The recommendation is that students should develop their code in four phases: (i) a code that can carry out Monte Carlo simulations for the Lennard-Jones liquid (a model of liquid argon). In a second phase (ii), this code should be made more efficient through the use of neighbour lists. The third problem to be addressed is (iii) simulation of liquid silicon using the Stillinger-Weber model potential. Finally (iv), this too should be made more efficient by using neighbour lists. All the required theory and all the key equations needed are provided as a document that is part of this manual. This document should be read with care, in particular because it also specifies more precisely exactly what the expected functionality of the code should be. Students who only accomplish the first one, two or three of the above coding tasks will be able to pass the unit, though with a lower grade than those who complete the whole set of tasks.

**Embrace debugging!**

**Debugging is an essential part of coding and has been ever since the early days (see https://www.computerhistory.org/tdih/september/9/). In fact, despite all the accumulated expertise related to programming techniques, in many (or most!) projects, more time will be spent on debugging code than on writing new code. So don't get frustrated if your code does not work the first time, and instead learn to enjoy the challenge! There are many ways to debug, but they all rely on the same basic techniques: looking at what has changed since your code last worked, checking that the code actually does what you intend it to do (it can be very instructive simply to check that variables contain the things that you think they should, for which purpose you can add lots of 'print' statements that output variables at key points in the code; you can also use an integrated development environment that allows you to check the content of variables during execution), and that it flows in the way you think it should (a print statement saying "I just entered function xxxxx" can be really revealing, if it turns out that your code never actually does enter that function....). Some people prefer to use IDEs (https://en.wikipedia.org/wiki/Integrated_development_environment) for code development; these do indeed streamline some of the debugging procedure, so they can be quite helpful (though for a medium project like this one, my view is that they are not *necessary*). If you cannot solve a bug after some period of time, consider that you may have done something conceptually wrong, so look back at tip 3 on planning.**

# Writing a Code to Model Atomic Liquids Through Monte Carlo Simulation

Jeremy Harvey

August 24, 2022

**Abstract**

This document lays out the basic background needed to construct a simple Monte Carlo simulation code for atomic liquids, using both the simple Lennard-Jones model for non-bonded liquids (e.g. liquid argon) and the Stillinger model for liquids involving directional bonding, such as liquid silicon. To speed up simulation, a neighbour list will be used. The structure of the liquids will be probed by computing the radial distribution function $g(r)$. All the basic equations needed to perform the corresponding coding are provided for both cases.

## 1 Monte Carlo Simulations

Monte Carlo simulations provide a straightforward way to generate ensembles of structures for a given system at a given temperature $T$. The basic idea is to make random trial changes in structure, and to accept or reject these changes using as criterion the Boltzmann distribution for the corresponding temperature. Consider two structures $\mathbf{r}^{(i)}$ and $\mathbf{r}^{(j)}$ of a system with $N$ atoms. We use here the vector notation $\mathbf{r}^{(i)}$ to refer to a particular structure adopted by the system, comprising all $3N$ coordinates of the particles $1 \ldots N$ in the system for that given structure, i.e. $\mathbf{r}^{(i)} \equiv \{\mathbf{r}_1^{(i)}, \mathbf{r}_2^{(i)}, \ldots \mathbf{r}_N^{(i)}\}$. In this notation, $\mathbf{r}_k^{(i)}$ refers to the three Cartesian coordinates of the $k$-th atom for the $i$-th structure of the whole system. At a given temperature $T$, the relative probability of observing the two structures should be given by:

$$\frac{\mathcal{P}(\mathbf{r}^{(i)})}{\mathcal{P}(\mathbf{r}^{(j)})} = \exp\left(-\frac{V(\mathbf{r}^{(i)}) - V(\mathbf{r}^{(j)})}{k_B T}\right) \tag{1}$$

Where $V(\mathbf{r}^{(i)})$ is the potential energy for that structure. The above relation can be satisfied by choosing a method to update the structure of the system such that the ratio of the probabilities for changing from structure $\mathbf{r}^{(i)}$ to structure $\mathbf{r}^{(j)}$, $\mathcal{P}(\mathbf{r}^{(i)} \to \mathbf{r}^{(j)})$ and for the reverse change from $\mathbf{r}^{(j)}$ to structure $\mathbf{r}^{(i)}$ is also given by the Boltzmann factor:

$$\frac{\mathcal{P}(\mathbf{r}^{(j)} \to \mathbf{r}^{(i)})}{\mathcal{P}(\mathbf{r}^{(i)} \to \mathbf{r}^{(j)})} = \exp\left(-\frac{V(\mathbf{r}^{(i)}) - V(\mathbf{r}^{(j)})}{k_B T}\right) \tag{2}$$

The Monte Carlo method relies on updating the structure of the system being simulated in a way that satisfies the condition of eq. 2. Specifically, in your code, you will implement the method as described here:

(a) Choose an initial structure $\mathbf{r}^{(0)}$ (the way to do this will be described below), then repeatedly:

(b) Generate a random change in the structure, $\Delta\mathbf{r}$. In the jargon of Monte Carlo simulation, this is called a **trial move**. Generating the trial move can in principle be done in many different ways, for example it can involve a change in position of all atoms in the system, of several atoms or of just one atom. In most cases, the most efficient approach involves a <u>displacement of just *one*</u> atom and this is what your code should do in the present case. While many procedures to generate the trial moves can be used, they must satisfy the following important requirement: the probability of generating a given trial move that would convert $\mathbf{r}^{(i)}$ into $\mathbf{r}^{(j)}$ must be the same as the probability of generating the reverse trial move, that would convert $\mathbf{r}^{(j)}$ into $\mathbf{r}^{(i)}$.

(c) Calculate the change in potential energy of the system associated with the change, $\Delta V(\Delta\mathbf{r})$. It is the *change* in energy that is important – the algorithm does not require the potential energy corresponding to the initial structure $V(\mathbf{r}^{(i)})$ or of the modified structure, $V(\mathbf{r}^{(i)} + \Delta\mathbf{r})$, and it may be more efficient to calculate the difference rather than both overall energies.

(d) Decide whether or not to accept the change in structure, based on the value of the energy change that would result from this change, using the following rules:

   i) If $\Delta V(\Delta\mathbf{r})$ is negative or zero, the change should be accepted.

   ii) If $\Delta V(\Delta\mathbf{r})$ is positive, then one should compute the Boltzmann factor $\alpha = \mathcal{P}(\Delta\mathbf{r}) = \exp(-\Delta V(\Delta\mathbf{r})/k_B T)$. This is close to 1 if the energy change is small (or the temperature high), and close to 0 for larger energy changes or lower temperature.

   iii) One then chooses a random number $w$ from a uniform distribution between 0 and 1.

   iv) If $\mathcal{P}(\Delta\mathbf{r}) \geq w$ then the change in structure should be accepted. This is more likely to occur if the Boltzmann factor is close to 1.

   v) If however $\mathcal{P}(\Delta\mathbf{r}) < w$ then the change in structure should be rejected.

   Note that the procedure described here for deciding whether to accept the move or not means that the probability of *accepting* a given trial move is certainly **not** symmetric. Instead, it obeys eq. 2. In point (b) above, it is specified that the procedure for *generating* trial moves must be symmetric – but the procedure for accepting them is not.

(e) If the change is accepted, the structure should be updated, with $\mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} + \Delta\mathbf{r}$. If the change is rejected, one keeps the current structure, $\mathbf{r}^{(i+1)} = \mathbf{r}^{(i)}$.

As mentioned above, the steps (b) to (e) need to be repeated many times, after which the series of structures $\{\mathbf{r}^{(i)}\}$ with $i = 1, \ldots n_{\text{step}}$ can be analyzed and used to make predictions concerning the properties of the system. In case the initial structure $\mathbf{r}^{(0)}$ is a very unlikely structure, it may be wise to discard some initial proportion of structures $\mathbf{r}^{(0)} \ldots \mathbf{r}^{(n_{\text{eq}})}$ of the sampled structures, in order to perform an *equilibration* of the system, allowing it to relax to a more typical structure. The number of steps $n_{\text{eq}}$ needed for this typically needs to be chosen by trial and error. Given that in most cases only one atom moves in a given trial move, on average, after having considered $N$ trial moves (this is sometimes call a 'sweep' of trial moves), then a proportion $\overline{\alpha}$ of the atoms will have changed position, where $\overline{\alpha}$ is the mean *success rate* of the trial moves, and lies between 0 and 1. In general, in order to obtain an efficient Monte

Carlo simulation, the trial moves need to be selected in such a way that $\overline{\alpha}$ is not too small (otherwise, very few changes in structure are performed), but $\overline{\alpha}$ should not be too large either, as that would indicate that the trial moves being considered are not large enough, so one is not rapidly covering the range of possible structures. As a ballpark, a value of $\overline{\alpha}$ of the order of 1/3 to 1/2 will represent the best compromise between these two requirements.

For simple atomistic systems like those treated here, the trial move $\Delta\mathbf{r}$ is usually selected by picking four random numbers, $k$, and $w_x$, $w_y$ and $w_z$.[1] $k$ is an integer between 1 and $N$, and the factors $w_\zeta$ are evenly distributed between 0 and 1. Atom $k$ is moved, with its $x, y$ and $z$ coordinates being changed by respectively $2\Delta r_{\max}(w_x - 1/2)$, $2\Delta r_{\max}(w_y - 1/2)$ and $2\Delta r_{\max}(w_z - 1/2)$, i.e. by up to $\Delta r_{\max}$ in either direction. The scalar quantity $\Delta r_{\max}$ defines the maximum change that can occur for each coordinate. Choosing a larger value will typically cause $\overline{\alpha}$ to be smaller, while a smaller value of $\Delta r_{\max}$ will typically lead to a larger value of $\overline{\alpha}$. The procedure described here trivially satisfies the requirement for the probability of a move $\Delta\mathbf{r}$ to be equal to that for the reverse move $-\Delta\mathbf{r}$, since movements in each direction are equally likely to be generated.

# 2  Periodic Boundary Conditions

The present exercise aims at simulating atomic liquids. Liquids have no structure on longer lengthscales, but at short range, the atom-atom interactions lead to noticeable structure. To be meaningful, simulations must be performed on systems with sufficient atoms such that the size of the system is larger than the lengthscale on which the structure manifests itself. For liquids, this is typically some small multiple of the typical nearest-neighbour atom distance, so some tens of Å. In principle, the system to be modelled could therefore be a sphere with diameter of this order of magnitude. However, in the case of a spherical system, unless it is quite large (diameter above 100 Å), most of its atoms will be at the surface, which is not advantageous because surface atoms tend to behave differently from those in the bulk. Accordingly, many simulations, including those that you will perform with your code, use so-called **periodic boundary conditions** with the **minimum-image convention**, as illustrated in fig. 1. The idea is as follows: the simulation system comprising $N$ atoms (here, four atoms, labelled $A - D$) is surrounded by an infinite number of replicas of itself in each direction. In the simplest case, shown in fig. 1, each replica has a square or cubic shape with length $L$ (there are 8 close neighbours in two dimensions, but 26 in three dimensions), though more complex periodic shapes are also possible. The length of the repeating cell $L$ must be chosen such that it is significantly larger than the characteristic lengthscale of the problem at hand – so in our case, $L$ must be at least a few tens of Å.

---

[1]The art of generating random numbers in a computer is one which plays a very important role in Monte Carlo simulations. There are a number of more or less sophisticated algorithms available to generate *pseudo-random* numbers (numbers that appear to be random but are generated in a deterministic way). Many of these algorithms can be initiated with a *seed* – a starting parameter which, if if the same value is used twice when running the code twice, will lead to *the same* sequence of 'random' numbers being generated. This can be useful for debugging purposes. For production, the seed can itself be chosen randomly or can be taken from some extrinsic property such as the clock time, so that each run generates independent results. Some of the simpler algorithms for random number generation yield results that are un-random enough that artefacts can result in the code using them; examples in the case of molecular Monte Carlo simulation are described in Ghersi, Parakh and Mezei, *J. Comput. Chem.*, 2017, **38**, 2713 – 2720 (https://doi.org/10.1002/jcc.25065), and in the papers referred to in that work. The standard random number generators available with the gfortran Fortran compiler and the Julia programming language are however reported to yield no noticeable artefacts.
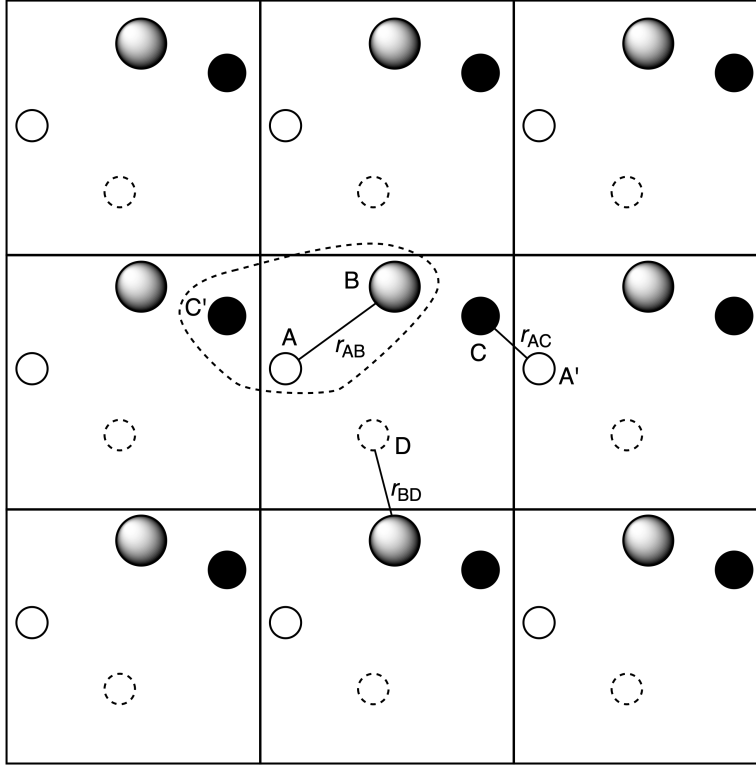
Figure 1: Schematic representation of periodic boundary conditions and the minimum-image convention in two dimensions.

In order to calculate the energy of the system, one effectively needs to take into account interactions between *all* pairs of atoms in the infinite system formed by all of the replicas. For some types of interaction, especially Coulombic interactions, this requires sophisticated approaches relying on Fourier transforms in order to evaluate the infinite number of terms that arise. Here, however, we will only be considering quite short-ranged interactions, and they can be treated using cut-offs and the *minimum-image convention*. The use of cut-offs implies that atoms are considered to interact with one another, i.e. to contribute to the overall energy expression, only if their distance is smaller than some cut-off distance $r_c$. This can be illustrated in terms of the energy expression of eq. 3, which includes two-body and three-body terms $V^{(2)}$ and $V^{(3)}$ (extrapolation to more complicated energy expressions is straightforward but not needed for this exercise). The two-body term for a given pair of atoms depends only on the distance $r_{ij}$ between them, while the three-body term as written here depends on the distances $r_{ij}$ and $r_{ik}$ (in general the three-body term could also depend on the distance $r_{jk}$ but this is assumed not to be the case in eq. 3). Coupled to the use of cut-offs is the fact that in the minimum-image convention, when considering the $V^{(2)}$ contribution of two atoms (e.g. atoms A and C in fig. 1, one chooses only the interaction between the two *closest* atoms chosen with one atom in the central periodic image. As shown in the figure, for $r_{\mathrm{AB}}$, the shortest A–B distance involves two atoms both within the central image, but for $r_{\mathrm{AC}}$, a shorter distance is obtained when atom C is in the central image, while the periodic image A' is considered for atom A. Likewise, the three-body term $V^{(3)}_{ABC}$ for this hypothetical system would involve considering the circled trio of atoms C', A and B, as these yield the shortest distances $r_{AB}$ and $r_{AC}$.

4

$$\begin{cases} V = \sum_{i,j} V_{ij}^{(2)}(\mathbf{r}_i, \mathbf{r}_j) + \sum_{i,j,k} V_{ijk}^{(3)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) \\ |r_{ij}| > r_c \Rightarrow V_{ij}^{(2)} = 0 \\ |r_{ij}| > r_c \quad \text{or} \quad |r_{ik}| > r_c \Rightarrow V_{ijk}^{(3)} = 0 \end{cases} \quad (3)$$

Periodic boundary conditions with the minimum-image convention can be easily implemented in code: when computing the energy for a normal finite system, one typically needs to refer to a set of interatomic vectors $\mathbf{r}_{ij}$, which can be obtained from the Cartesian coordinates as shown in eq. 4. In the minimum-image convention, one still considers each pair of atoms $(i, j)$ within the central image, but $\mathbf{r}_{ij}$ is replaced by the corresponding minimum-image convention vector $\mathbf{r}_{ij}^{\mathrm{MI}}$. In the simplest case of a cubic periodic image, which you will use here, this vector is constructed as shown in eq. 5. In this equation, $n_x$, $n_y$ and $n_z$ are integers, chosen so that each of the quantities $\alpha_j - \alpha_i + n_\alpha L$ ($\alpha = x, y, z$) falls in the interval between $-L/2$ and $L/2$.

$$\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i = \left\{ \begin{array}{c} \Delta x_{ij} \\ \Delta y_{ij} \\ \Delta z_{ij} \end{array} \right\} = \left\{ \begin{array}{c} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{array} \right\} \quad (4)$$

$$\mathbf{r}_{ij}^{\mathrm{MI}} = (\mathbf{r}_j - \mathbf{r}_i)^{\mathrm{MI}} = \left\{ \begin{array}{c} \Delta x_{ij}^{\mathrm{MI}} \\ \Delta y_{ij}^{\mathrm{MI}} \\ \Delta z_{ij}^{\mathrm{MI}} \end{array} \right\} = \left\{ \begin{array}{c} x_j - x_i + n_x L \\ y_j - y_i + n_y L \\ z_j - z_i + n_z L \end{array} \right\} \quad (5)$$

In a simulation using periodic boundary conditions, one often makes the choice that if an atom $i$ moves "out of" the primary simulation image, i.e. if one of its Cartesian coordinates $\alpha_i > L$ or $\alpha_i < 0$ (or equivalently if $\alpha_i > L/2$ or $\alpha_i < -L/2$), then it is moved "back into the box" on the "other side" by replacing $\alpha_i$ by $\alpha_i + L$ or $\alpha_i - L$ as needed, so that one always has $0 \le \alpha_i \le L$ (or $-L/2 \le \alpha_i \le L/2$). This makes visualization easier, but is not strictly necessary provided that eq. 5 is applied, i.e. provided that whenever $\mathbf{r}_{ij}$ (or the distance $r_{ij}$) appears in the energy expression, it is replaced by $\mathbf{r}_{ij}^{\mathrm{MI}}$ (or the corresponding distance $r_{ij}^{\mathrm{MI}}$). Certainly the choice of whether to enforce $0 \le \alpha_i \le L$ or an equivalent convention whereby $-L/2 \le \alpha_i \le L/2$ is entirely a matter of personal preference.

# 3   Potential Energy Models for Atomic Liquids

For any system to be modelled, one must first choose a method to compute the potential energy for a given set of atomic coordinates. This can be done with quantum-chemical methods such as density functional theory. Another increasingly popular choice is a machine-learning method trained using quantum-chemical methods. In this work, which only addresses some quite simple systems, we will use simple empirical expressions (primitive molecular mechanics forcefields) because we want to be able to carry out calculations using very limited computational resources. For the systems we will study, the expressions we will use actually give quite good results – it is not always the case that more expensive computations yield better results.

## 3.1   The Lennard-Jones Fluid

The first system to be modelled here is a model for liquids comprised of rare gas atoms such as argon. The potential energy for such systems can be calculated with various *ab initio* methods, and it is found that the overall energy can be expressed quite accurately as a sum of two-body terms ($V^{(2)}$ in eq. 3). Very accurate results do require three-body correction terms, but these

will be neglected here. It is found that the standard Lennard-Jones expression, eq. 6 and fig. 2, provides a fairly accurate model for $V^{(2)}$:

$$V_{ij}^{\mathrm{LJ}} = 4\epsilon \left\{ \left( \frac{\sigma}{r_{ij}} \right)^{12} - \left( \frac{\sigma}{r_{ij}} \right)^6 \right\} \tag{6}$$
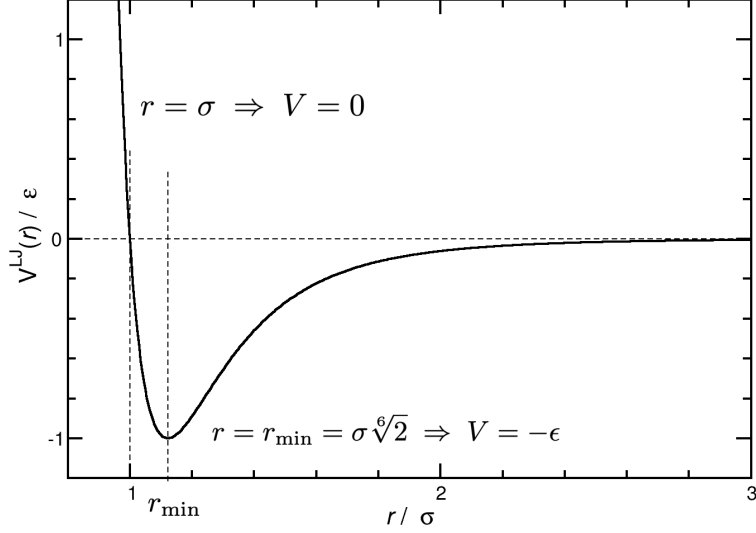


Figure 2: The two-body Lennard-Jones potential.

For a system with many argon atoms, one then has the following expression (where the distances $r_{ij}$ should be interpreted as minimum-image convention distances $r_{ij}^{\mathrm{MI}}$, the length of the vector $\mathbf{r}_{ij}^{\mathrm{MI}}$ of eq. 5 if one is using periodic boundary conditions):

$$V^{\mathrm{LJ}} = 4\epsilon \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left\{ \left( \frac{\sigma}{r_{ij}} \right)^{12} - \left( \frac{\sigma}{r_{ij}} \right)^6 \right\} \tag{7}$$

The parameters $\epsilon$ (well depth) and $\sigma$ (radius of the "hard" core of the atom) can be given physical values. For example, for argon, values of 3.4 Å ($\sigma$) and 1.0 kJ mol$^{-1}$ ($\epsilon$) can be used. However, for the purposes of running simulations, there is no need to assign specific values to these parameters, one can instead run the simulations using so-called *reduced variables*, whereby the unit of energy is $\epsilon$, the unit of distance $\sigma$, the unit of mass is $m$, the mass of the particles, and the unit of temperature is such that the Boltzmann constant $k_B = 1$. By making these choices one implicitly also chooses reduced units for other quantities. The density $\rho$ simply becomes the same as the number density $\rho = N/V$, and the volume of the simulation cell $V$ becomes $V' = V/\sigma^3$, so that the number density in reduced units can be expressed as $\rho^* = \rho\sigma^3$. To take an example, considering argon at its boiling point (85.5 K in SI units), and using a value of $\epsilon$ of 1.0 kJ mol$^{-1}$, this means that in reduced units, the boiling point $T_b^* = k_B T/\epsilon = 0.727$ in reduced units.

As mentioned above, in periodic boundary conditions, one needs to use a cut-off for the potential. The Lennard-Jones potential of eqs. 6 and 7 falls off rapidly for larger $r_{ij}$, so using a cut-off is not a problem. For many purposes, using $r_c = 2.5\sigma$ is sufficient and this value (or a slightly higher one, between $r_c = 3\sigma$ and $4\sigma$) can be used here.

6

The Lennard-Jones potential of eq. 7 is exactly additive and involves only two-body terms. This means that when computing the change in energy for a trial move, $\Delta V(\Delta \mathbf{r})$, it is not necessary to compute the whole potential. If atom $k$ is moved from $\mathbf{r}_k^{\text{init}}$ to $\mathbf{r}_k^{\text{trial}}$, then only the terms involving that atom need to be considered, yielding eq. 8 for $\Delta V(\Delta \mathbf{r})$ (using reduced variables). The single summation in this equation, that replaces the double sum in eq. 7, makes it much faster to evaluate.

$$\Delta V(\Delta \mathbf{r}_k) = 4 \sum_{i=1; i \neq k}^{N} \left\{ \left( \frac{1}{r_{ik}^{\text{trial}}} \right)^{12} - \left( \frac{1}{r_{ik}^{\text{trial}}} \right)^{6} \right\} - \left\{ \left( \frac{1}{r_{ik}^{\text{init}}} \right)^{12} - \left( \frac{1}{r_{ik}^{\text{init}}} \right)^{6} \right\} \tag{8}$$

When performing Monte-Carlo simulations on the Lennard-Jones fluid, several suitable choices for the initial structure $\mathbf{r}^{(0)}$ can be made. One option is simply to place the atoms at random in the periodic box, i.e. one generates $N$ triplets $(m_x, m_y, m_z)$ of random numbers $m$ uniformly distributed between 0 and 1, and then chooses $\mathbf{r}_k^{(0)} = (m_x L, m_y L, m_z L)$ for each of the atoms. This approach works well at low density, but at higher density, the chance that two atoms will be very close to one another increases, and this means the initial structure will be very high in potential energy. This can cause the simulation to take some time to equilibrate. A modified option is to generate coordinates for the atoms one at a time. The first atom is placed randomly as above, then for the following atoms random positions are proposed, but before accepting them, a test is made to check what is the shortest distance to a previously positioned atom. If this shortest distance falls below some threshold $r_{\text{min}}$ (which could be of the order of 0.9 in reduced units), this proposed position is rejected and a new one is generated. Again this approach works well at lower density, but for higher density (especially if $r_{\text{min}}$ is large), one may need many proposed structures for the later atoms in order to find a suitable position. A third option is to place $M^3$ atoms on a cubic grid, where $M$ is the first integer larger than $\sqrt[3]{N}$. The atoms are placed on a cubic grid with dimension $M \times M \times M$, at $(L/M, L/M, L/M)$, $(L/M, L/M, 2L/M)$ and so on – and then $M^3 - N$ unneeded atoms can be selected at random and deleted.

# 4 The Radial Distribution Function

In order to compare the outcome of simulations with experiment or with other simulations, one needs to have some property to base the comparison on. This property could in principle be the mean energy or some other thermodynamic function, but the easiest property to compare is the mean *structure* in the form of the radial distribution function $g(r)$. This function is defined as the ratio of the density of atoms found at a given distance $r$ from a reference atom, $\rho(r)$, and the bulk density $\rho$:

$$g(r) = \frac{\rho(r)}{\rho} \tag{15}$$

This function can readily be evaluated during a simulation. One defines a histogram $N_i(r_i)$ in which one records how many times during the simulation two atoms are separated by a distance falling in the interval $[r_i, r_i + \Delta r)$, where $\Delta r$ is some small value (small enough to generate a smooth $g(r)$ function, but large enough that sampling does not require too many steps). The histogram is typically accumulated from $r_0 = 0$ to some maximum value $r_{\max}$, where it should be noted that when using periodic boundary conditions, the value of $g(r)$ becomes meaningless for $r > L/2$, so $r_{\max}$ should certainly be smaller than this. The histogram can be updated after every sweep (or every few sweeps) of Monte Carlo steps (updating after every step is not useful, since most atoms do not move for a given Monte Carlo step), by computing all atom-atom distances $r_{ij}$ (with $j > i$) and incrementing the histogram appropriately, keeping track of the number of times $N_{\mathrm{accum}}$ that the histogram has been updated. After the simulation has finished, one can generate the radial distribution function by comparing the number of atom pairs found to have had distances in each interval $[r_i, r_i + \Delta r)$ with that expected for the given density (the volume of the shell with thickness $\Delta r$ is $4/3\ \pi(r_i + \Delta r)^3 - 4\pi r^3$). In practice,

one needs to use the following expression (the factor $N/2$, with $N$ the number of atoms, is present because one has included all atom-atom distances $r_{ij}$ (with $j > i$) when generating the histogram).

$$g(r_i) = \frac{N_i}{N_{\mathrm{accum}} \times N/2 \times 4/3\,\pi((r_i + \Delta r)^3 - r_i^3)\rho} \qquad (16)$$

For reference and help with debugging, Figure 3 shows the computed radial distribution function for both systems to be studied, the Lennard-Jones fluid and the Stillinger-Weber model of liquid silicon, with typical simulation conditions. The corresponding data files are also available.
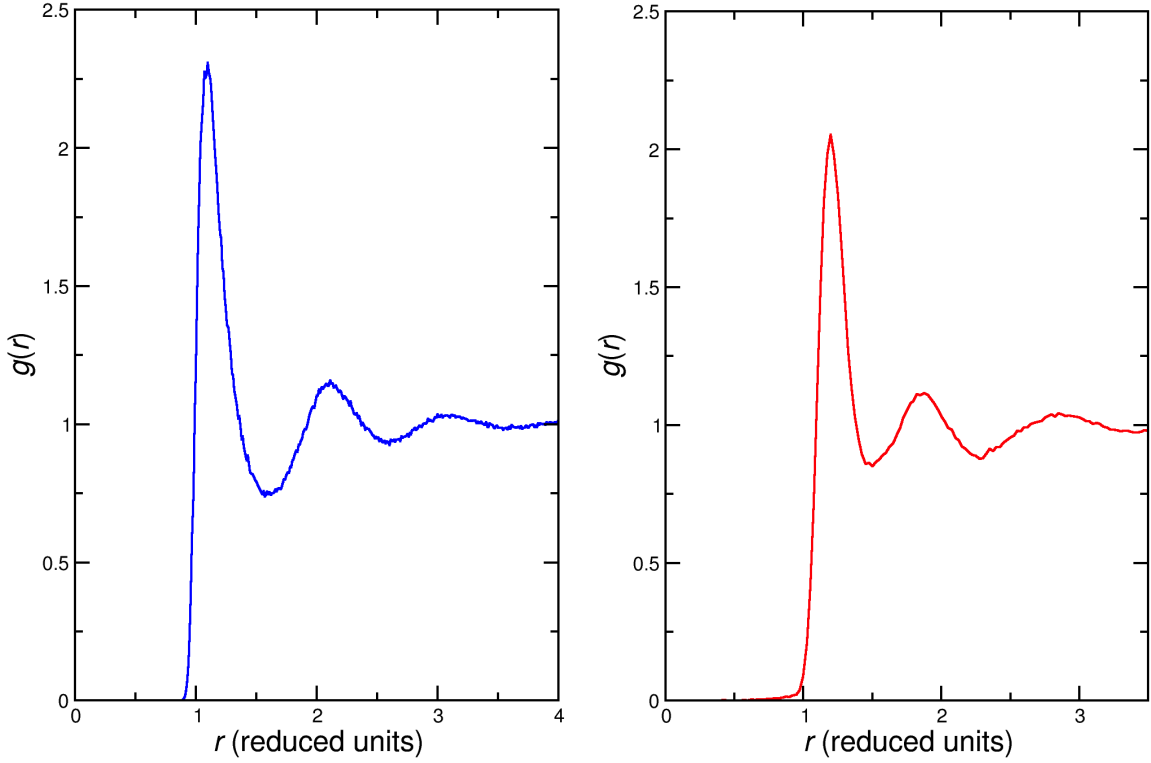


Figure 3: Sample radial distribution functions $g(r)$ for (left) the Lennard-Jones fluid, and (right) the Stillinger-Weber model of silicon. Conditions used: LJ fluid: 729 atoms, $r_c = 3.0$, $\rho^* = 0.7$, $L = 10.14$, $T^* = 1.2$, 2000 sweeps for equilibration, 500 for production, initial structure: regular cubic lattice, maximum move: 0.2, $g(r)$ sampled from $r = 0$ to $r = 4.0$, at intervals $\Delta r = 0.01$. Stillinger-Weber model: 216 atoms, $L = 7.6461$ reduced units ($\rho^* = 0.4832$), $r_{\mathrm{skin}} = 0.4$, $T^* = 0.10$, $\Delta r_{\mathrm{max}} = 0.15$, 2500 equilibration sweeps, 2000 sweeps for production, initial structure: random, with a minimum interatomic distance of 0.9, $g(r)$ sampled from $r = 0$ to $r = 3.5$, at intervals $\Delta r = 0.025$.

# 5 Efficient Coding: Neighbour Lists

Evaluating the energy expressions of eqs. 7 or 9, or the energy change expressions of eq. 8 (or the equivalent more complicated ones for the Stillinger potential, eqs. 19, 20 and 21 below) at first sight requires performing a loop over all pairs or triples of atoms in the system. Hence the computational expense of each step might be expected to scale as $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$, making the

calculations quite demanding for larger $N$. As already mentioned, in Monte Carlo simulations, only the energy terms involving the atom which is being moved in the trial move actually need to be considered, meaning that only a single loop (Lennard-Jones) or a double loop (Stillinger) is needed, reducing the scaling to $\mathcal{O}(N)$ or $\mathcal{O}(N^2)$. Still, this can be quite computationally demanding if the system contains many atoms. It is also inefficient, since many of the atoms will be positioned at a distance from the reference atom greater than the cut-off distance $r_c$, and hence make no contribution to the energy.

Careful program design can to some extent alleviate this problem. For example, for each pair of atoms $i$ and $j$ considered, one can first compute the square[2] of the distance between them (using the minimum-image convention), $(r_{ij}^{\mathrm{MI}})^2$, and then test to see if this is larger than $r_c^2$, in which case one moves on to the next atom without performing any additional operations. In this way, the full computational expense is incurred only for those atoms that are close to one another, and only the value $(r_{ij}^{\mathrm{MI}})^2$ needs to be computed for all pairs (and $(r_{ij}^{\mathrm{MI}})^2$ and $(r_{ik}^{\mathrm{MI}})^2$ for triples $(i, j, k)$). Even so, this remains unnecessarily computationally expensive, as the computation of $(r_{ij}^{\mathrm{MI}})^2$ does not have negligible computational expense. For a fluid with number density $\rho^*$, on average only $\rho^* 4\pi r_c^3/3$ atoms will be within $r_c$ of a given atom, and this is usually much smaller than $N$. Hence the quantity $(r_{ij}^{\mathrm{MI}})^2$ will need to be evaluated many times without making any contribution to the energy. For example, for $\rho^* = 0.7$, for a simulation cell with $L = 10$, there will be a total of 700 atoms, of which only just over 10% (79), on average, will be closer than $r_c$ to a given atom, if $r_c = 3\sigma$.

Many simulation codes accordingly make use of advanced techniques to increase efficiency, such as the use of *neighbour list*. These are lists $\mathbf{N}^L(j) \equiv \{k\}_j$, generated for each atom $j$ in the system at regular intervals during the simulation, and containing the atom numbers $k$ of each of the atoms that have a distance $r_{kj}^{\mathrm{MI}}$ to atom $j$ that is less than some threshold $r_n$. We can write that:

$$r_{kj}^{\mathrm{MI}} < r_n \Rightarrow k \in \mathbf{N}^L(j) \quad \& \quad j \in \mathbf{N}^L(k) \tag{17}$$

Once the lists $\mathbf{N}^L$ are available, one simply needs to loop over the atoms in the appropriate list, rather than over all atoms, when evaluating the energy in the case of the Lennard-Jones potential. For the case of the Stillinger potential, one also loops over neighbour lists rather than over all atoms – the details of which atoms to include in the loops are discussed below. Note that the threshold $r_n$ must be at least equal to the cut-off distance. In fact, it needs to be *larger* than the cut-off distance, because otherwise the neighbour list would need to be reevaluated after each atomic motion. Choosing a distance $r_n = r_c + r_{\mathrm{skin}}$ as shown in fig. 4, with $r_{\mathrm{skin}}$ some distance such that the atoms occupying the 'skin' between the sphere of radius $r_c$ and that of radius $r_n$ are those which are likely to move closer than $r_c$ to the central atom within the next few Monte Carlo moves, enables one to update the neighbour list only every few moves. The decision when to trigger the reevaluation of the neighbour list can be taken by looking to see how much each atom has moved since the list was created. In detail, one needs to store the positions of all atoms at the moment when the neighbour list is created, then after each successful Monte Carlo step, one can compute the total distance (using the minimum image convention) that this atom has moved since the neighbour list was created. In case this distance is larger than $r_{\mathrm{skin}}/2$, the neighbour list must be refreshed. A larger $r_{\mathrm{skin}}$ means that there are more atoms in each neighbour list, so evaluating the potential takes longer, but also means that the neighbour list needs to be renewed less frequently, so there is a trade-off in

---

[2]For the Lennard-Jones potential, one should only ever calculate the square of interatomic distances – the potential of eq. 6 can be evaluated in terms of the square, so computing $r_{ij}$ via taking the square root of $r_{ij}^2$ is a waste of computational time.

terms of efficiency. In any case, $r_{\mathrm{skin}}$ needs to be more than twice as large as the maximum length of a Monte Carlo move $\Delta r_{\mathrm{max}}$, because otherwise a single move could bring an atom to within less than $r_c$ of the central atom.
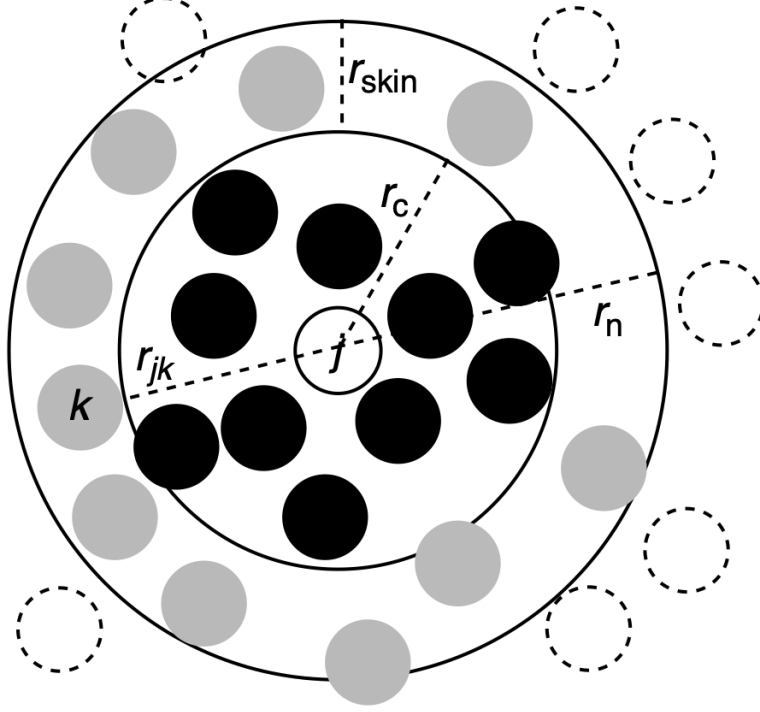


Figure 4: Principle behind the use of a neighbour list. The atoms shown as black circles have $r_{jk} < r_c$, while the grey atoms have $r_{jk} < r_n$. The atoms shown as dashed circles have $r_{jk} > r_n$ and are not included in the neighbour list.

In detail, it is recommended that the neighbour list be stored in an integer array of dimensions $n_{\mathrm{max}} \times N$ in Fortran (or $N \times n_{\mathrm{max}}$ in C++). $n_{\mathrm{max}}$ is the maximum number of expected neighbours per atom, and should be some multiple of $\rho 4\pi r_n^3/3$ – perhaps twice or three times, in order to allow for regions of higher density within the whole. As well as storing the identities of the atoms that are closer than $r_n$ in this array, one needs a one-dimensional integer array to store the number of neighbours that each atom has, and a double precision array of dimension $3 \times N$ to store the atom positions at the moment the neighbour list was generated.

For the Lennard-Jones case, eq. 7 suggests that the neighbour list for atom $i$ would only need to contain the atoms $j$ with $j > i$, but given that the Monte Carlo code will typically use instead eq. 8, in fact, both $j > i$ and $j < i$ neighbours should be stored. Then eq. 8 can be applied by looping only over the atoms within the neighbour list of the moved atom (but still applying the cutoff, for those atoms with a distance of more than $r_c$ but less than $r_n$):

$$\Delta V = \sum_{j \in \mathbf{N}^L(i)} \left\{ \left( \frac{1}{r_{ij}^{\mathrm{trial}}} \right)^{12} - \left( \frac{1}{r_{ij}^{\mathrm{trial}}} \right)^6 \right\} - \left\{ \left( \frac{1}{r_{ij}^{\mathrm{init}}} \right)^{12} - \left( \frac{1}{r_{ij}^{\mathrm{init}}} \right)^6 \right\} \tag{18}$$

For the Stillinger case, the neighbour list should likewise contain all neighbours.

The change in energy upon moving atom $i$ from position $\mathbf{r}_i^{\mathrm{init}}$ to $\mathbf{r}_i^{\mathrm{trial}}$ can be obtained as the sum of the change in the two-body and three-body terms of eq. 9, as follows:

$$\Delta V_2 = \sum_{j \in \mathbf{N}^L(i)} f_2(r_{ij}^{\mathrm{trial}}) - \sum_{j \in \mathbf{N}^L(i)} f_2(r_{ij}^{\mathrm{init}}) \tag{19}$$

For the three-body term, one has:

$$\Delta V_3 = V_3^i(\mathbf{r}^{\text{trial}}) - V_3^i(\mathbf{r}^{\text{init}}) \tag{20}$$

Where the partial three-body energy associated with atom $i$, $V_3^i$, is given by the expression below. The first summation includes all three-body terms in which atom $i$ is the central atom. The second summation includes all three-body terms in which an atom $j$ in the neighbour list of atom $i$ is the central atom.

$$V_3^i = \sum_{j,k \in \mathbf{N}^L(i), j<k} h(r_{ij}, r_{ik}, \theta_{jik}) + \sum_{j \in \mathbf{N}^L(i)} \sum_{k \in \mathbf{N}^L(j), k \neq i} h(r_{ji}, r_{jk}, \theta_{ijk}) \tag{21}$$