Coherent State Path Integral Monte Carlo for ^{4}He

Riccardo Fantoni*
Università di Trieste, Dipartimento di Fisica, strada Costiera 11, 34151 Grignano (Trieste), Italy

(Dated: November 3, 2025)

Supplementary material to the article “Coherent State Path Integral Monte Carlo”.

Keywords: Quantum Many Body; Quantum Monte Carlo; Coherent States; Path Integral

I. INTRODUCTION

In the publication “Coherent State Path Integral Monte Carlo” we performed several Path Integral Monte Carlo (PIMC) simulations for a two dimensional, $d = 2$, ^{4}He liquid [1] with either Boltzmann or Bose statistics. The liquid has a surface density $\rho = N/\Omega$ where N is the number of helium atoms in an area Ω of a flat surface, at an inverse temperature $\beta = 1/k_BT$ with k_B Boltzmann constant. The Hamiltonian of the fluid is $\hat{H} = \hat{T} + \upsilon\hat{V} = \hat{P}^2/2m + \upsilon V(Q)$ with $Q = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N)$ the particles positions and momenta $P = (\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2, \ldots, \hat{\boldsymbol{p}}_N) = -i(\boldsymbol{\nabla}_{\boldsymbol{q}_1}, \boldsymbol{\nabla}_{\boldsymbol{q}_2}, \ldots, \boldsymbol{\nabla}_{\boldsymbol{q}_N})$. We compared two different PIMC versions: The conventional Plane Waves PIMC (**PW**PIMC) [2] and our Coherent States PIMC (**CS**PIMC) new algorithm. The two PIMC differ for the expression of the hot kinetic density matrix at an imaginary timestep $\tau = \beta/M$ with M a large number of timeslices. In our quantum simulations we have to take care of two limiting procedures: The continuum limit where the ultraviolet cutoff $\tau \to 0$ or $M \to \infty$ at fixed absolute temperature T and the thermodynamic limit where $N \to \infty$ and $\Omega \to \infty$ at fixed density ρ. We will explicitly worry about the continuum limit in the last two Table VII and VIII for Boltzmann and Bose statistics respectively. To mimic the thermodynamic limit we will simply use a periodic square cell which permeates the whole infinite space. This will give rise to spurious finite size effects unavoidable on a computer.

The PWPIMC requires a multidimensional integral over dNM coordinates whereas our CSPIMC requires $5dMN$ integrations, the usual dNM real particles coordinates, $2dNM$ ghost particles coordinates, and $2dNM$ ghost particles momenta. The coherent states are generated by an Harmonic Oscillator (HO) with mass $m_{h.o.}$ and elastic constant $k = m_{h.o.}\omega^2$. We will introduce the parameter $\xi \equiv m_{h.o.}\omega/2$ and the adimensional one $\varphi \equiv \xi\tau/m = (\sigma_{p.w.}/\sigma_{c.s.})^2/2$ where $\sigma_{p.w.} \equiv \sqrt{2\lambda\tau}$ is the standard deviation in the plane wave scheme, with $\lambda = 1/2m$, and $\sigma_{c.s.} \equiv \sqrt{1/m_{h.o.}\omega}$ is the standard deviation in the coherent state scheme. We will work in units where $\hbar = k_B = 1$ so that the imaginary time has dimensions of temperature.

II. THE SHORT TIME DENSITY MATRIX FOR CSPIMC

The short time τ Green function for CSPIMC can be written as follows:

$$\rho(Q, Q'; \tau) \approx e^{-\tau\upsilon V(Q')}\frac{1}{N!}\sum_{\mathcal{P}}(-1)^{\mathcal{P}}\prod_{\alpha=1}^{N}\zeta_\alpha\left[\boldsymbol{q}_{\mathcal{P}\alpha}|\boldsymbol{q}'_\alpha; \tau, m, \xi\right], \tag{2.1}$$

with $\mathcal{P}$ a permutation of the N **real** identical particles and

$$\zeta_\alpha\left[\boldsymbol{q}|\boldsymbol{q}'; \tau, m, \xi\right] \approx \int \frac{d\boldsymbol{q}_a\, d\boldsymbol{p}_a}{(2\pi)^d}\frac{d\boldsymbol{q}_b\, d\boldsymbol{p}_b}{(2\pi)^d}\psi_\alpha^{\boldsymbol{a}}(\boldsymbol{q})\psi_\alpha^{\boldsymbol{b}\,*}(\boldsymbol{q}')G_{\boldsymbol{a},\boldsymbol{b}}\exp\left[-\tau(\boldsymbol{p}_a^2 + \boldsymbol{p}_b^2)/4m\right],$$

* riccardo.fantoni@scuola.istruzione.it

where $\psi^{\boldsymbol{a}}_\alpha(\boldsymbol{q}) \equiv \psi^{\boldsymbol{a}}(\boldsymbol{q}_\alpha)$ is the CS wave function with ghost canonical variables $\boldsymbol{q_a}$ and $\boldsymbol{p_a}$, the **ghost** of the real particle α, namely

$$\psi^{\boldsymbol{a}}(\boldsymbol{q}_\alpha) \equiv \langle \boldsymbol{q}_\alpha | \boldsymbol{q}_a, \boldsymbol{p}_a\rangle \tag{2.2}$$

$$= \left(\frac{m_{h.o.}\omega}{\pi}\right)^{d/4} \exp\left\{-\frac{m_{h.o.}\omega}{2}\left[\boldsymbol{q}_\alpha - \sqrt{\frac{2}{m_{h.o.}\omega}}\mathrm{Re}(\boldsymbol{a})\right]^2 + i\boldsymbol{q}_\alpha\cdot\sqrt{2m_{h.o.}\omega}\mathrm{Im}(\boldsymbol{a}) - i2\mathrm{Re}(\boldsymbol{a})\cdot\mathrm{Im}(\boldsymbol{a})\right\},$$

$$G_{\boldsymbol{a},\boldsymbol{b}} = \exp\left[-\frac{1}{2}(|\boldsymbol{a}|^2+|\boldsymbol{b}|^2) + \boldsymbol{a}^*\cdot\boldsymbol{b} + \frac{i}{2}(\boldsymbol{q}_a\cdot\boldsymbol{p}_a - \boldsymbol{q}_b\cdot\boldsymbol{p}_b)\right], \tag{2.3}$$

$$\boldsymbol{a} = \frac{1}{\sqrt{2m_{h.o.}\omega}}(m_{h.o.}\omega\boldsymbol{q}_a + i\boldsymbol{p}_a), \tag{2.4}$$

$$\boldsymbol{b} = \frac{1}{\sqrt{2m_{h.o.}\omega}}(m_{h.o.}\omega\boldsymbol{q}_b + i\boldsymbol{p}_b), \tag{2.5}$$

where $G_{\boldsymbol{a},\boldsymbol{b}}$ is the scalar product of the two ghosts "a" and "b". All this is used to find the acceptance probability in `subroutine cs` in the code listing at the end of this material.

III. RESULTS

We compare conventional PWPIMC with our CSPIMC algorithm on specific simulations. We performed computer experiments for $N = 16$ ^{4}He atoms ($m = 0.0830594 \approx 1/12$ Å^{-2}K^{-1}) in two dimensions $d = 2$, in a square periodic cell of area $\Omega = L^2$, interacting with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556$ Å, $\varepsilon = 10.22$ K and a cutoff distance $r_{\mathrm{cut}} = 2.5$ Å (so that $v(r) = 0$ for $r > r_{\mathrm{cut}}\sigma$ without long range corrections), at a density $\rho = N/L^2 = 0.05$ Å^{-2} and various temperatures T or timeslices number M, either with Boltzmann and Bose statistics. As we can see from the phase diagram of this fluid [1] at this density it undergoes a phase transition from a fluid phase at high temperature to a superfluid phase at low temperatures.

We take as initial configuration each atom sitting at all timeslices on a random position choosen so to avoid overlaps with the other atoms.

The Monte Carlo used is the standard Metropolis algorithm [4, 5]. In our simulation we choose as transition move a uniform displacement of each of the dMN real path coordinate $\boldsymbol{q}_\alpha(i\tau) \to \boldsymbol{q}_\alpha(i\tau) + (1/2-\eta)\boldsymbol{\Delta}$ for $\alpha = 1,\ldots,N$ and $i = 1,\ldots,M$, where η is a uniform pseudo-random number in $[0, 1)$ and $\boldsymbol{\Delta}$ a fixed 3-dimensional vector whose magnitude is chosen so to have acceptance ratios close to 1/2. And of each of the $4dMN$ ghost path canonical variables $\boldsymbol{q}_{a_\alpha}(i\tau) \to \boldsymbol{q}_{a_\alpha}(i\tau) + (1/2-\eta)\boldsymbol{\Delta}$, $\boldsymbol{p}_{a_\alpha}(i\tau) \to \boldsymbol{p}_{a_\alpha}(i\tau) + (1/2-\eta)\boldsymbol{\Delta}$, and $\boldsymbol{q}_{b_\alpha}(i\tau) \to \boldsymbol{q}_{b_\alpha}(i\tau) + (1/2-\eta)\boldsymbol{\Delta}$, $\boldsymbol{p}_{b_\alpha}(i\tau) \to \boldsymbol{p}_{b_\alpha}(i\tau) + (1/2-\eta)\boldsymbol{\Delta}$. So that the transition probability density is just a constant and drops out of the acceptance probability. A MC step consists of displacing the M timeslices of a randomly chosen ghost and real particle one by one (in order to activate just this move in the CSPIMC simulation it is necessary to set `TRUE` the entry 24 of the input data file `data-cs-2.in`) and of a displacement of a fixed random number $\leq M$ of timeslices, connecting two randomly chosen real particles, all at once, in a sort of "brownian bridge" (for details look at the `subroutine bridge` in the code listing at the end of this material), so to realize an exchange, *swap*, of two real particles. Here it is also necessary to bridge the "a" and "b" ghosts of particle α so that $\boldsymbol{q}^{new}_{a_\alpha} = \boldsymbol{q}^{new}_{b_\alpha} = \boldsymbol{q}^{new}_\alpha$ (where $\boldsymbol{q}^{new}_\alpha$ is the new position of the real particle α after the brownian bridge), and swap the real particles together with their two ghosts each (in order to activate just this move in the CSPIMC simulation it is necessary to set `TRUE` the entry 25 of the input data file `data-cs-2.in`). [1]

Our PWPIMC simulations (see Tables I, II and III, IV) confirm that at high temperature (classical limit) the nature of the statistics is not important. Whereas it becomes important at low temperatures (quantum regime). The zero temperature (ground state) limit can only be reached through an extrapolation of the PIMC results.

In Tables I and II we use PWPIMC at constant $\tau = 0.025$ K^{-1} (same as Ref. [1]) and various temperatures.

In Tables III and V we compare the PWPIMC with the CSPIMC for Boltzmann statistics at fixed $M = 250$ and various temperatures (In order to use Boltzmann statistics with our CSPIMC code listed in the Appendix we chose `FALSE` in entry 24 of the input data file `data-cs-2.in`). In Tables IV and VI we do the same for Bose statistics (In order to use Bose statistics with our CSPIMC code listed in the Appendix we chose `TRUE` in entry 2 and `FALSE` in entries 24 and 25 of the input data file `data-cs-2.in`).

In all our tables we denote with $\mathcal{E}_p = \upsilon\langle V\rangle$ and

$$\mathcal{E}_k = \begin{cases} \frac{dN}{2\tau} - \frac{\langle (Q_i - Q_{i-1})^2\rangle}{4\lambda\tau^2} & \text{PWPIMC} \\ \frac{dN}{2\tau}\frac{\varphi}{1+\varphi}\phi - \frac{\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle}{4m} & \text{CSPIMC} \end{cases} \tag{3.1}$$

[1] We also tried two independent brownian bridges one between two random real particles and one between two random ghost particle but this does not work because the ghost particle $\boldsymbol{q}_{a_\alpha}$ may freely end up very close to a real particle $\boldsymbol{q}_\beta$. And due to the harmonic coupling the ghost will carry in its neighborhood a real particle $\boldsymbol{q}_\alpha$ which may then overlap with the real particle β producing an explosion of the potential energy.

where $Q_i = Q(i\tau)$ are the N real particles coordinates at the ith timestep and P_a, P_b are the N momenta of the two ghosts. This corresponds to the *thermodynamic* estimator of Ref. [2]. The parameter ϕ, in the CSPIMC case, is necessary in order to find agreement with the kinetic energy of the PWPIMC ,as explained in the main article. Note that the kinetic energy from Eq. (3.1) is the small difference between two large quantities, infinite in the continuum $\tau \to 0$ limit.

TABLE I. Results from **PW**PIMC for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and various temperatures T with $\tau = 0.025$, with **Boltzmann statistics**. In the Table $\mathcal{E}_k$ and $\mathcal{E}_p$ are the total kinetic and potential energies respectively. $\mathcal{E} = \mathcal{E}_k + \mathcal{E}_p$ is the total internal energy.

T (K)	$\mathcal{E}_k$ (K)	$-\mathcal{E}_p$ (K)	$-\mathcal{E}$ (K)
1.0	66.6(2)	94.4(1)	27.8
0.5	60.8(2)	92.9(1)	32.1
0.2	59.0(2)	93.00(8)	34.0
0.1	58.2(2)	93.23(8)	35.0

TABLE II. Results from **PW**PIMC for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and various temperatures T with $\tau = 0.025$, with **Bose statistics**. In the Table $\mathcal{E}_k$ and $\mathcal{E}_p$ are the total kinetic and potential energies respectively. $\mathcal{E} = \mathcal{E}_k + \mathcal{E}_p$ is the total internal energy.

T (K)	$\mathcal{E}_k$ (K)	$-\mathcal{E}_p$ (K)	$-\mathcal{E}$ (K)
1.0	57.1(1)	94.8(3)	37.7
0.5	55.0(5)	93.4(2)	38.4
0.2	55.8(1)	93.55(5)	37.7
0.1	55.9(1)	93.46(5)	37.6

In Tables VII and VIII we performed simulations at fixed $\varphi = \pi/2$, and fixed temperatures $T = 1$ (K) and $T = 0.2$ (K) respectively, at increasing numbers M of timeslices. We see that it is necessary to keep φ constant upon taking the continuum limit in order to find agreement between the results for the CSPIMC and the ones for the PWPIMC for both the kinetic and potential energies. The convergence for the kinetic energy measured through the estimator of Eq. (3.1) is rather slow for CSPIMC and it would be desirable to try also other alternative estimators. Also the equilibration time starting from a random configuration of the atoms is rather long, about 6×10^6 MC steps. We find that, in the continuum $\tau \to 0$ limit, ϕ tends to $\approx \sqrt{2}$, instead of $\approx \sqrt{3}$ as happens in the non interacting case (this is shown in the main article in Table II there), and this is due to the fact that for the interacting system there are additional terms in $\mathcal{E}_k$ due to the interaction as shown by the *direct* estimator of Ref. [2].

TABLE III. Results from **PW**PIMC for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and various temperatures T with $M = 250$, with **Boltzmann statistics**. The runs are 2×10^6 steps long. In the Table $\mathcal{E}_k$ and $\mathcal{E}_p$ are the total kinetic and potential energies respectively. $\mathcal{E} = \mathcal{E}_k + \mathcal{E}_p$ is the total internal energy (note that unlike Ref. [1] we fix M and not τ).

T (K)	$\mathcal{E}_k$ (K)	$-\mathcal{E}_p$ (K)	$-\mathcal{E}$ (K)
1.0	82.3(6)	83.6(1)	1.3(6)
0.5	74.6(4)	84.8(1)	10.2(4)
0.2	62.5(4)	90.5(1)	28.0(4)
0.1	49.0(2)	101.1(1)	52.1(2)

TABLE IV. Results from **PW**PIMC for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and various temperatures T with $M = 250$, with **Bose statistics**. The runs are 2×10^6 steps long. In the Table $\mathcal{E}_k$ and $\mathcal{E}_p$ are the total kinetic and potential energies respectively. $\mathcal{E} = \mathcal{E}_k + \mathcal{E}_p$ is the total internal energy (note that unlike Ref. [1] we fix M and not τ).

T (K)	$\mathcal{E}_k$ (K)	$-\mathcal{E}_p$ (K)	$-\mathcal{E}$ (K)
1.0	77(1)	84.1(4)	7.1(4)
0.5	69.7(7)	84.6(4)	14.9(7)
0.2	60.0(5)	90.6(1)	30.6(5)
0.1	49.3(3)	101.1(2)	51.8(3)

In order to reproduce the PWPIMC results for the total kinetic and potential energies it is necessary to fix in the CSPIMC $\varphi = m_{h.o.}\omega\tau/2m = \pi/2$ and take the continuum limit $\xi = m_{h.o.}\omega/2 \to \infty$ or $\tau \to 0$. From Table VII we see that at fixed $\varphi = \pi/2$, ϕ remains close to $\sqrt{2}$ at small τ. In order to find reasonable results for the kinetic energy in the small τ cases of Tables VI and VIII it is necessary to approach equilibrium “adiabatically” (fast) by keeping the acceptance ratios for the single slice displacement move below or much below 1/2 [2]. This is also

[2] This suggests that it would be more convenient to distinguish

TABLE V. Results from **CS**PIMC with $\varphi = \pi/2$ for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and various temperatures T with $M = 250$, with **Boltzmann statistics**. The runs are 2×10^6 steps long. In the Table $\mathcal{E}_k = \frac{N}{\tau}\frac{\pi}{2+\pi}\phi - \frac{\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle}{4m}$ is taken from Table III and $\mathcal{E}_p$ is the total potential energy.

T (K)	$\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle/4m$ (K)	$\mathcal{E}_k$ (K)	$-\mathcal{E}_p$ (K)	ϕ
1.0	3383(5)	82.3	85.3(4)	1.42
0.5	1735(3)	74.6	85.9(2)	1.48
0.2	668.1(9)	62.5	90.4(2)	1.49
0.1	317.6(4)	49.0	99.3(2)	1.50

TABLE VI. Results from **CS**PIMC with $\varphi = \pi/2$ for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and various temperatures T with $M = 250$, with **Bose statistics**. The runs are 2×10^6 steps long. In the Table $\mathcal{E}_k = \frac{N}{\tau}\frac{\pi}{2+\pi}\phi - \frac{\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle}{4m}$ is taken from Table IV and $\mathcal{E}_p$ is the total potential energy. We counted very few atoms swaps. For example at the lower temperature of the table, we counted just 13 swaps over 10^6 MC steps.

T (K)	$\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle/4m$ (K)	$\mathcal{E}_k$ (K)	$-\mathcal{E}_p$ (K)	ϕ
1.0	3868(6)	77	85.8(4)	1.61
0.5	1722(3)	69.7	85.6(2)	1.47
0.2	643.9(8)	60.0	94.3(2)	1.44
0.1	328.0(4)	49.3	100.4(1)	1.54

important for the measure of the potential energy since it doesn't cost anything for two ghosts to approach each other, then it may happen that in correspondence of the binding of the ghosts of two different particles these also bind producing an artificial positive jump in the potential energy. It is therefore necessary to give a kick $\mathbf{\Delta}$ larger than the extent $\sigma_{c.s.}$ of the cloud of ghosts around their particle in order to unbind the pair.

The slight disagreement in the potential energy between PWPIMC and CSPIMC is due to the different approach to the continuum $\tau \to 0$ limit for the two algorithms, where in the CSPIMC this is affected also by the other available parameter ξ.

between a displacement $\mathbf{\Delta}$ for the positions and one for the momenta.

TABLE VII. Results from **CS**PIMC with $\varphi = \pi/2$ for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and a temperature $T = 1$ with various values of M, with **Boltzmann statistics**. In the Table $\mathcal{E}_k = \frac{N}{\tau}\frac{\pi}{2+\pi}\phi - \frac{\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle}{4m} \approx 82.3$ and $\mathcal{E}_p$ are the total kinetic and potential energies.

M	$\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle/4m$ (K)	ϕ	$-\mathcal{E}_p$ (K)
250	3383(5)	1.42	85.3(4)
500	6818(4)	1.41	80.6(3)
750	10232(5)	1.41	80.1(4)
1000	13696(5)	1.41	79.9(4)

TABLE VIII. Results from **CS**PIMC with $\varphi = \pi/2$ for $N = 16$ ^{4}He atoms ($m = 0.0830594$) in two dimensions, with a Lennard-Jones pair potential with parameters [3] $\sigma = 2.556, \varepsilon = 10.22$ and a cutoff distance $r_{\rm cut} = 2.5$ (so that $v(r) = 0$ for $r > r_{\rm cut}\sigma$ without long range corrections), at a density $N/L^2 = 0.05$ and a temperature $T = 0.2$ with various values of M, with **Bose statistics**. In the Table $\mathcal{E}_k = \frac{N}{\tau}\frac{\pi}{2+\pi}\phi - \frac{\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle}{4m} \approx 60.0$ and $\mathcal{E}_p$ are the total kinetic and potential energies. In order to find reasonable results for the small τ entries in the table is important to have acceptance ratios for the single slice displacement move less or much less than 1/2.

M	$\langle P_a^2(i\tau)+P_b^2(i\tau)\rangle/4m$ (K)	ϕ	$-\mathcal{E}_p$ (K)
250	643.9(8)	1.44	94.3(2)
500	1371(2)	1.46	85.2(2)
750	2514(4)	1.75	83.5(1)
1000	5163(3)	2.67	79.3(1)

IV. CONCLUSIONS

In this supplementary material we proved the statement offered in the publication "Coherent State Path Integral Monte Carlo" that in the CSPIMC algorithm it is necessary to take the continuum limit at fixed $\varphi = \xi\tau/m = \pi/2$. We also presented various simulation results supporting the findings of the main article. It remains an open problem to theoretically asses the, for the time being, empiric values for the two parameters φ and ϕ.

ACKNOWLEDGMENTS

I would like to thank prof. Saverio Moroni for his support in creating the computer code for the plane waves simulations for ^{4}He 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 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-2.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: FTNO                                     **
C      ** units:               hbar=k_B=1                               **
C      ** BETA=1/TEMP=FTNO*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-2.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,*) FTNO
        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

        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/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(*,'('' HO MASS*OMEGA              '',E10.4 )') MOHO
        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

        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, FTN0
              MEXT(J,I)  = I
           ENDDO
        ENDDO

        IF ( INIT .EQ. 0 ) THEN
           PRINT*,"PATHS INITIALIZATION ..............."
           IF (IFZERO) GOTO 333
           CALL INITCN ( CNFILE )          ! random configuration
           CALL READCN ( CNFILE )          ! random configuration
 333       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
        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
C        WLRC12 =   4.0  * VLRC12
C        WLRC6  =   2.0  * VLRC6
C        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(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
           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)
c            if (out1.eq.1) goto 7777     ! wall (change also BRIDGE)
              if (ip.ne.kp) then
                 call bridge(rx(:,cp,kp),rx(:,mcm,ip),std,
     :                rxpp,out2)
c               if (out2.eq.1) goto 7777  ! wall (change also BRIDGE)
              endif
           else
              call bridge(rx(:,cp,ip),rx(:,mcm,nkp),std,
     :             rxp,out1)
c            if (out1.eq.1) goto 7777     ! wall (change also BRIDGE)
              if (ip.ne.kp) then
                 call bridge(rx(:,cp,kp),rx(:,mcm,nip),std,
     :                rxpp,out2)
c               if (out2.eq.1) goto 7777  ! wall (change also BRIDGE)
              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

      ps=ps+vnew-vold

      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+1,ip)
:             ,pb(:,c1,ip),rx(:,c1,ip),rx(:,c1+1,ip),zo)
         zold=zold*zo**2
      enddo
      else
      do c1=cp+1,ftn0
         call cs(ra(:,c1,ip),pa(:,c1,ip),rb(:,c1+1,ip)
:             ,pb(:,c1,ip),rx(:,c1,ip),rx(:,c1+1,ip),zo)
         zold=zold*zo**2
      enddo
      do c1=1,mcm-1
         call cs(ra(:,c1,nkp),pa(:,c1,nkp),rb(:,c1+1,nkp)
:             ,pb(:,c1,nkp),rx(:,c1,nkp),rx(:,c1+1,nkp),zo)
         zold=zold*zo**2
      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+1,kp)
:                ,pb(:,c1,kp),rx(:,c1,kp),rx(:,c1+1,kp),zo)
            zold=zold*zo**2
         enddo
         else
         do c1=cp+1,ftn0
            call cs(ra(:,c1,kp),pa(:,c1,kp),rb(:,c1+1,kp)
:                ,pb(:,c1,kp),rx(:,c1,kp),rx(:,c1+1,kp),zo)
            zold=zold*zo**2
         enddo
         do c1=1,mcm-1
            call cs(ra(:,c1,nip),pa(:,c1,nip),rb(:,c1+1,nip)
:                ,pb(:,c1,nip),rx(:,c1,nip),rx(:,c1+1,nip),zo)
            zold=zold*zo**2
         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(rxp(:,ll),pa(:,c1,ip),rb(:,c1+1,ip)
:             ,pb(:,c1,ip),rxp(:,ll),rx(:,c1+1,ip),zo)
         znew=znew*zo
         call cs(ra(:,c1-1,ip),pa(:,c1,ip),rxp(:,ll)
:             ,pb(:,c1,ip),rx(:,c1-1,ip),rxp(:,ll),zo)
         znew=znew*zo
      enddo
      else
      do c1=cp+1,ftn0
         ll=ll+1
         call cs(rxp(:,ll),pa(:,c1,ip),rb(:,c1+1,ip)
:             ,pb(:,c1+1,ip),rxp(:,ll),rx(:,c1+1,ip),zo)
         znew=znew*zo
         call cs(ra(:,c1-1,ip),pa(:,c1,ip),rxp(:,ll)
:             ,pb(:,c1,ip),rx(:,c1-1,ip),rxp(:,ll),zo)
         znew=znew*zo
      enddo
      do c1=1,mcm-1
         ll=ll+1
         call cs(rxp(:,ll),pa(:,c1,nkp),rb(:,c1+1,nkp)
:             ,pb(:,c1,nkp),rxp(:,ll),rx(:,c1+1,nkp),zo)
         znew=znew*zo
         call cs(ra(:,c1-1,nkp),pa(:,c1,nkp),rxp(:,ll)
:             ,pb(:,c1,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(rxpp(:,ll),pa(:,c1,kp),rb(:,c1+1,kp)
:                ,pb(:,c1,kp),rxpp(:,ll),rx(:,c1+1,kp),zo)
                 znew=znew*zo
                 call cs(ra(:,c1-1,kp),pa(:,c1,kp),rxpp(:,ll)
     :                ,pb(:,c1,kp),rx(:,c1-1,kp),rxpp(:,ll),zo)
                 znew=znew*zo
              enddo
              else
              do c1=cp+1,ftn0
                 ll=ll+1
                 call cs(rxpp(:,ll),pa(:,c1,kp),rb(:,c1+1,kp)
     :                ,pb(:,c1,kp),rxpp(:,ll),rx(:,c1+1,kp),zo)
                 znew=znew*zo
                 call cs(ra(:,c1-1,kp),pa(:,c1,kp),rxpp(:,ll)
     :                ,pb(:,c1,kp),rx(:,c1-1,kp),rxpp(:,ll),zo)
                 znew=znew*zo
              enddo
              do c1=1,mcm-1
                 ll=ll+1
                 call cs(rxpp(:,ll),pa(:,c1,nip),rb(:,c1+1,nip)
     :                ,pb(:,c1,nip),rxpp(:,ll),rx(:,c1+1,nip),zo)
                 znew=znew*zo
                 call cs(ra(:,c1-1,nip),pa(:,c1,nip),rxpp(:,ll)
     :                ,pb(:,c1,nip),rx(:,c1-1,nip),rxpp(:,ll),zo)
                 znew=znew*zo
              enddo
              endif
           endif

           call acc_pcs(ps,ip,kp,cp)
           call acccs(ps,ip,kp,cp,rxp,rxpp)
           ps=ps*dble(znew)/dble(zold)

           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 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
           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

 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    ! uncomment if only swap&bridge move

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

            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,rb(:,c1+1,ip),pbold,
     :           rxiold,rx(:,c1+1,ip),zold)
            call cs(ra(:,c1-1,ip),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,rb(:,c1+1,ip),pbnew,
     :           rxinew,rx(:,c1+1,ip),znew)
            call cs(ra(:,c1-1,ip),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/FTNO, V/FTNO
               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       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/FTNO, V/FTNO
               WRITE(*,*) '            <ke>           <v>'
               WRITE(*,'(2(2XE13.7))') ACKE/ACM/FTNO, ACV/ACM/FTNO
            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 '')')

      STOP
      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,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)

      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)=rxp(idim,l)
               rb(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)
               ra(idim,k,ip)=rxp(idim,l)
               rb(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)
               ra(idim,k,nip)=rxp(idim,l)
               rb(idim,k,nip)=rxp(idim,l)
            enddo
         enddo
      endif
      do idim=1,dim
         ra(idim,j,ip)=rx(idim,j,ip)
         rb(idim,j,ip)=rx(idim,j,ip)
         ra(idim,mcm,ip)=rx(idim,mcm,ip)
         rb(idim,mcm,ip)=rx(idim,mcm,ip)
         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)
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-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,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 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    ** 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    ** 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 GHOST 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 GHOST 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
            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, MYFMT) LEXT(I), (RX(IDIM,FTNO+1,I),IDIM=1,DIM),
    :  (RX(IDIM,FTNO+1,I),IDIM=1,DIM), (RX(IDIM,FTNO+1,I),IDIM=1,DIM),
    :  (RX(IDIM,FTNO+1,I),IDIM=1,DIM), (RX(IDIM,FTNO+1,I),IDIM=1,DIM)
         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 -------------------------------------
    :--------------'
      WRITE ( CNUNIT,* ) '#',FTNO,NP

      DO 100 I = 1, NP
         WRITE ( CNUNIT,* ) '#', NEXT(I)
         DO 90 J = 1, FTNO
            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
         DO IDIM = 1, DIM
            RXI(IDIM) = RX(IDIM,FTNO+1,I)
         ENDDO
         WRITE (CNUNIT, MYFMT) LEXT(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)
         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 = 4.76015472795910701328763470057d0 )

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 acc_pcs(p,ip,kp,j)
        implicit none
!     complete acceptance probability
        INCLUDE       'mc-cs.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
              rho=rho+rxink**2+rxnik**2-rxini**2-rxknk**2
           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
              rho=rho+rxink**2+rxnik**2-rxini**2-rxknk**2
           endif
        enddo
        rho=ftm*rho/(2.*mbm*ls)
        p=p*exp(-3*rho)
        return
        end


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

        integer*8 ip,kp,j,idim
        real*8 p,rxp(mdim,0:n),rxpp(mdim,0:n)
        real*8 rho,rrx,rrxp,rrxpp
        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=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
         rrxp=rxp(idim,ll)-rxp(idim,ll+1)
         rrxp=rrxp-dnint(rrxp/sigma)*sigma
         rho=rho-3*rrxp**2
      enddo
   enddo
else
   do i=j,ftn0
      ll=ll+1
      do idim=1,dim
         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
         rrxp=rxp(idim,ll)-rxp(idim,ll+1)
         rrxp=rrxp-dnint(rrxp/sigma)*sigma
         rho=rho-3*rrxp**2
      enddo
   enddo
   do i=1,mcm-1
      ll=ll+1
      do idim=1,dim
         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
         rrxp=rxp(idim,ll)-rxp(idim,ll+1)
         rrxp=rrxp-dnint(rrxp/sigma)*sigma
         rho=rho-3*rrxp**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=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
            rrxpp=rxpp(idim,ll)-rxpp(idim,ll+1)
            rrxpp=rrxpp-dnint(rrxpp/sigma)*sigma
            rho=rho-3*rrxpp**2
         enddo
      enddo
   else
      do i=j,ftn0
         ll=ll+1
         do idim=1,dim
            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
            rrxpp=rxpp(idim,ll)-rxpp(idim,ll+1)
            rrxpp=rrxpp-dnint(rrxpp/sigma)*sigma
            rho=rho-3*rrxpp**2
         enddo
      enddo
      do i=1,mcm-1
         ll=ll+1
         do idim=1,dim
            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
            rrxpp=rxpp(idim,ll)-rxpp(idim,ll+1)
            rrxpp=rrxpp-dnint(rrxpp/sigma)*sigma
            rho=rho-3*rrxpp**2

         enddo
      enddo
   endif
endif
rho=ftm*rho/(2.*ls)
p=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-2.in
--------------------------------------------------
0 number of spatial dimensions  (1,2,3,...<= MDIM)
2
1 seed of the random sequence                 RAND
3
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)
5000000000000
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
1.d0
16 maximum displacement/box edge
.005d0
17 number of bridge timeslices       (>= 1; <= #5)
150
18 potential cutoff distance (LJ)   SUBROUTINE POT
2.5d0
19 sig    (LJ 2.566)                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
20.7649d0
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
```

[1] M. C. Gordillo and D. M. Ceperley, Path-integral calculation of the two-dimensional ^{4}He phase diagram, Phys. Rev. B **58**, 6447 (1998).

[2] D. M. Ceperley, Path integrals in the theory of condensed helium, Rev. Mod. Phys. **67**, 279 (1995).

[3] W. L. McMillan, Ground State of Liquid He^4, Phys. Rev. **138**, 442 (1964).

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. M. Teller, and E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. **1087**, 21 (1953).

[5] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods* (John Wiley & Sons Inc., New York, 1986).