, rxp(:,ll), ip,
     :              rxp(:,ll), kp, i,
     :              vvv, www )
            else
               call ppnergy ( potk, rxp(:,ll), ip,
     :              rxpp(:,ll), kp, i,
     :              vvv, www )
            endif
            vnew=vnew+vvv
            wnew=wnew+www
         enddo
      else
         do i=cp+1,ftn0
            ll=ll+1
            if (ip.eq.kp) then
               call ppnergy ( potk, rxp(:,ll), ip,
     :              rxp(:,ll), kp, i,
     :              vvv, www )
            else
               call ppnergy ( potk, rxp(:,ll), ip,
     :              rxpp(:,ll), kp, i,
     :              vvv, www )
            endif
            vnew=vnew+vvv
            wnew=wnew+www
         enddo
         do i=1,mcm-1
            ll=ll+1
            if (ip.eq.kp) then
               call ppnergy ( potk, rxp(:,ll), nip,
     :              rxp(:,ll), nkp, i,
     :              vvv, www )
            else
               call ppnergy ( potk, rxpp(:,ll), nip,
     :              rxp(:,ll), nkp, i,
     :              vvv, www )
            endif
            vnew=vnew+vvv
            wnew=wnew+www
         enddo
      endif

      zold=1.d0
      if(cp+mbm.le.ftn0)then
      do c1=cp+1,mcm-1
         call cs(ra(:,c1,ip),pa(:,c1,ip),rb(:,c1,ip)
     :        ,pb(:,c1,ip),rx(:,c1,ip),rx(:,c1+1,ip),zo)
         zold=zold*zo
         call cs(ra(:,c1-1,ip),pa(:,c1-1,ip),rb(:,c1-1,ip)
     :        ,pb(:,c1-1,ip),rx(:,c1-1,ip),rx(:,c1,ip),zo)
         zold=zold*zo
      enddo
      else
      do c1=cp+1,ftn0
         call cs(ra(:,c1,ip),pa(:,c1,ip),rb(:,c1,ip)
     :        ,pb(:,c1,ip),rx(:,c1,ip),rx(:,c1+1,ip),zo)
         zold=zold*zo
         call cs(ra(:,c1-1,ip),pa(:,c1-1,ip),rb(:,c1-1,ip)
     :        ,pb(:,c1-1,ip),rx(:,c1-1,ip),rx(:,c1,ip),zo)
         zold=zold*zo
      enddo
      do c1=1,mcm-1
         call cs(ra(:,c1,nkp),pa(:,c1,nkp),rb(:,c1,nkp)
     :        ,pb(:,c1,nkp),rx(:,c1,nkp),rx(:,c1+1,nkp),zo)
         zold=zold*zo
         call cs(ra(:,c1-1,nkp),pa(:,c1-1,nkp),rb(:,c1-1,nkp)
     :        ,pb(:,c1-1,nkp),rx(:,c1-1,nkp),rx(:,c1,nkp),zo)
         zold=zold*zo
      enddo
      endif
      if (ip.ne.kp) then
         if(cp+mbm.le.ftn0)then
         do c1=cp+1,mcm-1
            call cs(ra(:,c1,kp),pa(:,c1,kp),rb(:,c1,kp)
     :           ,pb(:,c1,kp),rx(:,c1,kp),rx(:,c1+1,kp),zo)
            zold=zold*zo
            call cs(ra(:,c1-1,kp),pa(:,c1-1,kp),rb(:,c1-1,kp)
     :           ,pb(:,c1-1,kp),rx(:,c1-1,kp),rx(:,c1,kp),zo)
            zold=zold*zo
         enddo
         else
         do c1=cp+1,ftn0
            call cs(ra(:,c1,kp),pa(:,c1,kp),rb(:,c1,kp)
     :           ,pb(:,c1,kp),rx(:,c1,kp),rx(:,c1+1,kp),zo)
            zold=zold*zo
            call cs(ra(:,c1-1,kp),pa(:,c1-1,kp),rb(:,c1-1,kp)
     :           ,pb(:,c1-1,kp),rx(:,c1-1,kp),rx(:,c1,kp),zo)
            zold=zold*zo
         enddo
         do c1=1,mcm-1
            call cs(ra(:,c1,nip),pa(:,c1,nip),rb(:,c1,nip)
     :           ,pb(:,c1,nip),rx(:,c1,nip),rx(:,c1+1,nip),zo)
            zold=zold*zo
            call cs(ra(:,c1-1,nip),pa(:,c1-1,nip),rb(:,c1-1,nip)
     :           ,pb(:,c1-1,nip),rx(:,c1-1,nip),rx(:,c1,nip),zo)
            zold=zold*zo
         enddo
         endif
      endif

      znew=1.d0
      ll=0
      if(cp+mbm.le.ftn0)then
      do c1=cp+1,mcm-1
         ll=ll+1
         call cs(rap(:,ll),pa(:,c1,ip),rbp(:,ll)
     :        ,pb(:,c1,ip),rxp(:,ll),rx(:,c1+1,ip),zo)
         znew=znew*zo
         call cs(rap(:,ll),pa(:,c1-1,ip),rbp(:,ll)
     :        ,pb(:,c1-1,ip),rx(:,c1-1,ip),rxp(:,ll),zo)
         znew=znew*zo
      enddo
      else
      do c1=cp+1,ftn0
         ll=ll+1
         call cs(rap(:,ll),pa(:,c1,ip),rbp(:,ll)
     :        ,pb(:,c1,ip),rxp(:,ll),rx(:,c1+1,ip),zo)
         znew=znew*zo
         call cs(rap(:,ll),pa(:,c1-1,ip),rbp(:,ll)
     :        ,pb(:,c1-1,ip),rx(:,c1-1,ip),rxp(:,ll),zo)
         znew=znew*zo
      enddo
      do c1=1,mcm-1
         ll=ll+1
         call cs(rap(:,ll),pa(:,c1,nkp),rbp(:,ll)
     :        ,pb(:,c1,nkp),rxp(:,ll),rx(:,c1+1,nkp),zo)
         znew=znew*zo
         call cs(rap(:,ll),pa(:,c1-1,nkp),rbp(:,ll)
     :        ,pb(:,c1-1,nkp),rx(:,c1-1,nkp),rxp(:,ll),zo)
         znew=znew*zo
      enddo
      endif
      if (ip.ne.kp) then
         ll=0
         if(cp+mbm.le.ftn0)then
         do c1=cp+1,mcm-1
            ll=ll+1
            call cs(rapp(:,ll),pa(:,c1,kp),rbpp(:,ll)
     :           ,pb(:,c1,kp),rxpp(:,ll),rx(:,c1+1,kp),zo)
            znew=znew*zo
            call cs(rapp(:,ll),pa(:,c1-1,kp),rbpp(:,ll)
     :           ,pb(:,c1-1,kp),rx(:,c1-1,kp),rxpp(:,ll),zo)
            znew=znew*zo
         enddo
```

```
               else
               do c1=cp+1,ftn0
                  ll=ll+1
                  call cs(rapp(:,ll),pa(:,c1,kp),rbpp(:,ll)
     :                 ,pb(:,c1,kp),rxpp(:,ll),rx(:,c1+1,kp),zo)
                  znew=znew*zo
                  call cs(rapp(:,ll),pa(:,c1-1,kp),rbpp(:,ll)
     :                 ,pb(:,c1-1,kp),rx(:,c1-1,kp),rxpp(:,ll),zo)
                  znew=znew*zo
               enddo
               do c1=1,mcm-1
                  ll=ll+1
                  call cs(rapp(:,ll),pa(:,c1,nip),rbpp(:,ll)
     :                 ,pb(:,c1,nip),rxpp(:,ll),rx(:,c1+1,nip),zo)
                  znew=znew*zo
                  call cs(rapp(:,ll),pa(:,c1-1,nip),rbpp(:,ll)
     :                 ,pb(:,c1-1,nip),rx(:,c1-1,nip),rxpp(:,ll),zo)
                  znew=znew*zo
               enddo
               endif
            endif

            call acccs(ps,ip,kp,cp,rxp,rxpp,rap,rapp,rbp,rbpp)
            ps=ps*exp(-ls*(vnew-vold))*dble(znew)/dble(zold)

            if (ps.gt.ranf(dummy)) then
               acatmab = acatmab + 1.d0
               call update(cp,rxp,rap,rbp,ip,nkp)
               if (ip.ne.kp) then
                  call update(cp,rxpp,rapp,rbpp,kp,nip)
                  call swap(cp,ip,nip,kp,nkp)
                  call switch(next(ip),next(kp))
                  call switch(mext(cp,ip),mext(cp,kp))
                  acatmas = acatmas + 1.d0
               endif

               kenew=0.d0
               do k=1,np
                  if(k.eq.ip.or.k.eq.kp.or.k.eq.nip.or.k.eq.nkp)then
                     do i=1,ftn0
                     call cskkknergy ( k, i, kkk )
                        kenew=kenew+kkk
                     enddo
                  endif
               enddo

               ke=ke+kenew-keold
               v=v+vnew-vold
               w=w+wnew-wold
            endif

c     build the permutations cycles in lext
            call permcyc()

 7777       continue

            IF (STEP.GT.IEQUI) THEN
               ACM = ACM + 1.0

C     ** CALCULATE INSTANTANEOUS VALUES **

                  IF (POTK .EQ. 'LJ') THEN
                     VN = ( V + VLRC )
                  ELSE
                     VN = V
                  ENDIF

C     ** ACCUMULATE AVERAGES **

               ACV    = ACV    + VN
               ACVSQ  = ACVSQ  + VN*VN
               ACKE   = ACKE   + KE
               ACKESQ = ACKESQ + KE*KE
            ENDIF

            IF (IFBRIDGE) GOTO 97

C     ****************************************************************
C     ** ENDS BRIDGE&SWAP MOVE                                      **
C     ****************************************************************

 77         CONTINUE

C     ****************************************************************
C     ** DISPLACEMENT MOVE                                          **
C     ****************************************************************

C     ** PREVIOUS IP **

            DO I=1,NP
               J=NEXT(I)
               PREV(J)=I
            ENDDO

            NIP = NEXT(IP)
            PIP = PREV(IP)

            DO 90 C1 = 1, FTNO       ! exact needs previous IP
C            DO 90 C1 = 2, FTNO      ! exact
C            DO 90 C1 = 2, FTNO-1    ! exact

               DO IDIM = 1, DIM
                  RXIOLD(IDIM) = RX(IDIM,C1,IP)
                  raold(idim)  = ra(idim,c1,ip)
                  rbold(idim)  = rb(idim,c1,ip)
                  paold(idim)  = pa(idim,c1,ip)
                  pbold(idim)  = pb(idim,c1,ip)
               ENDDO

C     ** CALCULATE THE ENERGY OF I IN THE OLD CONFIGURATION **

               CALL PENERGY ( POTK, RXIOLD, IP, C1,
     :              VOLD, WOLD )

               CALL CSKENERGY ( PAOLD, PBOLD, IP, C1,
     :              KEOLD )

               call cs(raold,paold,rbold,pbold,
     :              rxiold,rx(:,c1+1,ip),zold)
               call cs(raold,paold,rbold,pbold,
     :              rx(:,c1-1,ip),rxiold,zo)
               zold=zold*zo

C     ** INSTANTANEOUS VALUE OF THE ACTION **

               ACTOLD = KEOLD + VOLD

C     ** MOVE I AND PICKUP THE CENTRAL IMAGE **

               DO IDIM = 1, DIM
                  RXINEW(IDIM) = RXIOLD(IDIM) +
     :                 ( 2.0 * RANF ( DUMMY ) - 1.0 )*DRMAX
                  RXINEW(IDIM) = RXINEW(IDIM) -
     :                 DNINT ( RXINEW(IDIM)/SIGMA )*SIGMA
                  ranew(idim) = raold(idim) +
     :                 ( 2.0 * ranf ( dummy ) - 1.0 )*drmax
                  ranew(idim) = ranew(idim) -
     :                 dnint ( ranew(idim)/sigma )*sigma
                  rbnew(idim) = rbold(idim) +
     :                 ( 2.0 * ranf ( dummy ) - 1.0 )*drmax
                  rbnew(idim) = rbnew(idim) -
     :                 dnint ( rbnew(idim)/sigma )*sigma
                  panew(idim) = paold(idim) +
     :                 ( 2.0 * ranf ( dummy ) - 1.0 )*drmax
                  pbnew(idim) = pbold(idim) +
     :                 ( 2.0 * ranf ( dummy ) - 1.0 )*drmax
               ENDDO

C     ** CALCULATE THE ENERGY OF I IN THE NEW CONFIGURATION **

               CALL PENERGY ( POTK, RXINEW, IP, C1,
     :              VNEW, WNEW )

               CALL CSKENERGY ( PANEW, PBNEW, IP, C1,
     :              KENEW )

               call cs(ranew,panew,rbnew,pbnew,
     :              rxinew,rx(:,c1+1,ip),znew)
               call cs(ranew,panew,rbnew,pbnew,
     :              rx(:,c1-1,ip),rxinew,zn)
               znew=znew*zn

C     ** INSTANTANEOUS VALUE OF THE ACTION **

               ACTNEW = KENEW + VNEW

               DELTV  = VNEW  - VOLD
               DELTKE = KENEW - KEOLD

C     ** CHECK FOR ACCEPTANCE **

               DELTACTB = dble(znew)/dble(zold)*exp(-LS*(VNEW-VOLD))

               IF ( DELTACTB .GT. RANF ( DUMMY ) ) THEN
                  V          = V  + DELTV
                  KE         = KE + DELTKE
                  ACATMA = ACATMA + 1.0
                  DO IDIM = 1, DIM
                     RX(IDIM,C1,IP)  = RXINEW(IDIM)
                     ra(idim,c1,ip)  = ranew(idim)
                     rb(idim,c1,ip)  = rbnew(idim)
                     pa(idim,c1,ip)  = panew(idim)
                     pb(idim,c1,ip)  = pbnew(idim)
c     imaginary time periodic boundary conditions
                     IF (C1.EQ.1) THEN
                        RX(IDIM,FTNO+1,PIP)=RX(IDIM,C1,IP)
                        RA(IDIM,FTNO+1,PIP)=RA(IDIM,C1,IP)
                        RB(IDIM,FTNO+1,PIP)=RB(IDIM,C1,IP)
                     ENDIF
                     IF (C1.EQ.FTNO) THEN
                        RX(IDIM,0,NIP)=RX(IDIM,C1,IP)
                        RA(IDIM,0,NIP)=RA(IDIM,C1,IP)
                        RB(IDIM,0,NIP)=RB(IDIM,C1,IP)
                     ENDIF
                  ENDDO
               ENDIF

               IF (STEP.GT.IEQUI) THEN
                  ACM = ACM + 1.0

C     ** CALCULATE INSTANTANEOUS VALUES **

                  IF (POTK .EQ. 'LJ') THEN
                     VN = ( V + VLRC )
                  ELSE
                     VN = V
                  ENDIF

C     ** ACCUMULATE AVERAGES **

               ACV    = ACV    + VN
               ACVSQ  = ACVSQ  + VN*VN
               ACKE   = ACKE   + KE
               ACKESQ = ACKESQ + KE*KE
            ENDIF

C     ****************************************************************
C     ** END DISPLACEMENT MOVE                                      **
C     ****************************************************************

 90         CONTINUE
```

```
C     ****************************************************************
C     ** ENDS LOOP OVER TIME SLICES                                 **
C     ****************************************************************

 97       CONTINUE

C     ** WRITE OUT THE INSTANTANEOUS VALUES ON FORT.9 **
            IF ( MOD ( STEP, IPRINT ) .EQ. 0 ) THEN
               WRITE(9,*) STEP, KE/FTN0, V/FTN0
            ENDIF

C     ** CREATE POSITION DISTRIBUTION **

C          CALL DISTR(30_8,STEP*FTN0*NP/30)

C     ** PERFORM PERIODIC OPERATIONS **

          IF ( MOD ( STEP, IRATIO ) .EQ. 0 ) THEN

C     ** ADJUST MAXIMUM DISPLACEMENT **

            RATIO  = ACATMA  / DBLE ( FTN0 * IRATIO )
            RATIOB = ACATMAB / DBLE ( IRATIO )
            RATIOS = ACATMAS / DBLE ( IRATIO )

            IF ( RATIO .GT. 0.5 ) THEN

C               DRMAX  = DRMAX  * 1.05

            ELSE

C               DRMAX  = DRMAX  * 0.95

            ENDIF

            ACATMA  = 0.0
            ACATMAB = 0.0
            ACATMAS = 0.0

          ENDIF

          IF ( MOD ( STEP, IPRINT ) .EQ. 0 ) THEN

C     ** WRITE OUT RUNTIME INFORMATION **

            WRITE(*,*) '   step       drmax'
            WRITE(*,'(I8,  2XE10.4) ') STEP, DRMAX
            WRITE(*,*) '    acm       ratio      ratiob      ratios
     :       ke           v'
            WRITE(*,'(I8,5(2XE10.4))') INT(ACM),
     :           RATIO, RATIOB, RATIOS,
     :           KE/FTN0, V/FTN0
            WRITE(*,*) '          <ke>           <v>'
            WRITE(*,'(2(2XE13.7))') ACKE/ACM/FTN0, ACV/ACM/FTN0
          ENDIF

          IF ( MOD ( STEP, ISAVE ) .EQ. 0 ) THEN

C     ** WRITE OUT THE CONFIGURATION AT INTERVALS **

            CALL WRITCN ( CNFILE )
          ENDIF

 100    CONTINUE

C     ****************************************************************
C     ** ENDS THE LOOP OVER CYCLES                                  **
C     ****************************************************************

        WRITE(*,'(//'' END OF MARKOV CHAIN          ''//)')

C     ** CHECKS FINAL VALUE OF THE POTENTIAL ENERGY IS CONSISTENT **

        CALL CSSUMUP (POTK, OVRLAP, KEEND, VEND)

        IF ( ABS(VEND - V) .GT. 1.0d-03 ) THEN

           WRITE(*,'('' PROBLEM WITH V ENERGY !!!'')')
           WRITE(*,'('' VEND  = '', E20.6)') VEND
           WRITE(*,'('' V     = '', E20.6)') V

        ENDIF

        IF ( ABS(KEEND - KE) .GT. 1.0d-03 ) THEN

           WRITE(*,'('' PROBLEM WITH KE ENERGY !!!'')')
           WRITE(*,'('' KEEND  = '', E20.6)') KEEND
           WRITE(*,'('' KE     = '', E20.6)') KE

        ENDIF

C     ** WRITE OUT THE FINAL CONFIGURATION FROM THE RUN **

        CALL WRITCN ( CNFILE )

C     ** CALCULATE AND WRITE OUT RUNNING AVERAGES **

        AVV   = ACV / ACM
        ACVSQ = ( ACVSQ / ACM ) - AVV ** 2
        AVKE   = ACKE / ACM
        ACKESQ = ( ACKESQ / ACM ) - AVKE ** 2

C    ** CALCULATE FLUCTUATIONS **

        IF ( ACVSQ  .GT. 0.0 ) FLV  = SQRT ( ACVSQ/ACM )/FTN0
        IF ( ACKESQ .GT. 0.0 ) FLKE = SQRT ( ACKESQ/ACM )/FTN0

        WRITE(*,'(/'' AVERAGES ''/ )')
        WRITE(*,'('' <V/N>                 = '',E12.6)') AVV/FTN0
        WRITE(*,'('' <KE/N>                = '',E12.6)') AVKE/FTN0

        WRITE(*,'(/'' FLUCTUATIONS ''/)')

        WRITE(*,'('' FLUCTUATION IN <V/N>  = '',E12.6)') FLV
        WRITE(*,'('' FLUCTUATION IN <KE/N> = '',E12.6)') FLKE
        WRITE(*,'(/'' END OF SIMULATION ''/)')

C     ****************************************************************
C     ****************************************************************
C     ****************************************************************
C     ** THE END                                                    **
C     ****************************************************************
C     ****************************************************************
C     ****************************************************************
        STOP
        END


        subroutine permcyc()
        implicit none
!       given a particle i, lext(i) returns the cycle it belongs to
        INCLUDE       'mc-cs.par'

        integer*8 i,j,k,ii(mnp),jj,kk,ll

        kk=1
        do i=1,np
           ll=1
           ii(ll)=i
 1         jj=next(ii(ll))
           do j=1,ll
              if(jj.eq.ii(j))goto 2
           enddo
           ll=ll+1
           ii(ll)=jj
           goto 1
 2         do k=1,ll
              lext(ii(k))=kk
           enddo
           kk=kk+1
        enddo

        return
        end


        subroutine switch(i,j)
!       switch i and j
        implicit none
        integer*8 i,j,k
        k=i
        i=j
        j=k
        return
        end


        subroutine update(j,rxp,rap,rbp,ip,nip)
        implicit none
!       updates a portion of the current path x using the proposed path xp
        INCLUDE       'mc-cs.par'

        integer*8 ip,nip,j,l,k,idim
        integer*8 mcm
        real*8 rxp(mdim,0:n)
        real*8 rap(mdim,0:n),rbp(mdim,0:n)

        mcm = j+mbm-floor((j+mbm-.1)/ftn0)*ftn0

        l=0
        if(j+mbm.le.ftn0)then
           do k=j+1,mcm-1
              l=l+1
              do idim=1,dim
                 rx(idim,k,ip)=rxp(idim,l)
                 ra(idim,k,ip)=rap(idim,l)
                 rb(idim,k,ip)=rbp(idim,l)
              enddo
           enddo
        else
           do k=j+1,ftn0
              l=l+1
              do idim=1,dim
                 rx(idim,k,ip)=rxp(idim,l)
                 ra(idim,k,ip)=rap(idim,l)
                 rb(idim,k,ip)=rbp(idim,l)
              enddo
           enddo
           do k=1,mcm-1
              l=l+1
              do idim=1,dim
                 rx(idim,k,nip)=rxp(idim,l)
                 ra(idim,k,nip)=rap(idim,l)
                 rb(idim,k,nip)=rbp(idim,l)
              enddo
           enddo
        endif
        do idim=1,dim
           rx(idim,ftn0+1,ip)=rx(idim,1,nip)
           rx(idim,0,nip)=rx(idim,ftn0,ip)
           ra(idim,ftn0+1,ip)=ra(idim,1,nip)
           ra(idim,0,nip)=ra(idim,ftn0,ip)
           rb(idim,ftn0+1,ip)=rb(idim,1,nip)
           rb(idim,0,nip)=rb(idim,ftn0,ip)
        enddo

        return
        end


        subroutine swap(j,ip,nip,kp,nkp)
        implicit none
```

```
!     updates a portion of the current path x using the proposed path xp
      INCLUDE       'mc-cs.par'

      integer*8 ip,nip,kp,nkp,j,k,idim
      integer*8 mcm
      real*8    rr

      mcm = j+mbm-floor((j+mbm-.1)/ftn0)*ftn0

      if(j+mbm.le.ftn0)then
         do k=mcm,ftn0
            do idim=1,dim
               rr=rx(idim,k,ip)
               rx(idim,k,ip)=rx(idim,k,kp)
               rx(idim,k,kp)=rr
               rr=ra(idim,k,ip)
               ra(idim,k,ip)=ra(idim,k,kp)
               ra(idim,k,kp)=rr
               rr=rb(idim,k,ip)
               rb(idim,k,ip)=rb(idim,k,kp)
               rb(idim,k,kp)=rr
               rr=pa(idim,k,ip)
               pa(idim,k,ip)=pa(idim,k,kp)
               pa(idim,k,kp)=rr
               rr=pb(idim,k,ip)
               pb(idim,k,ip)=pb(idim,k,kp)
               pb(idim,k,kp)=rr
            enddo
         enddo
         do idim=1,dim
            rx(idim,ftn0+1,ip)=rx(idim,1,nkp)
            rx(idim,ftn0+1,kp)=rx(idim,1,nip)
            rx(idim,0,nip)=rx(idim,ftn0,kp)
            rx(idim,0,nkp)=rx(idim,ftn0,ip)
            ra(idim,ftn0+1,ip)=ra(idim,1,nkp)
            ra(idim,ftn0+1,kp)=ra(idim,1,nip)
            ra(idim,0,nip)=ra(idim,ftn0,kp)
            ra(idim,0,nkp)=ra(idim,ftn0,ip)
            rb(idim,ftn0+1,ip)=rb(idim,1,nkp)
            rb(idim,ftn0+1,kp)=rb(idim,1,nip)
            rb(idim,0,nip)=rb(idim,ftn0,kp)
            rb(idim,0,nkp)=rb(idim,ftn0,ip)
         enddo
      endif
      return
      end


      subroutine bridge(x0,x1,std,xnew,out)
      implicit none
!     sample m gaussians with std from xnew(0)=x0 to xnew(ftn0)=x1
      INCLUDE       'mc-cs.par'

      integer*8 l1,l2,l3,j,out,idim
      real*8 std,d,s,xi
      real*8 x0(mdim),x1(mdim),xnew(mdim,0:n)

      out=0
      l3=mbm
      do idim=1,dim
         xnew(idim,0)=x0(idim)
         xnew(idim,l3)=x0(idim)+((x1(idim)-x0(idim))-
     :         dnint((x1(idim)-x0(idim))/sigma)*sigma)
      enddo
      do j=1,mbm-1
         l1=j-1
         l2=j
         s=std*(dble(l3-l2)/dble(l3-l1))**0.5d0
         do idim=1,dim
            d=xnew(idim,l3)-xnew(idim,l1)
            d=d-dnint(d/sigma)*sigma
            xnew(idim,j)=xnew(idim,l1)+d/dble(l3-l1)+xi(s)
            xnew(idim,j)=xnew(idim,j)-dnint(xnew(idim,j)/sigma)*sigma
         enddo
      enddo
      return
      end


      function xi(std)
!     sample a gaussian with standard deviation std (box-muller method)
      implicit none
      real*8 xi,std,pi,ranf
      data pi/3.14159265358979323846264338328d0/
      xi=cos(pi*ranf(0.d0))*std*sqrt(-2.d0*log(tiny(pi)+ranf(0.d0)))
      return
      end


      SUBROUTINE DISTR(NN,NORM)
      IMPLICIT NONE
C     WRITES ON FORT.10 THE X-POSITION DISTRIBUTION
      INCLUDE       'mc-cs.par'

      REAL*8        DS,DIST(0:1000)
      INTEGER*8     NN,NORM,I,J,K
      SAVE          DIST

      DS=SIGMA/NN

      DO I=0,NN
         DO J=1,FTNO
            DO K=1,NP
               IF(-SIGMA/2+(I-.5)*DS.LT.RX(1,J,K).AND.
     :              RX(1,J,K).LT.-SIGMA/2+(I+.5)*DS) THEN
                  DIST(I)=DIST(I)+1.DO
               ENDIF
            ENDDO
         ENDDO
         WRITE(10,*) -SIGMA/2+I*DS,DIST(I)/NORM
      ENDDO
      CLOSE(UNIT=10)

      RETURN
      END


      FUNCTION FACT ( N )
      IMPLICIT NONE
C     FACTORIAL FUNCTION
      INTEGER*8 FACT,N,P,I
      P=1
      DO I=1,N
         P=P*I
      ENDDO
      FACT=P
      END


      SUBROUTINE CSSUMUP (POTK, OVRLAP, KE, V)
      IMPLICIT NONE
C     *****************************************************************
C     ** CALCULATES THE TOTAL ACTION                                 **
C     **                                                             **
C     ** USAGE:                                                      **
C     **                                                             **
C     ** THE SUBROUTINE RETURNS THE TOTAL ACTION AT THE              **
C     ** BEGINNING AND END OF THE RUN.                               **
C     *****************************************************************
      INCLUDE       'mc-cs.par'

      REAL*8        V, KE, VV, KK
      LOGICAL       OVRLAP
      CHARACTER     POTK*(*)

      REAL*8        RXII, RXIJ
      REAL*8        VIJ, WIJ, RIJSQ, W, WW

      INTEGER*8     TAU, I, J, IDIM
C     *****************************************************************

C     POTENTIAL ACTION

      VV      = 0.0

C     ** LOOP OVER ALL THE PAIRS IN THE LIQUID **

      DO TAU = 1, FTNO
         DO 100 I = 1, NP - 1
            DO 99 J = I + 1, NP
               RIJSQ = 0.d0
               DO IDIM = 1, DIM
                  RXIJ = RX(IDIM,TAU,I) - RX(IDIM,TAU,J)
C     ** MINIMUM IMAGE THE PAIR SEPARATIONS **
                  RXIJ = RXIJ -
     :                 DNINT ( RXIJ/SIGMA )*SIGMA
               RIJSQ = RIJSQ + RXIJ * RXIJ
               ENDDO

               CALL POT (RIJSQ, VIJ, WIJ, POTK)
               VV   = VV + VIJ
C              WW   = WW + WIJ
 99          CONTINUE
 100      CONTINUE
          V=VV
      ENDDO

C     KINETIC ACTION

      KK     = 0.0

      DO TAU = 1, FTNO
         DO I = 1, NP
            DO IDIM = 1, DIM
               KK=KK+(PA(IDIM,TAU,I)**2.+PB(IDIM,TAU,I)**2.)/4/FTM
            ENDDO
         ENDDO
         KE=KK
      ENDDO

      RETURN
      END


      SUBROUTINE PENERGY ( POTK, RXI, I, TAU,
     :    V, W )
      IMPLICIT NONE
C     *****************************************************************
C     ** RETURNS THE POTENTIAL ENERGY OF ATOM I WITH ALL OTHER ATOMS.  **
C     **                                                             **
C     ** PRINCIPAL VARIABLES:                                        **
C     **                                                             **
C     ** INTEGER I                THE ATOM OF INTEREST               **
C     ** INTEGER NP               THE NUMBER OF ATOMS                **
C     ** INTEGER TAU              THE TIMESTEP                       **
C     ** REAL*8    RX, RY, RZ       THE ATOM POSITIONS               **
C     ** REAL*8    RXI,RYI,RZI      THE COORDINATES OF ATOM I        **
C     ** REAL*8    V                THE POTENTIAL ENERGY OF ATOM I   **
C     ** REAL*8    W                THE VIRIAL OF ATOM I             **
C     **                                                             **
C     ** USAGE:                                                      **
C     **                                                             **
C     ** THIS SUBROUTINE IS USED TO CALCULATE THE CHANGE OF ENERGY    **
C     ** DURING A TRIAL MOVE OF ATOM I. IT IS CALLED BEFORE AND      **
C     ** AFTER THE RANDOM DISPLACEMENT OF I.                         **
C     *****************************************************************
      INCLUDE     'mc-cs.par'

      REAL*8        RXI(MDIM), V, W
      INTEGER*8     I, J, TAU, IDIM
      CHARACTER     POTK*(*)
```

```
      REAL*8       RXIJ, RIJSQ, VIJ, WIJ

C     ***************************************************************

      V      = 0.0
      W      = 0.0

C     ** LOOP OVER ALL MOLECULES EXCEPT I  **

      DO 100 J = 1, NP

         IF ( I .NE. J ) THEN
            RIJSQ = 0.d0
            DO IDIM = 1, DIM
               RXIJ = RXI(IDIM) - RX(IDIM,TAU,J)

               RXIJ = RXIJ -
     :              DNINT ( RXIJ/SIGMA )*SIGMA
               RIJSQ = RIJSQ + RXIJ * RXIJ
            ENDDO

            CALL POT (RIJSQ, VIJ, WIJ, POTK)
            V   = V + VIJ
c            W   = W + WIJ

 90       ENDIF

100     CONTINUE

        RETURN
        END


        SUBROUTINE PPNERGY ( POTK, RXI, I, RXJ, J,
     :     TAU, V, W )
        IMPLICIT NONE
C     ***************************************************************
C     ** RETURNS THE POTENTIAL ENERGY OF ATOMS I AND J WITH          **
C     ** ALL OTHER ATOMS PLUS THE ONE BETWEEN I AND J                **
C     **                                                             **
C     ** PRINCIPAL VARIABLES:                                        **
C     **                                                             **
C     ** INTEGER I, J              THE ATOMS OF INTEREST             **
C     ** INTEGER NP                THE NUMBER OF ATOMS               **
C     ** INTEGER TAU               THE TIMESTEP                      **
C     ** REAL*8    RX, RY, RZ        THE ATOM POSITIONS              **
C     ** REAL*8    RXI,RYI,RZI       THE COORDINATES OF ATOM I       **
C     ** REAL*8    RXJ,RYJ,RZJ       THE COORDINATES OF ATOM J       **
C     ** REAL*8    V                 THE POTENTIAL ENERGY OF ATOM I  **
C     ** REAL*8    W                 THE VIRIAL OF ATOM I            **
C     **                                                             **
C     ** USAGE:                                                      **
C     **                                                             **
C     ** THIS SUBROUTINE IS USED TO CALCULATE THE CHANGE OF ENERGY   **
C     ** DURING A TRIAL MOVE OF ATOM I. IT IS CALLED BEFORE AND      **
C     ** AFTER THE RANDOM DISPLACEMENT OF I.                         **
C     ***************************************************************
        INCLUDE     'mc-cs.par'

        REAL*8        RXI(MDIM), RXJ(MDIM), V, W
        INTEGER*8     I, J, K, TAU, IDIM
        CHARACTER     POTK*(*)

        REAL*8        RXIJ, RIJSQ, VIJ, WIJ

C     ***************************************************************

        V      = 0.0
        W      = 0.0

C     ** LOOP OVER ALL MOLECULES EXCEPT I AND J **

        DO 100 K = 1, NP
           IF ( K .NE. I .AND. K .NE. J ) THEN
              RIJSQ = 0.d0
              DO IDIM = 1, DIM
                 RXIJ = RXI(IDIM) - RX(IDIM,TAU,K)

                 RXIJ = RXIJ -
     :                DNINT ( RXIJ/SIGMA )*SIGMA
                 RIJSQ = RIJSQ + RXIJ * RXIJ
              ENDDO

              CALL POT (RIJSQ, VIJ, WIJ, POTK)
              V   = V + VIJ
              W   = W + WIJ

              IF ( I .NE. J ) THEN
                 RIJSQ = 0.d0
                 DO IDIM = 1, DIM
                    RXIJ = RXJ(IDIM) - RX(IDIM,TAU,K)

                    RXIJ = RXIJ -
     :                   DNINT ( RXIJ/SIGMA )*SIGMA
                    RIJSQ = RIJSQ + RXIJ * RXIJ
                 ENDDO

                 CALL POT (RIJSQ, VIJ, WIJ, POTK)
                 V   = V + VIJ
                 W   = W + WIJ

              ENDIF

           ENDIF

 100    CONTINUE

C       RETURN

        IF (I .NE. J) THEN
           RIJSQ = 0.d0
           DO IDIM = 1, DIM
              RXIJ = RXI(IDIM) - RXJ(IDIM)

              RXIJ = RXIJ -
     :             DNINT ( RXIJ/SIGMA )*SIGMA
              RIJSQ = RIJSQ + RXIJ * RXIJ
           ENDDO

           CALL POT (RIJSQ, VIJ, WIJ, POTK)
           V   = V + VIJ
           W   = W + WIJ
        ENDIF

        RETURN
        END


        SUBROUTINE CSKENERGY ( PAI, PBI, I, TAU, KE )
        IMPLICIT NONE
C     ***************************************************************
C     ** RETURNS THE KINETIC ENERGY OF ATOM I WITH ALL OTHER ATOMS.  **
C     **                                                             **
C     ** PRINCIPAL VARIABLES:                                        **
C     **                                                             **
C     ** INTEGER I                 THE ATOM OF INTEREST              **
C     ** INTEGER NP                THE NUMBER OF ATOMS               **
C     ** INTEGER TAU               THE TIMESTEP                      **
C     ** REAL*8    RX, RY, RZ        THE ATOM POSITIONS              **
C     ** REAL*8    RXI,RYI,RZI       THE COORDINATES OF ATOM I       **
C     ** REAL*8    KE                THE KINETIC ENERGY OF ATOM I    **
C     **                                                             **
C     ** USAGE:                                                      **
C     **                                                             **
C     ** THIS SUBROUTINE IS USED TO CALCULATE THE CHANGE OF ENERGY   **
C     ** DURING A TRIAL MOVE OF ATOM I. IT IS CALLED BEFORE AND      **
C     ** AFTER THE RANDOM DISPLACEMENT OF I.                         **
C     ***************************************************************
        INCLUDE     'mc-cs.par'

        REAL*8        PAI(MDIM), PBI(MDIM), KE
        INTEGER*8     I, TAU, IDIM

C     ***************************************************************

        KE = 0.d0

        DO IDIM = 1, DIM

           KE=KE+(PAI(IDIM)**2.+PBI(IDIM)**2.)/4/FTM

        ENDDO

        RETURN
        END


        SUBROUTINE CSKKKNERGY ( I, TAU, KE )
        IMPLICIT NONE
C     ***************************************************************
C     ** RETURNS THE KINETIC ENERGY OF ATOM I WITH ALL OTHER ATOMS.  **
C     **                                                             **
C     ** PRINCIPAL VARIABLES:                                        **
C     **                                                             **
C     ** INTEGER I                 THE ATOM OF INTEREST              **
C     ** INTEGER NP                THE NUMBER OF ATOMS               **
C     ** INTEGER TAU               THE TIMESTEP                      **
C     ** REAL*8    RX, RY, RZ        THE ATOM POSITIONS              **
C     ** REAL*8    RXI,RYI,RZI       THE COORDINATES OF ATOM I       **
C     ** REAL*8    KE                THE KINETIC ENERGY OF ATOM I    **
C     **                                                             **
C     ** USAGE:                                                      **
C     **                                                             **
C     ** THIS SUBROUTINE IS USED TO CALCULATE THE CHANGE OF ENERGY   **
C     ** DURING A TRIAL MOVE OF ATOM I. IT IS CALLED BEFORE AND      **
C     ** AFTER THE RANDOM DISPLACEMENT OF I.                         **
C     ***************************************************************
        INCLUDE     'mc-cs.par'

        REAL*8        KE
        INTEGER*8     I, NI, TAU, IDIM

C     ***************************************************************

        KE = 0.d0

        DO IDIM = 1, DIM

           KE=KE+(PA(IDIM,TAU,I)**2.+PB(IDIM,TAU,I)**2.)/4/FTM

        ENDDO

        RETURN
        END


        REAL*8 FUNCTION RANF ( DUMMY )

C     ***************************************************************
C     ** RETURNS A UNIFORM RANDOM VARIATE IN THE RANGE 0 TO 1.       **
C     **                                                             **
C     **                ***************                              **
C     **                **  WARNING  **                              **
C     **                ***************                              **
C     **                                                             **
C     ** GOOD RANDOM NUMBER GENERATORS ARE MACHINE SPECIFIC.         **
C     ** PLEASE USE THE ONE RECOMMENDED FOR YOUR MACHINE.            **
C     **                                                             **
C     ** RAND(FLAG) returns a pseudo-random number from a uniform    **
C     ** distribution between 0 and 1. If FLAG is 0, the next number **
C     ** in the current sequence is returned; if FLAG is 1, the      **
```

```
C    ** generator is restarted by CALL SRAND(0); if FLAG has any      **
C    ** other value, it is used as a new seed with SRAND.             **
C    ******************************************************************
       REAL*8      DUMMY

       RANF = RAND ( )

       RETURN
       END


       SUBROUTINE READCN ( CNFILE )
       IMPLICIT NONE
C    ******************************************************************
C    ** SUBROUTINE TO READ IN THE CONFIGURATION FROM UNIT 10          **
C    ******************************************************************
       INCLUDE     'mc-cs.par'

       CHARACTER   CNFILE*(*), HASH*1, MYFMT*77

       INTEGER*8   CNUNIT, I, J, NNP, NI, IDIM
       PARAMETER ( CNUNIT = 10 )

C   ******************************************************************

       OPEN ( UNIT = CNUNIT, FILE = CNFILE, STATUS = 'OLD' )
       WRITE(MYFMT,'(A,I10,A)') '(I3,3X,',5*DIM,'(F12.6,3X))'

       READ ( CNUNIT,* ) HASH, FTNO, NNP
       IF ( NNP .NE. NP ) STOP 'N ERROR IN READCN'

       DO 100 I = 1, NNP
          READ ( CNUNIT,* ) HASH, NEXT(I)
          NI = NEXT(I)
          DO 90 J = 1, FTNO+1
             READ (CNUNIT, MYFMT) LEXT(I), (RX(IDIM,J,I),IDIM=1,DIM),
     :            (RA(IDIM,J,I),IDIM=1,DIM), (RB(IDIM,J,I),IDIM=1,DIM),
     :            (PA(IDIM,J,I),IDIM=1,DIM), (PB(IDIM,J,I),IDIM=1,DIM)
 90       ENDDO
          READ ( CNUNIT,13 )
          READ ( CNUNIT,13 )
          DO IDIM = 1, DIM
             RX(IDIM,0,NI) = RX(IDIM,FTNO,I)
          ENDDO
 100    ENDDO
 13     FORMAT(A2)

       CLOSE ( UNIT = CNUNIT )

       RETURN
       END


       SUBROUTINE WRITCN ( CNFILE )
       IMPLICIT NONE
C    ******************************************************************
C    ** SUBROUTINE TO WRITE OUT THE CONFIGURATION TO UNIT 10          **
C    ******************************************************************
       INCLUDE     'mc-cs.par'

       CHARACTER   CNFILE*(*), MYFMT*77

       REAL*8      RXI(DIM), RAI(DIM), RBI(DIM), PAI(DIM), PBI(DIM)
       INTEGER*8   CNUNIT, I, J, IDIM
       PARAMETER ( CNUNIT = 10 )

C   ******************************************************************

       OPEN ( UNIT = CNUNIT, FILE = CNFILE, STATUS = 'UNKNOWN' )
       WRITE(MYFMT,'(A,I10,A)') '(I3,3X,',5*DIM,'(F12.6,3X))'

       WRITE(*,*) 'output to file ------------------------------------
     :--------------'
       CALL FLUSH ( CNUNIT )
       WRITE ( CNUNIT,* ) '#',FTNO,NP

       DO 100 I = 1, NP
          WRITE ( CNUNIT,* ) '#', NEXT(I)
          DO 90 J = 1, FTNO+1
             DO IDIM = 1, DIM
                RXI(IDIM) = RX(IDIM,J,I)
                RAI(IDIM) = RA(IDIM,J,I)
                RBI(IDIM) = RB(IDIM,J,I)
                PAI(IDIM) = PA(IDIM,J,I)
                PBI(IDIM) = PB(IDIM,J,I)
             ENDDO
             WRITE (CNUNIT, MYFMT) LEXT(I),(RXI(IDIM),IDIM=1,DIM),
     :            (RAI(IDIM),IDIM=1,DIM), (RBI(IDIM),IDIM=1,DIM),
     :            (PAI(IDIM),IDIM=1,DIM), (PBI(IDIM),IDIM=1,DIM)
 90       ENDDO
          WRITE ( CNUNIT,13 ) ''
          WRITE ( CNUNIT,13 ) ''
 100    ENDDO
 13     FORMAT(A2)

       CLOSE ( UNIT = CNUNIT )

       RETURN
       END


       SUBROUTINE INITCN ( CNFILE )
       IMPLICIT NONE
C    ******************************************************************
C    ** SUBROUTINE TO INITIALIZE THE CONFIGURATION TO UNIT 10         **
C    ******************************************************************
       INCLUDE     'mc-cs.par'

       REAL*8      RANF
       CHARACTER   CNFILE*(*), MYFMT*77

       INTEGER*8   CNUNIT, I, J, IDIM, I1
       REAL*8      RXI(DIM), RXII(DIM, NP), RXIJ, RIJSQ
       PARAMETER ( CNUNIT = 10 )

C   ******************************************************************

       OPEN ( UNIT = CNUNIT, FILE = CNFILE, STATUS = 'UNKNOWN' )
       WRITE(MYFMT,'(A,I10,A)') '(I3,3X,',5*DIM,'(F12.6,3X))'

       WRITE ( CNUNIT,* ) '#',FTNO,NP
       RXII = 0.d0

       DO 100 I = 1, NP
          WRITE ( CNUNIT,* ) '#', NEXT(I)
 10       DO IDIM = 1, DIM
             RXI(IDIM) = SIGMA*(RANF(0.d0)-.5)
          ENDDO
          DO I1 = 1, NP
             RIJSQ = 0.d0
             DO IDIM = 1, DIM
                RXIJ = RXI(IDIM)-RXII(IDIM,I1)
                RXIJ = RXIJ  -
     :               DNINT ( RXIJ/SIGMA )*SIGMA
                RIJSQ = RIJSQ + RXIJ * RXIJ
             ENDDO
             IF (RIJSQ.LE.EPSR*EPSR) THEN
                GOTO 10
             ENDIF
          ENDDO
          DO IDIM = 1, DIM
             RXII(IDIM,I) = RXI(IDIM)
          ENDDO
          DO J = 1, FTNO+1
             WRITE (CNUNIT, MYFMT) I, (RXI(IDIM),IDIM=1,DIM),
     :            (RXI(IDIM),IDIM=1,DIM), (RXI(IDIM),IDIM=1,DIM),
     :            (RXI(IDIM),IDIM=1,DIM), (RXI(IDIM),IDIM=1,DIM)
          ENDDO
          WRITE ( CNUNIT,13 ) ''
          WRITE ( CNUNIT,13 ) ''
 100    CONTINUE
 13     FORMAT(A2)

       CLOSE ( UNIT = CNUNIT )

       RETURN
       END


       SUBROUTINE POT (RIJSQ, VIJ, WIJ, POTK)
       IMPLICIT NONE
C    ******************************************************************
C    ** SUBROUTINE FOR THE PAIR POTENTIAL                             **
C    ******************************************************************
       INCLUDE     'mc-cs.par'

       CHARACTER POTK*(*)
       REAL*8    RIJ, RIJSQ, RMIN, RMSQ, RCSQ, SR2, SR6, VIJ, WIJ
       REAL*8    COU3, F
       PARAMETER ( COU3 = 4.760154727959107013287634700 57d0 )

C       PI = ACOS(-1.d0)

       VIJ = 0.0
       WIJ = 0.0
       IF (DIM .GT. 3) RETURN
       IF (POTK .EQ. 'FREE') RETURN

       IF (POTK .EQ. 'LJ') THEN
C          RMIN = 2.d0**(1./6)
C          RMIN = TINY(PI)
C          RMIN = EPSA
          RMIN = 0.d0
          RIJSQ = RIJSQ / (SIG * SIG)
          RMSQ = RMIN*RMIN
          RCSQ = RCUT*RCUT
          IF (RIJSQ .LT. RMSQ) THEN
             RIJSQ = RMSQ
             SR2 = 1.d0 / RIJSQ
             SR6 = SR2 * SR2 * SR2
             VIJ = SR6 * ( SR6 - 1.d0 )
             WIJ = SR6 * ( SR6 - 5.d-1 )
             VIJ = 4.d0 * EPS * VIJ
             WIJ = 48.d0 * EPS * WIJ / 3.d0
          ELSEIF (RIJSQ .GT. RMSQ .AND. RIJSQ .LT. RCSQ) THEN
             SR2 = 1.d0 / RIJSQ
             SR6 = SR2 * SR2 * SR2
             VIJ = SR6 * ( SR6 - 1.d0 )
             WIJ = SR6 * ( SR6 - 5.d-1 )
             VIJ = 4.d0 * EPS * VIJ
             WIJ = 48.d0 * EPS * WIJ / 3.d0
          ENDIF
       ELSEIF (POTK .EQ. 'BUMP') THEN
          RIJ = SQRT( RIJSQ )
          IF (RIJ .LE. SIG) THEN
             VIJ = EPSR
          ENDIF
       ELSEIF (POTK .EQ. 'PSW') THEN
          RIJ = SQRT( RIJSQ )
          IF (RIJ .LE. SIG) THEN
             VIJ = EPSR
          ELSEIF (RIJ .LE. SIG + RCUT .AND. RIJ .GT. SIG) THEN
             VIJ = -EPSA
          ENDIF
       ELSEIF (POTK .EQ. 'COULOMB') THEN
          F = NP/(NP-1)
          RIJ = SQRT( RIJSQ )
          IF (RIJ .GT. EPS) THEN
             IF (DIM .EQ. 3) THEN
                VIJ = -F*COU3*SIG/SIGMA/2.
                VIJ = VIJ + SIG/RIJ
             ELSEIF (DIM .EQ. 2) THEN
```

```
               VIJ = -F*(6.-PI+LOG(4.)-4.*LOG(SIGMA/SIG))/4.
               VIJ = VIJ - LOG(RIJ/SIG)
            ELSEIF (DIM .EQ. 1) THEN
               VIJ = F*SIGMA/SIG/4.
               VIJ = VIJ - RIJ/SIG
            ENDIF
         ELSE
            RIJ = EPS
            IF (DIM .EQ. 3) THEN
               VIJ = -F*COU3*SIG/SIGMA/2.
               VIJ = VIJ + SIG/RIJ
            ELSEIF (DIM .EQ. 2) THEN
               VIJ = -F*(6.-PI+LOG(4.)-4.*LOG(SIGMA/SIG))/4.
               VIJ = VIJ - LOG(RIJ/SIG)
            ELSEIF (DIM .EQ. 1) THEN
               VIJ = F*SIGMA/SIG/4.
               VIJ = VIJ - RIJ/SIG
            ENDIF
          ENDIF
       ELSEIF (POTK .EQ. 'HARMONIC') THEN
          VIJ = 0.5*RIJSQ/SIG**2.
       ENDIF

       RETURN
       END


       subroutine cs(qqa,ppa,qqb,ppb,q,qp,zeta)
       implicit none
C    *****************************************************************
C    ** THE CHOERENT STATE                                          **
C    *****************************************************************
       INCLUDE     'mc-cs.par'

       complex*16 ii,aa(mdim),bb(mdim),psia,psib,gab,zeta
       real*8     mo,qqa(mdim),ppa(mdim),qqb(mdim),ppb(mdim)
       real*8     q(mdim),qp(mdim),aux
       real*8     rgab,rzeta
       integer*8  idim
       parameter ( ii = cmplx(0.d0,1.d0) )

c      harmonic oscillator
       mo = moho

       do idim=1,dim
          aa(idim) = (mo*qqa(idim)+ii*ppa(idim))/sqrt(2*mo)
          bb(idim) = (mo*qqb(idim)+ii*ppb(idim))/sqrt(2*mo)
       enddo

c      coherent states
       psia=0.d0
       psib=0.d0
       do idim=1,dim
          aux=q(idim)-sqrt(2/mo)*dble(aa(idim))
          aux=aux-dnint(aux/sigma)*sigma
          psia = psia - aux**2.*mo/2
          psia = psia + ii*aux*sqrt(2*mo)*dimag(aa(idim))

          aux=qp(idim)-sqrt(2/mo)*dble(bb(idim))
          aux=aux-dnint(aux/sigma)*sigma
          psib = psib - aux**2.*mo/2
          psib = psib + ii*aux*sqrt(2*mo)*dimag(bb(idim))
       enddo
       psia = exp(psia)
       psia = (mo/pi)**(dim/4.)*psia

       psib = exp(psib)
       psib = (mo/pi)**(dim/4.)*psib

c      normalization
       gab = 0.d0
       do idim=1,dim
          gab = gab - (cdabs(aa(idim))**2.+cdabs(bb(idim))**2.)/2
          gab = gab + dconjg(aa(idim))*bb(idim)
          gab = gab + (qqa(idim)*ppa(idim)-qqb(idim)*ppb(idim))*ii/2
       enddo
       gab  = exp(gab)
       rgab = dble(gab)

c      zeta
       zeta = 0.d0
       do idim=1,dim
          zeta = zeta - ls*(ppb(idim)**2.+ppa(idim)**2.)/4/ftm
       enddo
       zeta  = psia*dconjg(psib)*gab*exp(zeta)
       rzeta = dble(zeta)

c       print *, zeta

       return
       end


       subroutine update_cs(j,rxp,rap,rbp,pap,pbp,ip,nip)
       implicit none
!      updates a portion of the current path x using the proposed path xp
       INCLUDE       'mc-cs.par'

       integer*8 ip,nip,j,l,k,idim
       real*8 rxp(mdim,0:n)
       real*8 rap(mdim,0:n),rbp(mdim,0:n)
       real*8 pap(mdim,0:n),pbp(mdim,0:n)

       l=0
       do k=j+1,ftn0
          l=l+1
          do idim=1,dim
             rx(idim,k,ip)=rxp(idim,l)
             ra(idim,k,ip)=rap(idim,l)
             rb(idim,k,ip)=rbp(idim,l)
             pa(idim,k,ip)=pap(idim,l)
             pb(idim,k,ip)=pbp(idim,l)
          enddo
       enddo
       do k=1,j-1
          l=l+1
          do idim=1,dim
             rx(idim,k,nip)=rxp(idim,l)
             ra(idim,k,nip)=rap(idim,l)
             rb(idim,k,nip)=rbp(idim,l)
             pa(idim,k,nip)=pap(idim,l)
             pb(idim,k,nip)=pbp(idim,l)
          enddo
       enddo
       do idim=1,dim
          rx(idim,ftn0+1,ip)=rx(idim,1,nip)
          rx(idim,0,nip)=rx(idim,ftn0,ip)
       enddo

       return
       end


       subroutine acccs(p,ip,kp,j,rxp,rxpp,rap,rapp,rbp,rbpp)
       implicit none
!      complete acceptance probability
       INCLUDE       'mc-cs.par'

       integer*8 ip,kp,j,idim
       real*8 rxp(mdim,0:n),rxpp(mdim,0:n)
       real*8 rap(mdim,0:n),rapp(mdim,0:n)
       real*8 rbp(mdim,0:n),rbpp(mdim,0:n)
       real*8 p,rho,rrx
       integer*8 mcm,i,ll,nip,nkp

       mcm = j+mbm-floor((j+mbm-.1)/ftn0)*ftn0
       nip=next(ip)
       nkp=next(kp)

       rho=0.d0
       ll=-1
       if(j+mbm.le.ftn0)then
          do i=j,mcm-1
             ll=ll+1
             do idim=1,dim
                rrx=rxp(idim,ll)-rxp(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2
                rrx=rap(idim,ll)-rap(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2
                rrx=rbp(idim,ll)-rbp(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2

                rrx=rx(idim,i,ip)-rx(idim,i+1,ip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
                rrx=ra(idim,i,ip)-ra(idim,i+1,ip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
                rrx=rb(idim,i,ip)-rb(idim,i+1,ip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
             enddo
          enddo
       else
          do i=j,ftn0
             ll=ll+1
             do idim=1,dim
                rrx=rxp(idim,ll)-rxp(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2
                rrx=rap(idim,ll)-rap(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2
                rrx=rbp(idim,ll)-rbp(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2

                rrx=rx(idim,i,ip)-rx(idim,i+1,ip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
                rrx=ra(idim,i,ip)-ra(idim,i+1,ip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
                rrx=rb(idim,i,ip)-rb(idim,i+1,ip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
             enddo
          enddo
          do i=1,mcm-1
             ll=ll+1
             do idim=1,dim
                rrx=rxp(idim,ll)-rxp(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2
                rrx=rap(idim,ll)-rap(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2
                rrx=rbp(idim,ll)-rbp(idim,ll+1)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho+rrx**2

                rrx=rx(idim,i,nip)-rx(idim,i+1,nip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
                rrx=ra(idim,i,nip)-ra(idim,i+1,nip)
                rrx=rrx-dnint(rrx/sigma)*sigma
                rho=rho-rrx**2
                rrx=rb(idim,i,nip)-rb(idim,i+1,nip)
                rrx=rrx-dnint(rrx/sigma)*sigma
```

```
               rho=rho-rrx**2
            enddo
         enddo
      endif
      if (ip.ne.kp) then
         ll=-1
         if(j+mbm.le.ftn0)then
            do i=j,mcm-1
               ll=ll+1
               do idim=1,dim
                  rrx=rxpp(idim,ll)-rxpp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2
                  rrx=rapp(idim,ll)-rapp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2
                  rrx=rbpp(idim,ll)-rbpp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2

                  rrx=rx(idim,i,kp)-rx(idim,i+1,kp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
                  rrx=ra(idim,i,kp)-ra(idim,i+1,kp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
                  rrx=rb(idim,i,kp)-rb(idim,i+1,kp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
               enddo
            enddo
         else
            do i=j,ftn0
               ll=ll+1
               do idim=1,dim
                  rrx=rxpp(idim,ll)-rxpp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2
                  rrx=rapp(idim,ll)-rapp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2
                  rrx=rbpp(idim,ll)-rbpp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2

                  rrx=rx(idim,i,kp)-rx(idim,i+1,kp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
                  rrx=ra(idim,i,kp)-ra(idim,i+1,kp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
                  rrx=rb(idim,i,kp)-rb(idim,i+1,kp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
               enddo
            enddo
            do i=1,mcm-1
               ll=ll+1
               do idim=1,dim
                  rrx=rxpp(idim,ll)-rxpp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2
                  rrx=rapp(idim,ll)-rapp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2
                  rrx=rbpp(idim,ll)-rbpp(idim,ll+1)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho+rrx**2

                  rrx=rx(idim,i,nkp)-rx(idim,i+1,nkp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
                  rrx=ra(idim,i,nkp)-ra(idim,i+1,nkp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
                  rrx=rb(idim,i,nkp)-rb(idim,i+1,nkp)
                  rrx=rrx-dnint(rrx/sigma)*sigma
                  rho=rho-rrx**2
               enddo
            enddo
         endif
      endif
      rho=ftm*rho/(2.*ls)
      p=exp(-rho)
      return
      end


------------------------------------------------
      mc-cs.par
------------------------------------------------
C     ******************************************************************
```

```
C     **  MC-CS.PAR                                                   **
C     ******************************************************************
        INTEGER*8     MDIM, N, MNP
        REAL*8        PI

        PARAMETER ( MDIM = 10   ) ! maximum number of dimensions
        PARAMETER ( MNP  = 1000 ) ! maximum number of particles
        PARAMETER ( N    = 3000 ) ! maximum number of time slices

        PARAMETER ( PI = 3.14159265358979323846264338328d0 )

        REAL*8        RX(MDIM,0:N,MNP)
        REAL*8        RA(MDIM,0:N,MNP), RB(MDIM,0:N,MNP)
        REAL*8        PA(MDIM,0:N,MNP), PB(MDIM,0:N,MNP)
        REAL*8        FTM, LS
        REAL*8        SIGMA, SIG, EPSA, EPSR, EPS, RCUT
        REAL*8        MOHO
        INTEGER*8     FTN0, NP, NEXT(MNP), LEXT(MNP), MEXT(N,MNP), MBM
        INTEGER*8     DIM


! ghost path
        COMMON / BLOCK0 / RA, RB, PA, PB
! real path
        COMMON / BLOCK1 / RX, DIM
! pair-potential parameters
        COMMON / BLOCK2 / SIG, EPSR, EPSA, EPS, RCUT
! mass, timestep, box edge, # timeslices, # particles
        COMMON / BLOCK3 / FTM, LS, SIGMA, FTN0, NP
! permutations
        COMMON / BLOCK4 / NEXT, LEXT, MEXT, MBM
! harmonic oscillator for coherent states
        COMMON / BLOCK5 / MOHO


------------------------------------------------
        data-cs-new.in
------------------------------------------------
0 number of spatial dimensions  (1,2,3,...<= MDIM)
2
1 seed of the random sequence                  RAND
44
2 if bose                                     (T/F)
T
3 number of particles                      (<= MNP)
16
4 the potential                     SUBROUTINE POT
LJ
5 # time slices = 1/temperature/timestep     (< N)
250
6 mass                              (hbar = kb = 1)
0.0830594d0
7 # of cycles                                (nstep)
5000000000000000
8 # of steps between output lines          (iprint)
100
9 # of steps between configuration saves    (isave)
100
10 # of steps for equilibration             (iequi)
0
11 # of steps for acceptance ratios        (iratio)
100
12 configuration file name
conf.xyz
13 enter 0 if initialization needed
0
14 density                           THERMODYNAMICS
.05d0
15 temperature                       THERMODYNAMICS
.2d0
16 maximum displacement/box edge
.01d0
17 number of bridge timeslices       (>= 1; <= #5)
50
18 potential cutoff distance (LJ)   SUBROUTINE POT
2.5d0
19 sig    (LJ 2.556)                SUBROUTINE POT
2.556d0
20 epsr   (LJ INIT)                 SUBROUTINE POT
3.d0
21 epsa                             SUBROUTINE POT
1.d0
22 eps    (LJ 10.22)                SUBROUTINE POT
1.022d1
23 moho = mass_ho*omega                        HO
5.814d0
24 if only displace                          (T/F)
F
25 if only bridge                            (T/F)
F
26 if zero path initially  (0 in 13)         (T/F)
F
```