Supplementary material for the work: "Quantum Hard Spheres with Affine Quantization"

Riccardo Fantoni*
Università di Trieste, Dipartimento di Fisica, strada Costiera 11, 34151 Grignano (Trieste), Italy
(Dated: April 13, 2026)

Supplementary material to the article "Quantum Hard Spheres with Affine Quantization".

Keywords: Hard-Spheres; Affine-Quantization; Bose-Einstein Statistics; Path Integral Monte Carlo; Structure; Thermodynamics; Internal Energy; Equation of State; Superfluid; Superglass

I. INTRODUCTION

We here report the FORTRAN code listing of the code used in the simulations developed in the publication "Quantum Hard Spheres with Affine Quantization". This is shown in Appendix A. The model studied in that publication was that of a fluid of Affine-Quantization Hard-Sphere (AQHS) bosons.

The plane waves [1] Path Integral Monte Carlo (PIMC) method used in our computer experiments is the one described in Ref. [2]. A single MC step is made of a swap of a randomly chosen pair of particles through the *Lévy construction* of two Brownian bridge between two randomly chosen timeslices on different paths (see Section V.G of Ref. [2] and references therein for the multislice move) and of a displacement of all the timeslices of a particle path chosen at random (the singleslice move). This allows the permutation sampling necessary in calculating the symmetric density matrix required by the study of the Bose-Einstein statistics. The extent of the singleslice displacement is chosen at the beginning of the run through the input variable `ALPHA` or it can be adjusted during the run every `IRATIO` MC steps so to have acceptance ratios closest to 1/2 (this functionality is commented out in the present listing).

We discretize the imaginary time between 0 and `BETA` into `FTNO` timeslices of length `LS=BETA/FTNO`. One usually wants to compare simulations at fixed `LS`. The multislice move requires the reconstruction of a randomly chosen number `MBM`$\leq$`MBMM`$\leq$`FNTO` of timeslices between a randomly chosen timeslice `CP` on a randomly chosen particle `IP` and a timeslice on a randomly chosen particle `KP`. Here `MBMM` is the maximum number of timeslices in a Brownian bridge. The singleslice move requires the displacement of all the `FTNO` timeslices of the path of the particle `IP` chosen at random. So a single MC step requires a maximum of 2`MBMM`+`FTNO` slice moves. In the code we wrote the FORTRAN instructions for the multislice move in lowercase and the ones for the singleslice move in uppercase. The simulation uses only the multislice move if the logical variable `IFBRIDGE` is `TRUE` and only the singleslice move if the logical variable `IFDISP` is `TRUE`. The simulation lasts for a total number of MC steps `NSTEP`. All this variables can be chosen in the `data-qhs.in` simulation configuration input file at startup.

The code works in any space dimension `DIM`$\leq$`MDIM` with a number of particles `NP` of mass `FTM` with natural units $\hbar = k_B = 1$. One has to specify also the thermodynamic configuration giving the density `DENS` and the absolute temperature `TEMP`.

The pair potential between the particles `POTK` is encoded in `SUBROUTINE POT` at the end of the code listing. It includes the AQHS effective potential (`QHS`) and the canonical HS (`BUMP`). Other test cases included are the harmonic potenial (`HARMONIC`), the Coulomb potential (`COULOMB`), the penetrable square well potential (`PSW`), and the Lennard-Jones potential (`LJ`). The ideal gas is obtained choosing the string `FREE` in the `data-qhs.in` simulation configuration input file. The potential can be further specified through the variables `RCUT`, `SIG`, `EPSR`, `EPSA`, and `EPS` in the simulation configuration input file.

The simulation calculates both thermodynamic quantities and static structural properties. For the thermodynamic quantities we measure: the kinetic energy `KE`, the potential energy `V`, the virial of the forces `W` (which is used to find the pressure), and the superfluid fraction `SF`. These quantities are stored in the output file `fort.9` every `IPRINT` MC steps. Whereas the path configurations output file `CNFILE` is stored in a *.xyz* format every `ISAVE` MC steps. The first `IEQUI` MC steps are used for the equilibration of the Markov chain. Quantities are only calculated after these equilibration steps. In order to determine efficiently the superfluid fraction at low temperature, which requires large `FTNO`, it is convenient to deactivate the displace move choosing `TRUE` for `IFBRIDGE`. For the structural properties we measure the radial distribution function $g(r)$ which is written in the output file `rdf.dat`. In the current implementation of the code, the $g(r)$ is only calculated during the displacement move every `ICALCG` timeslices.

The initial path configurations file is created at the beginning of the run if the variable `INIT` is set to 0, otherwise it is read from the file `CNFILE` stored on disk. Initially the first timeslice of the particles are placed at a distance larger than `EPSA` with all other timeslices copied above the first one for each particle, unless the logical variable `IFZERO` is set to `TRUE` in which case all the paths are set to 0 at all timeslices.

The logical variable `IFB` is set in the simulation configuration input file. If it is set to `TRUE` Bose-Einstein

* riccardo.fantoni@scuola.istruzione.it
[1] For a coherent states PIMC see Ref. [1].

statistics is used, otherwise is used Boltzmann statistics.

The variable `SEED` is also set in the simulation configuration input file. It is the positive integer seed of the pseudo random number generator.

ACKNOWLEDGMENTS

I would like to thank prof. Saverio Moroni for his support in creating the PIMC computer code used for the simulation of the AQHS in Bose statistics. In particular for the development of the brownian bridge and the consequent particles permutation sampling.

Appendix A: The code

This is the code used for the PIMC computer experiment. We list here the main FORTRAN code `pimc.f` with the included `mc-bose.par` parameters file. And the input simulation configuration file `data-qhs.in` that is read at the beginning of the run.

```
*******************************************************************************
*** pimc.f ********************************************************************
*******************************************************************************
       PROGRAM PIMC
       IMPLICIT NONE
C     ****************************************************************
C     ** MONTE CARLO SIMULATION PROGRAM FOR [PW] PATH INTEGRAL       **
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     **                                                            **
C     ** OBSERVABLES:                                              **
C     ** kinetic energy       KE                                    **
C     ** potential energy     V                                     **
C     ** forces virial        W                                     **
C     ** superfluid fraction  SF                                    **
C     ** radial distribution function on 'rdf.dat'                  **
C     ****************************************************************
       INCLUDE       'mc-bose.par'                      ! parameters

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

       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        AVW, ACW, ACWSQ, FLW
       REAL*8        AVKE, ACKE, ACKESQ, FLKE
       REAL*8        ACT, ACTNEW, ACTOLD, DELTACT, DELTACTB
       REAL*8        RXIOLD(MDIM), RXINEW(MDIM)
       REAL*8        RXP(MDIM,0:N), RXPP(MDIM,0:N)
       REAL*8        STD, PS, KKK, VVV, WWW
       REAL*8        RATIO, RATIOS, RATIOB
       REAL*8        WIND(MDIM), WINDS(MDIM), AVWIND(MDIM)
       REAL*8        ACWIND(MDIM), ACWINDSQ(MDIM), FLWIND(MDIM)
       REAL*8        SFI, SF

       INTEGER*8     BIN, MAXBIN, MAXBIN2, HIST(1000000), ICALCG
       REAL*8        CONST, RLOWER, RUPPER
       REAL*8        TSTEP, NIDEAL, ACGR(1000000), GR

       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(*,'('' PLANE WAVES                                  '')')
       WRITE(*,'('' PATH INTEGRAL MONTE CARLO PROGRAM            '')')
       OPEN (UNIT=10, FILE='data-gr.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,*) ICALCG
       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,*) 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

       WRITE(*,'('' ********** PROGRAM PIMC ************      ''/)')
       WRITE(*,'('' PLANE WAVES                                  '')')
       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 (*,*) FTN0
       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 INTERVAL FOR RDF CALCULATION           '')')
       READ (*,*) ICALCG
       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(*,'('' 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. FTN0) THEN
         WRITE(*,*) "number of timeslices in bridge > ",FTN0
         STOP
      ENDIF

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/FTN0      ! 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(*,'('' # TIME SLICES            '',I10   )') FTN0
      WRITE(*,'('' TIME STEP                '',E10.4 )') LS

C    ** CONVERT INPUT DATA TO PROGRAM UNITS **

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

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

      MAXBIN  = INT ( SIGMA/DELR )
      MAXBIN2 = INT ( MAXBIN/2.0 )
      DO I = 1, MAXBIN
         HIST(I) = 0
      ENDDO

      IF (DIM .eq. 3) CONST = 4.d0 * PI * DENS / 3.d0
      IF (DIM .eq. 2) CONST = PI * DENS
      IF (DIM .eq. 1) CONST = 2.d0 * DENS

C     ** INITIALIZE CONFIGURATION **

      DO I = 1, NP
         NEXT(I) = I
         LEXT(I) = I
         DO J = 1, FTN0
            MEXT(J,I)  = I
         ENDDO
      ENDDO

      IF ( INIT .EQ. 0 ) THEN
         PRINT*,"PATHS INITIALIZATION ..............."
         IF (IFZERO) THEN
            RX=0.              ! all paths on the origin
            GOTO 13
         ENDIF
         CALL INITCN ( CNFILE ) ! random configuration
         CALL READCN ( CNFILE ) ! random configuration
 13      CONTINUE
      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
      ACW     = 0.0
      ACWSQ   = 0.0
      FLW     = 0.0
      ACKE    = 0.0
      ACKESQ  = 0.0
      FLKE    = 0.0
      ACWIND  = 0.0
      ACWINDSQ= 0.0
      FLWIND  = 0.0
      ACM     = 0.0
      ACATMA  = 0.0
      ACATMAB = 0.0
      ACATMAS = 0.0
      DO BIN = 1, MAXBIN2
         ACGR(BIN)   = 0.0
      ENDDO

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 SUMUP (POTK, OVRLAP, KE, V, W)

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

      WRITE(*,'('' INITIAL V                '',E10.4)') VS
      WRITE(*,'('' INITIAL W                '',E10.4)') WS
      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(FTN0*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   ! uncomment if only displacement move

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

         nip=next(ip)
         nkp=next(kp)
         ps=0.d0

         vold=0.d0
         wold=0.d0
         if(cp+mbm.le.ftn0)then
            do i=cp+1,mcm-1
               call pppnergy ( 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 pppnergy ( 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 pppnergy ( 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 kkknergy ( rx(:,i,k), 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)
```

```
            if (ip.ne.kp) then
               call bridge(rx(:,cp,kp),rx(:,mcm,ip),std,
     :              rxpp,out2)
            endif
         else
            call bridge(rx(:,cp,ip),rx(:,mcm,nkp),std,
     :           rxp,out1)
            if (ip.ne.kp) then
               call bridge(rx(:,cp,kp),rx(:,mcm,nip),std,
     :              rxpp,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 pppnergy ( potk, rxp(:,ll), ip,
     :                 rxp(:,ll), kp, i,
     :                 vvv, www )
               else
                  call pppnergy ( 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 pppnergy ( potk, rxp(:,ll), ip,
     :                 rxp(:,ll), kp, i,
     :                 vvv, www )
               else
                  call pppnergy ( 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 pppnergy ( potk, rxp(:,ll), nip,
     :                 rxp(:,ll), nkp, i,
     :                 vvv, www )
               else
                  call pppnergy ( potk, rxpp(:,ll), nip,
     :                 rxp(:,ll), nkp, i,
     :                 vvv, www )
               endif
               vnew=vnew+vvv
               wnew=wnew+www
            enddo
         endif

         ps=ps+vnew-vold

         call acc_p(ps,ip,kp,cp)
         if (ps.gt.ranf(dummy)) then
            acatmab = acatmab + 1.d0
            call update(cp,rxp,ip,nkp)
            if (ip.ne.kp) then
               call update(cp,rxpp,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 kkknergy ( rx(:,i,k), 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
               WN = W
            ENDIF
            CALL CWIND( WIND )

C     ** ACCUMULATE AVERAGES **

            ACV    = ACV    + VN
            ACVSQ  = ACVSQ  + VN*VN
            ACW    = ACW    + WN
            ACWSQ  = ACWSQ  + WN*WN
            ACKE   = ACKE   + KE
            ACKESQ = ACKESQ + KE*KE
            DO I=1,DIM
               ACWIND(I)   = ACWIND(I)   + WIND(I)
               ACWINDSQ(I) = ACWINDSQ(I) + WIND(I)*WIND(I)
            ENDDO
         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, FTN0

            DO IDIM = 1, DIM
               RXIOLD(IDIM) = RX(IDIM,C1,IP)
            ENDDO

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

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

            CALL KENERGY ( RXIOLD, IP, C1,
     :           KEOLD )

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
            ENDDO

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

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

            CALL KENERGY ( RXINEW, IP, C1,
     :           KENEW )

C     ** INSTANTANEOUS VALUE OF THE ACTION **

            ACTNEW = KENEW + VNEW

            DELTV  = VNEW  - VOLD
            DELTW  = WNEW  - WOLD
            DELTKE = KENEW - KEOLD

C     ** CHECK FOR ACCEPTANCE **

            DELTACT  = ACTNEW - ACTOLD
            DELTACTB = LS*DELTACT

            IF ( DELTACTB .LT. 75.0 ) THEN

               IF ( DELTACT .LE. 0.0 ) THEN
                  V          = V  + DELTV
                  W          = W  + DELTW
                  KE         = KE + DELTKE
                  ACATMA = ACATMA + 1.0
                  DO IDIM = 1, DIM
                     RX(IDIM,C1,IP)  = RXINEW(IDIM)
c     imaginary time periodic boundary conditions
                     IF (C1.EQ.1) THEN
                        RX(IDIM,FTN0+1,PIP)=RX(IDIM,C1,IP)
                     ENDIF
                     IF (C1.EQ.FTN0) THEN
                        RX(IDIM,0,NIP)=RX(IDIM,C1,IP)
                     ENDIF
                  ENDDO

               ELSEIF ( EXP ( - DELTACTB ) .GT. RANF ( DUMMY ) ) THEN
                  V          = V  + DELTV
                  W          = W  + DELTW
                  KE         = KE + DELTKE
                  ACATMA = ACATMA + 1.0
                  DO IDIM = 1, DIM
                     RX(IDIM,C1,IP)  = RXINEW(IDIM)
c     imaginary time periodic boundary conditions
                     IF (C1.EQ.1) THEN
```

```
                        RX(IDIM,FTNO+1,PIP)=RX(IDIM,C1,IP)
                     ENDIF
                     IF (C1.EQ.FTNO) THEN
                        RX(IDIM,0,NIP)=RX(IDIM,C1,IP)
                     ENDIF
                  ENDDO

               ENDIF

            ENDIF

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

C     ** CALCULATE INSTANTANEOUS VALUES **

C     HISTOGRAM FOR g(r)
               IF ( MOD ( C1, ICALCG ) .EQ. 0 ) THEN
                  CALL GOFR( C1, DENS, MAXBIN, HIST )
                  DO BIN = 1, MAXBIN2
                     ACGR(BIN) = ACGR(BIN)+DBLE(HIST(BIN))
                  ENDDO
               ENDIF

               IF (POTK .EQ. 'LJ') THEN
                  VN = ( V + VLRC )
               ELSE
                  VN = V
                  WN = W
               ENDIF
               CALL CWIND( WIND )

C     ** ACCUMULATE AVERAGES **

               ACV    = ACV    + VN
               ACVSQ  = ACVSQ  + VN*VN
               ACW    = ACW    + WN
               ACWSQ  = ACWSQ  + WN*WN
               ACKE   = ACKE   + KE
               ACKESQ = ACKESQ + KE*KE
               DO I=1,DIM
                  ACWIND(I)   = ACWIND(I)   + WIND(I)
                  ACWINDSQ(I) = ACWINDSQ(I) + WIND(I)*WIND(I)
               ENDDO
            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 **

         SF  = 0.0
         SFI = 0.0
         DO IDIM = 1, DIM
            SFI = SFI + WIND(IDIM)*WIND(IDIM)
            SF  = SF  + ACWINDSQ(IDIM)
         ENDDO
         SFI = SFI*FTM/(DIM*BETA*NP)
         SF  = SF*FTM/(DIM*BETA*NP*ACM)

         CALL FLUSH(9)
         IF ( MOD ( STEP, IPRINT ) .EQ. 0 ) THEN
            WRITE(9,*) STEP, DIM*NP/2/LS-KE/FTNO, V/FTNO, W/FTNO, SFI
         ENDIF

C     ** CREATE POSITION DISTRIBUTION **

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

C     ** PERFORM PERIODIC OPERATIONS **

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

C     ** ADJUST MAXIMUM DISPLACEMENT **

            RATIO  = ACATMA  / DBLE ( FTNO * 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            acm  dr/sig    ard
     :     arb       ars'
            WRITE(*,'(2(2XI13),(2Xf6.4),4(2Xf8.6))') STEP, INT(ACM),
     :                                 ALPHA, RATIO, RATIOB, RATIOS
            WRITE(*,*) '            ke              v              w
     :        sf'
            WRITE(*,'(4(2XE13.7))') KE/FTNO, V/FTNO, W/FTNO,
     :                                 SFI
            WRITE(*,*) '          <ke>            <v>            <w>
     :        <sf>'
            WRITE(*,'(4(2XE13.7))') ACKE/ACM/FTNO, ACV/ACM/FTNO,
     :                                 ACW/ACM/FTNO, SF

         ENDIF

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

C     ** WRITE OUT THE CONFIGURATION AT INTERVALS **

            CALL WRITCN ( CNFILE )

C          ** WRITE OUT THE g(r) **

            IF ( STEP .GT. IEQUI ) THEN
               TSTEP = DBLE((STEP - IEQUI)*INT(FTNO/ICALCG))
               OPEN (UNIT = 10, FILE = 'rdf.dat', STATUS = 'UNKNOWN')
               DO BIN = 1, MAXBIN2
                  RLOWER = DBLE ( BIN - 1 ) * DELR
                  RUPPER = RLOWER + DELR
                  NIDEAL = CONST*( RUPPER**DIM - RLOWER**DIM )
                  GR = DBLE(HIST(BIN))/TSTEP/DBLE(NP)/NIDEAL
                  CALL FLUSH(10)
                  WRITE (10,*) DBLE(BIN-1)*DELR, GR
               ENDDO
               CLOSE (UNIT = 10)
            ENDIF

         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 SUMUP (POTK, OVRLAP, KEEND, VEND, WEND)

         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 )/FTNO
         IF ( ACKESQ .GT. 0.0 ) FLKE = SQRT ( ACKESQ/ACM )/FTNO

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

         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-bose.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 acc_p(p,ip,kp,j)
      implicit none
!     complete acceptance probability
      INCLUDE       'mc-bose.par'

      integer*8 ip,kp,j,idim
      real*8 p, rho
      real*8 rxink,rxnik
      real*8 rxini,rxknk
      integer*8 mcm

      p=exp(-ls*p)         ! contribution from the pair potential
      if (ip.eq.kp) return

      mcm = j+mbm-floor((j+mbm-.1)/ftn0)*ftn0
      rho=0.d0
      do idim=1,dim
         if(j+mbm.le.ftn0)then
            rxink=rx(idim,j,ip)-rx(idim,mcm,kp)
            rxnik=rx(idim,j,kp)-rx(idim,mcm,ip)
            rxini=rx(idim,j,ip)-rx(idim,mcm,ip)
            rxknk=rx(idim,j,kp)-rx(idim,mcm,kp)
            rxink=rxink-dnint(rxink/sigma)*sigma
            rxnik=rxnik-dnint(rxnik/sigma)*sigma
            rxini=rxini-dnint(rxini/sigma)*sigma
            rxknk=rxknk-dnint(rxknk/sigma)*sigma
         else
            rxink=rx(idim,j,ip)-rx(idim,mcm,next(kp))
            rxnik=rx(idim,j,kp)-rx(idim,mcm,next(ip))
            rxini=rx(idim,j,ip)-rx(idim,mcm,next(ip))
            rxknk=rx(idim,j,kp)-rx(idim,mcm,next(kp))
            rxink=rxink-dnint(rxink/sigma)*sigma
            rxnik=rxnik-dnint(rxnik/sigma)*sigma
            rxini=rxini-dnint(rxini/sigma)*sigma
            rxknk=rxknk-dnint(rxknk/sigma)*sigma
         endif
         rho=rho+rxink**2+rxnik**2-rxini**2-rxknk**2
      enddo
      rho=ftm*rho/(2.*mbm*ls)
      p=p*exp(-rho)
      return
      end


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

      integer*8 ip,nip,kp,nkp,j,l,k,idim
      integer*8 mcm
      real*8 rxp(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)
            enddo
         enddo
      else
         do k=j+1,ftn0
            l=l+1
            do idim=1,dim
               rx(idim,k,ip)=rxp(idim,l)
            enddo
         enddo
         do k=1,mcm-1
            l=l+1
            do idim=1,dim
               rx(idim,k,nip)=rxp(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)
      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-bose.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
            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)
         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-bose.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)
c            if (xnew(idim,j).gt.sigma/2.or.
c     :           xnew(idim,j).lt.-sigma/2) then
c               out=1
c               return
c            endif
            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-bose.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,FTN0
            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.D0
               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 SUMUP (POTK, OVRLAP, KE, V, W)
        IMPLICIT NONE
C     ******************************************************************
C     ** CALCULATES THE TOTAL ENERGY                                  **
C     **                                                              **
C     ** USAGE:                                                       **
C     **                                                              **
C     ** THE SUBROUTINE RETURNS THE TOTAL ENERGY AT THE               **
C     ** BEGINNING AND END OF THE RUN.                                **
C     ******************************************************************
        INCLUDE       'mc-bose.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     POTENTIAL ACTION

        VV      = 0.0
        WW      = 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
                 WW   = WW + WIJ
 99           CONTINUE
 100       CONTINUE
           V=VV
           W=WW
        ENDDO

C     KINETIC ACTION

        KK     = 0.0

        DO TAU = 1, FTNO
           DO I = 1, NP
              DO IDIM = 1, DIM
                 RXII = RX(IDIM,TAU,I)-RX(IDIM,TAU+1,I)
                 RXII = RXII -
     :                DNINT ( RXII/SIGMA )*SIGMA
                 KK=KK+FTM*(RXII**2.)/(2.D0*LS**2.)
              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-bose.par'

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

        REAL*8        RXIJ, RIJSQ, VIJ, WIJ

        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
              W   = W + WIJ

 90        ENDIF

100     CONTINUE

        RETURN
        END


        SUBROUTINE PPPNERGY ( 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-bose.par'

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

        REAL*8        RXIJ, RIJSQ, VIJ, WIJ

        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 KENERGY ( RXI, 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-bose.par'

         REAL*8        RXI(MDIM), KE
         REAL*8        RXIP, RXIS, RXII
         INTEGER*8     I, TAU, IDIM

         KE = 0.d0
         DO IDIM = 1, DIM
            RXIP = RX(IDIM,TAU-1,I)
            RXIS = RX(IDIM,TAU+1,I)
            RXII = RXI(IDIM) - RXIP
            RXII = RXII - DNINT ( RXII/SIGMA )*SIGMA
            KE=KE+0.5*FTM*(RXII/LS)**2.
            RXII = RXI(IDIM) - RXIS
            RXII = RXII - DNINT ( RXII/SIGMA )*SIGMA
            KE=KE+0.5*FTM*(RXII/LS)**2.
         ENDDO

         RETURN
         END


         SUBROUTINE KKKNERGY ( RXI, 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-bose.par'

         REAL*8        RXI(MDIM), KE
         REAL*8        RXIS, RXII
         INTEGER*8     I, TAU, IDIM

         KE = 0.d0
         DO IDIM = 1, DIM
            RXIS = RX(IDIM,TAU+1,I)
            RXII = RXI(IDIM) - RXIS
            RXII = RXII - DNINT ( RXII/SIGMA )*SIGMA
            KE=KE+0.5*FTM*(RXII/LS)**2.
         ENDDO

         RETURN
         END



         SUBROUTINE CWIND ( WIND )
         IMPLICIT NONE
C     ****************************************************************
C     ** RETURNS THE WINDING NUMBER.                                  **
C     ****************************************************************
         INCLUDE     'mc-bose.par'

         REAL*8        WIND(MDIM), WI, RX1, RX2
         INTEGER*8     I, J, IDIM

         WIND = 0.d0
         DO I = 1, NP
            DO IDIM = 1, DIM
               DO J = 1, FTNO - 1
                  RX1 = RX(IDIM,J  ,I)
                  RX2 = RX(IDIM,J+1,I)
                  WI  = RX2-RX1-DNINT((RX2-RX1)/SIGMA)*SIGMA
                  WIND(IDIM) = WIND(IDIM) + WI
               ENDDO
            ENDDO
         ENDDO
 20      CONTINUE
         RETURN
         END


         REAL*8 FUNCTION RANF ( DUMMY )
         IMPLICIT NONE
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     ****************************************************************
         INTEGER*8   L, C, M
         PARAMETER ( L = 1029, C = 221591, M = 1048576 )
         INTEGER*8   SEED
         SAVE        SEED
         DATA        SEED / 0 /
         REAL*8      DUMMY

C        SEED = MOD ( SEED * L + C, M )
C        RANF = DBLE( SEED ) / M

         RANF = RAND ( )

         RETURN
         END


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

         CHARACTER   CNFILE*(*), MYFMT*77

         INTEGER*8   CNUNIT, I, J, IDIM
         REAL*8      R
         PARAMETER ( CNUNIT = 10 )

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

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

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

         DO I = 1, NP
            WRITE ( CNUNIT,* ) '#', NEXT(I)
            DO J = 1, FTNO+1
               WRITE ( CNUNIT, MYFMT ) LEXT(I),
     :    (RX(IDIM,J,I)-DNINT(RX(IDIM,J,I)/SIGMA)*SIGMA,IDIM=1,DIM)
C              WRITE ( CNUNIT, MYFMT ) LEXT(I),
C     :    (RX(IDIM,J,I),IDIM=1,DIM)
            ENDDO
```

```
          WRITE ( CNUNIT,13 ) ''
          WRITE ( CNUNIT,13 ) ''
       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-bose.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,',DIM,'(F12.6,3X))'

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

       DO 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.EPSA*EPSA) 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)
             DO IDIM = 1, DIM
                RX(IDIM,J,I) = RXI(IDIM)
             ENDDO
          ENDDO
          WRITE ( CNUNIT,13 ) ''
          WRITE ( CNUNIT,13 ) ''
       ENDDO
 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-bose.par'

       CHARACTER POTK*(*)
       REAL*8    RIJ, RIJSQ, RMIN, RMSQ, RCSQ, SR2, SR6, VIJ, WIJ
       REAL*8    COU3, QHS3, F
       PARAMETER ( COU3 = 4.760154727959107013287634700 57d0 )
       PARAMETER ( QHS3 = 24.d0 )

C       PI = ACOS(-1.d0)

       VIJ = 0.d0
       WIJ = 0.d0
       IF (DIM .GT. 3) THEN
          WRITE(*,*) "DIM > 3 !!!"
          STOP
       ENDIF
       IF (POTK .EQ. 'FREE') RETURN

       IF (POTK .EQ. 'LJ') THEN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     W. L. McMillan, Phys. Rev. A 138, 442 (1965)     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C          RMIN = 2.d0**(1./6)
C          RMIN = EPSA
C          RMIN = TINY(PI)
          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. TINY(PI)) 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 = TINY(PI)
             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. 'QHS') THEN
          F = NP/(NP-1)
          IF (DIM .EQ. 3) THEN
             IF (RIJSQ .LE. SIG*SIG) THEN
                VIJ = 10.**10.
             ELSE
             VIJ = -F*QHS3*(SIGMA-SIG)/SIGMA**3/2./FTM
             VIJ = VIJ + (2.*RIJSQ+SIG**2.)/(RIJSQ-SIG*SIG)**2./2./FTM
             WIJ = -2.*RIJSQ*(RIJSQ+2.*SIG*SIG)/(RIJSQ-SIG*SIG)**3./FTM
             ENDIF
          ELSE
             WRITE(*,*) 'QHS in dimension different from 3'
             STOP
          ENDIF
       ELSEIF (POTK .EQ. 'HARMONIC') THEN
          VIJ = 0.5*RIJSQ/SIG**2.
       ENDIF

       RETURN
       END


       SUBROUTINE GOFR ( C, DENS, MAXBIN, HIST )
       IMPLICIT NONE
C    ****************************************************************
C    ** CALCULATES THE RADIAL DISTRIBUTION FUNCTION HISTOGRAM.       **
C    ****************************************************************
       INCLUDE     'mc-bose.par'

       REAL*8        RXIJ, RIJSQ, RIJ, DENS
       INTEGER*8     C, IDIM, I, J, BIN, MAXBIN, HIST(MAXBIN)

       DO I = 1, NP - 1
          DO J = I + 1, NP
             RIJSQ = 0.
             DO IDIM = 1, DIM
                RXIJ  = RX(IDIM,C,I) - RX(IDIM,C,J)
                RXIJ  = RXIJ - ANINT ( RXIJ/SIGMA ) * SIGMA
                RIJSQ = RIJSQ + RXIJ * RXIJ
             ENDDO
C     CALCULATE HISTOGRAM FOR g(r)
             RIJ = SQRT ( RIJSQ )
             BIN = INT ( RIJ/DELR ) + 1

             IF (BIN .LE. MAXBIN) THEN
                HIST(BIN) = HIST(BIN) + 2
             ENDIF

          ENDDO
       ENDDO

       RETURN
       END



*******************************************************************************
*** mc-bose.par ***************************************************************
*******************************************************************************
  REAL*8        PI
  REAL*8        DELR
       INTEGER*8     MDIM, N, MNP
```

```
      PARAMETER ( PI = 3.14159265358979323846264338328d0 )

  PARAMETER ( MDIM = 10   )   ! maximum number of dimensions
  PARAMETER ( MNP  = 1000 )   ! maximum number of particles
  PARAMETER ( N    = 3000 )   ! maximum number of time slices
  PARAMETER ( DELR = 0.01d0 ) ! grid spacing for g(r)

  INTEGER*8     DIM
        REAL*8        RX(MDIM,0:N,MNP)
        REAL*8        FTM, LS, SIGMA
        REAL*8        SIG, EPSA, EPSR, EPS, RCUT
        INTEGER*8     FTNO, NP
  INTEGER*8     NEXT(MNP), LEXT(MNP), MEXT(N,MNP), MBM

        ! 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, FTNO, NP
        ! permutations
        COMMON / BLOCK4 / NEXT, LEXT, MEXT, MBM
```

```
*********************************************************************************
*** data-gr.in ******************************************************************
*********************************************************************************
0 number of spatial dimensions  (1,2,3,...<= MDIM)
3
1 seed of the random sequence                  RAND
3
2 if bose                                     (T/F)
T
3 number of particles                      (<= MNP)
30
4 the potential                     SUBROUTINE POT
QHS
5 # time slices = 1/temperature/timestep     (< N)
31
6 mass                              (hbar = kb = 1)
1.d0
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)
100
11 # of steps for acceptance ratios        (iratio)
100
12 # of steps for rdf calculation    (icalcg < #5)
3
13 configuration file name
conf.xyz
14 enter 0 if initialization needed
0
15 density                              THERMODYNAMICS
.1d0
16 temperature                          THERMODYNAMICS
.4d0
17 maximum displacement/box edge
.03d0
18 maximum # of bridge timeslices     (> 1; <= #5)
31
19 potential cutoff distance (LJ)   SUBROUTINE POT
2.d0
20 sig    (LJ 2.566)                SUBROUTINE POT
1.d0
21 epsr   (LJ INIT)                 SUBROUTINE POT
1.d10
22 epsa                             SUBROUTINE POT
1.d0
23 eps    (LJ 10.22)                SUBROUTINE POT
1.d1
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
```

AUTHOR DECLARATIONS

Conflicts of interest

None declared.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Funding

None declared.

[1] R. Fantoni, Coherent State Path Integral Monte Carlo, Eur. Phys. J. D **79**, 146 (2025).
[2] D. M. Ceperley, Path integrals in the theory of condensed helium, Rev. Mod. Phys. **67**, 279 (1995).