# Monte Carlo Simulation of the Lennard Jones Fluid in the Canonical Ensemble

The Molecular Sciences Software Institute — Eliseo Marin-Rimoldi and John D. Chodera

## I. INTRODUCTION

### A. Monte Carlo Integration

In statistical mechanics, we are interested in computing averages of thermodynamic properties as a function of atom positions and momenta[1–3]. A thermodynamic average depending only on configurational properties can be computed using the following expectation value integral

$$\langle Q \rangle = \int_V Q\left(\mathbf{r}^N\right) \rho\left(\mathbf{r}^N\right) d\mathbf{r}^N \tag{1}$$

$\mathbf{r}^N$ is a $3N$ dimensional vector containing the positions of the $N$ atoms, where $Q(\mathbf{r}^N)$ is the thermodynamic quantity of interest that depends only on the configuration $\mathbf{r}^N$, $\rho\left(\mathbf{r}^N\right)$ is the probability density whose functional form depends on the statistical mechanical ensemble of interest, and $V$ defines the volume of configuration space over which $\rho$ has support. Note that the integrals over momenta have been factored out, as they can be evaluated analytically. The integral (Eq. 1) is very hard to compute even for small atomic systems. For instance, a monoatomic system of 10 atoms leads to a 30-dimensional integral. Consequently, we need to resort to a numerical integration scheme if we want to study atomic systems.

Monte Carlo methods are numerical techniques frequently used to estimate complex multidimensional integrals which otherwise could not be performed[4,5]. For instance, the integral of the function $f(\mathbf{x})$, where $\mathbf{x} \in \Re^M$, is approximated as

$$I = \int_V f(\mathbf{x})\, d\mathbf{x} = \int_V \frac{f(\mathbf{x})}{h(\mathbf{x})} h(\mathbf{x})\, d\mathbf{x} = \left\langle \frac{f(\mathbf{x})}{h(\mathbf{x})} \right\rangle_{h(\mathbf{x})} \tag{2}$$

The idea of Monte Carlo integration is to estimate the expectation value $\left\langle \frac{f(\mathbf{x})}{h(\mathbf{x})} \right\rangle_{h(\mathbf{x})}$ by generating random samples of $\mathbf{x}$ from the probability density $h(\mathbf{x})$.

## B. Importance Sampling

In Equation 2, we are free to chose the probability distribution $h(\mathbf{x})$. The simplest case is to uniformly generate $\mathbf{x}$ in the volume $V$. In this way, $h(\mathbf{x})$ becomes constant as

$$h(\mathbf{x}) = \frac{1}{V} \tag{3}$$

Using this sampling density $h(\mathbf{x})$, the integral (Eq. 2) becomes

$$I = \int_V f(\mathbf{x})\, d\mathbf{x} \approx \frac{V}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \tag{4}$$

where N is the total number of random samples and $f(\mathbf{x}_i)$ is the integrand evaluated using the $i^{th}$ sample. While using a uniform sampling density often works sufficiently well for simple unidimensional cases, it generally fails to produce useful estimates for complex problems.

The problem at hand involves the evaluation of $3N$-dimensional integral Eq. 1, which is dominated by a small region of configuration space[1–3]. Using a uniform probability distribution $h(\mathbf{r}^N)$ over the configuration space hypervolume $V^{3N}$ to generate representative samples of this subset is not efficient, as most states generated this way would have a low weight.

A solution to this problem is to sample positions $\mathbf{r}^N$ from the desired equilibrium probability density $\rho\left(\mathbf{r}^N\right)$:

$$\mathbf{r}^N \sim \rho(\mathbf{r}^N) \tag{5}$$

This is a way to generate relevant configurations more frequently than configurations that have low probability. Mathematically, we set $h(\mathbf{r}^N) = \rho\left(\mathbf{r}^N\right)$. This idea is known as *importance sampling*.

Combining Eqs. 1 and 2 and the condition $h(\mathbf{r}^N) = \rho\left(\mathbf{r}^N\right)$, we find that

$$\langle Q \rangle \approx \frac{1}{N} \sum_{i=1}^{N} Q\left(\mathbf{r}_i^N\right). \tag{6}$$

Thus, we can get thermodynamic properties by simply computing an unweighted sample average, given that we perform importance sampling from $h(\mathbf{r}^N) = \rho\left(\mathbf{r}^N\right)$.

## C. Detailed Balance Condition

The question now becomes how to generate such atomic positions $\mathbf{r}^N$ (or *states*) distributed according to $\rho\left(\mathbf{r}^N\right)$. In 1953, Metropolis, Rosenbluth, Rosenbluth, and Teller introduced a solution based on Markov chains[6]. They proposed to use the detailed balance condition in order to ensure

proper configurational sampling from the statistical mechanical distribution of interest. In order to generate a new configuration $n$ from an old configuration $m$, the detailed balance condition is

$$\rho_m\left(\mathbf{r}^N\right)\alpha\left(m\to n\right)P_{acc}\left(m\to n\right)=\rho_n\left(\mathbf{r}^N\right)\alpha\left(n\to m\right)P_{acc}\left(n\to m\right) \tag{7}$$

Where $\rho_m\left(\mathbf{r}^N\right)$ is the probability of observing state $m$, $\alpha\left(m\to n\right)$ is the probability of attempting to generate a new state $n$ starting from a state $m$ and $P_{acc}\left(n\to m\right)$ is the probability of accepting such transition. Basically, the condition of detailed balance tells us that the "flux" of transitions from state $m$ to state $n$ equals the flux from state $n$ to state $m$ at equilibrium.

There are many ways to satisfy Eq. 7 by construction of different acceptance probabilities. While Metropolis et. al proposed a choice that both satisfies Eq. 7 and maximizes the average acceptance probability $\langle P_{acc}\rangle$[6], Hastings generalized this to the case where proposal probabilities are not symmetric, such that $\alpha\left(m\to n\right)\neq\alpha\left(n\to m\right)$, producing the *Metropolis-Hastings*[7] acceptance criteria:

$$P_{acc}(m\to n)=\min\left[1,\frac{\alpha\left(n\to m\right)}{\alpha\left(m\to n\right)}\frac{\rho_n\left(\mathbf{r}^N\right)}{\rho_m\left(\mathbf{r}^N\right)}\right]. \tag{8}$$

This algorithm is one of a general class of *Markov chain Monte Carlo (MCMC)* algorithms that generate Markov chains to sample a desired target density, and a great deal of the MCMC literature is valuable for molecular simulations[4].

## II. CANONICAL ENSEMBLE MONTE CARLO OF A LENNARD JONES FLUID

Assume we have $N$ monoatomic particles that interact using the Lennard-Jones (LJ) pairwise potential:

$$U(r)=4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r}\right)^{6}\right] \tag{9}$$

where $r$ is the interparticle distance, $\sigma$ is the distance where the interaction energy is zero, and $\epsilon$ is the well depth. For simulating argon, for example, a common choice is $\sigma=3.4$ Åand $\epsilon/k_B=120$ K.

Our goal is to generate a set of states of $N$ LJ particles distributed according to the canonical (NVT) ensemble

$$\rho_n\left(\mathbf{r}^N;\beta\right)=Z(\beta)^{-1}e^{-\beta U\left(\mathbf{r}^N\right)} \tag{10}$$

$$Z(\beta)\equiv\int_V e^{-\beta U\left(\mathbf{r}^N\right)}d\mathbf{r}^N \tag{11}$$

where $U\left(\mathbf{r}^N\right)$ is the potential energy of the system, $\beta=(k_BT)^{-1}$ is the inverse temperature, $k_B$ is the Boltzmann constant, and $T$ is the absolute temperature.

Note that $U\left(\mathbf{r}^N\right)$ is given by

$$U\left(\mathbf{r}^N\right) = \sum_{i<j} U\left(r_{ij}\right) \tag{12}$$

where $r_{ij} \equiv ||\mathbf{r}_j^N - \mathbf{r}_i^N||_2$ is the interparticle separation distance.

Substituting Eq. 10 into Eq. 8 and assuming $\alpha\left(n \to m\right) = \alpha\left(m \to n\right)$, we obtain

$$P_{acc}(m \to n) = \min\left[1, e^{-\beta \Delta U}\right] \tag{13}$$

where $\Delta U \equiv U(\mathbf{r}_m^N) - U(\mathbf{r}_n^N)$ is the difference in potential energy of the system between the new state $n$ and the old state $m$. Note that the argument of the energy $\mathbf{r}^N$ has been dropped for clarity.

## A. Flow of Calculations in a Metropolis Monte Carlo simulation

The following workflow can be used to implement the Metropolis algorithm to sample the canonical ensemble of configurations of LJ particles:

1. Generate an inital system state $m$.
2. Choose an atom with uniform probability from $\{1, \ldots, N\}$ from old state $m$.
3. Propose a new state $n$ by translating a LJ particle by a uniform random displacement $\Delta r \sim U(-\Delta x, +\Delta x)$ in each dimension. The displacement scale $\Delta x$ should not be too large as this would likely result in particle overlaps, but should not be too small as this would result in a slow sampling of configurational space. More on this below.
4. The difference in energy between the new and old states is computed.
5. The new state is accepted or rejected using the Metropolis criterion. Practically, this can be implemented as follows. If a move from $m$ to $n$ is "downhill", $\beta\Delta U \leq 0$, the move is always accepted. For "uphill" moves, a random number $\zeta$ is generated uniformly on (0,1). If $\zeta < \exp[-\beta\Delta U]$, the move is accepted. Otherwise, the move is rejected. If a non-symmetric proposal is used, this acceptance scheme will have to be modified to implement Eq. 8.

## B. Technical considerations

**Initial configuration.** While computed equilibrium expectations should be independent of the starting configuration, from a practical standpoint, we have to select an initial configuration that is not so atypical of equilibrium configurations that relaxation to equilibrium will require an enormous amount of simulation time. For this particular example, a good way to start is to place

atoms in a 3D lattice. During the Metropolis Monte Carlo simulation at temperatures and box volumes (or pressures) typical of the liquid state, the lattice will "melt" and liquid configurations will be obtained. Once a liquid configuration has been generated from a simulation, you can always save a "snapshot" from this simulation as a starting point for a new simulation of the liquid states, and relaxation to equilibrium will generally be much more rapid as long as the conditions are similar.

**Random number generation.** Computers cannot generate truly random numbers. Instead, they rely on *pseudorandom number generators* (PRNGs) that aim to produce random numbers with the desired statistical properties and long recurrence times between repeats of the same random number sequence. Using a low-quality random number generator can lead to simulation artifacts that can lead to incorrect physical behavior[8,9]. We recommend you avoid writing your own PRNGs, as this can lead to inadvertent implementation of an ill-conceived algorithm[10]. Instead, rely on high-quality, well-understood PRNGs and implementations that have are well-supported, such as the Mersenne Twister[11,12] implementation provided by `numpy.random`.

**Equilibration.** When the initial configuration is highly atypical compared to true samples from the equilibrium density—such as the initial 3D lattice conditions compared to a true disordered liquid state—it may require very long simulation times for the bias in equilibrium averages computed over the entire trajectory to become small compared to the statistical error[13]. It is therefore common practice to discard some initial part of the simulation to *equilibration* and to average over the subsequent *production* region to minimize this bias at the cost of potentially increasing statistical error by including less data in the average. While common practice traditionally had selected an arbitrary initial portion of the simulation to equilibrium, modern best practice recommends the use of an automated approach for selecting the optimal equilibration/production split point in a manner that maximizes the number of statistically uncorrelated samples in the production part of the trajectory[13]. To do this, you can use the `pymbar.timeseries.detectEquilibration` function from the `pymbar` module to analyze an array containing timeseries data for your observable (such as energies, box volumes, or densities)[13].

**System size and periodic boundary conditions.** A typical simulation of a Lennard-Jones fluid is carried out anywhere from 216 to 10,000 particles. This amount of particles is far away of being representative of a bulk liquid. To get around this problem, we employ a trick called periodic boundary conditions. The primary simulation box is surrounded by copy images of itself. For a cubic simulation, there would be 26 images around the central box. This trick has two implications

1. If the position of a particle (i.e. Cartesian coordinates) is outside the simulation box after a

particle translation, an identical particle should enter the box through the opposite face of the box, as shown in Figure.

2. When computing distances $r_{ij}$ used in the evaluation of the LJ potential (Eq. 12), we use the *minimum image distance*. See Image.

**Maximum displacement $\Delta x$.** As noted above, the Metropolis Monte Carlo algorithm requires translating a selected LJ particle by a random perturbation. This displacement should not be too large as this would result in particle overlaps and low acceptance rates; on the other hand, it should not be so small as to result in inefficient sampling of configuration space. A common practice is to adjust the maximum particle displacement $\Delta x$ during an explicit equilibration phase in order to achieve ∼50% acceptance translation rates over a recent window of $\sim N$ Monte Carlo trial moves.

**Energy truncation and tail corrections.** If two particles are separatedby more than a certain distance, we typically truncate their interaction energy if $r > r_c$, where $r_c$ denotes the *cutoff distance*. Truncating interactions removes contribution to the potential energy that might be non negligible and can lead to significant artifacts, such as significantly perturbed densities when a barostat is used to sample the NPT ensemble due to neglected long-range dispersion interactions. We can estimate the truncated interactions by incorporating an energy correction, known as the tail or long range correction. For the Lennard-Jones fluid, we assume that we have an homogeneous liquid at $r > r_c$ to obtain the correction for neglecting this contribution for all interacting pairs of particles[14]:

$$U_{\text{correction}} = \frac{8\pi N^2}{3V}\epsilon\sigma^3\left[\frac{1}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3\right] \tag{14}$$

For a Lennard-Jones fluid, it is common to set $r_c \sim 3\sigma$, since the pair interaction at this separation is small, $U(3\sigma) \approx 4[(3)^{-12} - (3)^{-6} \approx -0.0055\epsilon$, or about 0.5% of the well depth $\epsilon$. You will want to verify that your computed properties are relatively insensitive to the choice of cutoff $r_c$ so that a too-short cutoff does not induce artifacts in computed physical properties.

**Units.** Dealing with unit-bearing quantities—such as energies, distances, masses, and physical constants—in your code can be tricky. Errors can easily creep in when one part of the code assumes one set of implicit units while another part of the code assumes a different set of implicit units, and error like this have led to billion-dollar accidents (such as the loss of the Mars Polar Lander due to one part of the code using metric units while the other used English units[15]). To avoid costly mistakes that complicate debugging, we recommend one of two approaches: (1) use a unit library such as `pint` or `simtk.unit` that automatically handles unit coversions, or (2) select a

single compatible unit system (e.g. mks or Ångstroms-amu-picoseconds) and explicitly document the units for all unit-bearing constants and quantities; The first option is highly preferred, though care must still be paid to ensuring unit-bearing quantities possess the appropriate dimensionality and some considerations must be made for performance impact in parts of the code that must be performant since unit conversion incurs non-negligible overhead.

**Compatible constants.** While some constants are exact, many physical constants are known only to a number of significant digits. All values must be rounded to some number of significant figures when used in computer simulations, and these numbers have finite-precision floating-point representation that further reduces their accuracy. Perhaps surprisingly, choices of how many significant figures that each constant should be rounded to can have an impact on the resulting simulations if care is not take to ensure that constants frequently used together are not rounded in a self-consistent manner. For example, a major difference between potential energies computed in CHARMM and AMBER arises from an early arbitrary choice of the number of significant digits to round a set of critical Coulomb energy coefficients[?] . Fortunately, the National Institute of Standards and Technology (NIST) provides up-to-date guidelines on consistent sets of physical parameters in the 2014 CODATA Self-Consistent Physical Constants (updated every four years). It is highly recommended that the CODATA parameters be adopted unless there is strong need to reproduce flawed results from a specific molecular simulation package.

**Reduced units.** Lennard-Jones fluids have the surprisingly pleasant behavior of possessing universal behavior when expressed in terms of *reduced units*. That is, when plotted in reduced units, all Lennard-Jones fluids exhibit the same universal behavior despite the exact choices of $\epsilon$ and $\sigma$ used in the simulation. These reduced units are generally denoted by an asterisk, such as $T^*$ for reduced temperature, and are given by:

**length:** $L* = L/\sigma$

**volume:** $V^* = V/\sigma^3$

**density:** $\rho^* = N/V^* = N\sigma^3/V$

**temperature:** $T^* = k_B T/\epsilon$

**energy:** $U^* = U/\epsilon$

**pressure:** $p^* = p\sigma^3/\epsilon$

Using reduced units for input and output will allow you to compare your results with others.

### C. Reference calculations

See the following resources for NIST Lennard-Jones fluid reference data:

- Reference thermodynamic data ($N = 500$, $T^* = 0.85$ and $0.90$, $r_c = 3\sigma$)
- Reference snapshot potential energies and virials

## REFERENCES


[1] M. Tuckerman, Statistical mechanics: theory and molecular simulation, 2010.

[2] T. Hill, *An Introduction to Statistical Thermodynamics*, Dover Books on Physics, Dover Publications, 2012.

[3] D. McQuarrie, *Statistical Mechanics*, University Science Books, 2000.

[4] J. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer Series in Statistics, Springer, 2008.

[5] M. Newman and G. Barkema, *Monte Carlo Methods in Statistical Physics*, Clarendon Press, 1999.

[6] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, The Journal of Chemical Physics **21**, 1087 (1953).

[7] W. K. Hastings, Biometrika **57**, 97 (1970).

[8] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, Phys. Rev. Lett. **69**, 3382 (1992).

[9] D. S. Cerutti, R. Duke, P. L. Freddolino, H. Fan, and T. P. Lybrand, J. Chem. Theory Comput. **4**, 1669 (2008).

[10] Wikipedia, RANDU, https://en.wikipedia.org/wiki/RANDU, 2017, [Online; accessed 22-July-2017].

[11] Wikipedia, Mersenne Twister, https://en.wikipedia.org/wiki/Mersenne_Twister, 2017, [Online; accessed 22-July-2017].

[12] M. Matsumoto and T. Nishimura, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).

[13] J. D. Chodera, Journal of Chemical Theory and Computation **12**, 1799 (2016), PMID: 26771390.

[14] M. R. Shirts, D. L. Mobley, J. D. Chodera, and V. S. Pande, The Journal of Physical Chemistry B **111**, 13052 (2007), PMID: 17949030.

[15] JPL, RELEASE 99-113: MARS CLIMATE ORBITER TEAM FINDS LIKELY CAUSE OF LOSS, https://mars.nasa.gov/msp98/news/mco990930.html, 2017, [Online; accessed 22-July-2017].