Practical Molecular Dynamics with Matlab

2019

Dr Thomas Rodgers

Contents

List of Figures

Nomenclature

Roman		
$\mathbf{a}$	Acceleration	$\mathrm{m\,s^{-2}}$
A	Helmholtz energy	J
C_p	Heat capacity at constant pressure	$\mathrm{J\,mol^{-1}\,K^{-1}}$
C_v	Heat capacity at constant volume	$\mathrm{J\,mol^{-1}\,K^{-1}}$
E	Energy	J
$\mathbf{F}$	Force	N
G	Gibbs free energy	J
$mathcalH$	Classical Hamiltonian	–
H	Enthalpy	J
k	Boltzmann constant	$1.38 \times 10^{-23}\,\mathrm{m^2\,kg\,s^{-2}\,K^{-1}}$
KE	Kinetic energy	J
m	Mass	kg
N	Number of particles	–
$\mathbf{p}$	Momentum	$\mathrm{kg\,m\,s^{-1}}$
E_{bend}	Energy for bending an angle between three atoms	J
E_{el}	Electrostatic energy	J
E_{str}	Energy for stretching a bond between two atoms	J
E_{tors}	Energy for torsion of an angle around a bond	J
E_{VdW}	Van der Waals energy	J
E_{Xterms}	Coupling energy terms	J
P	Pressure	Pa
$\mathbf{r}$	Position	
S	Entropy	$\mathrm{J\,K^{-1}}$
T	Temperature	K
t	Time	s
U	Internal energy	J
$\mathbf{V}$	Potential energy	J
$\mathbf{v}$	Velocity	$\mathrm{m\,s^{-1}}$
V	Volume	$\mathrm{m^3}$
Greek		
β	Thermodynamic beta	$1/kT$
ϵ	Lennard-Jones energy	J
Λ	Thermal de Broglie wavelength	m
σ	Lennard-Jones length	m
σ	Standard deviation	–

Course Information

These notes provide a short practical course on molecular dynamics. Although it provides a quick recap, it is assumed that the user has a basic understanding of classical mechanics, statistical mechanics, and thermodynamics, but knows very little on methods for atomistic simulations. The basics of molecular dynamics are explained in terms of practical code writing.

No attempt is made to address advanced topics in molecular dynamics, such as calculation of free energies and non-equilibrium simulations, while sum important subjects are treated superficially. This handbook is meant to be a starting point, but does cover all key material for this course. Many books are devoted to computer simulations and you are advised to make use of them to find more information[1,2,3].

These notes contain example functions written for Matlab and are also available to download from blackboard. As part of the course you will be using these functions and creating some of your own to produce a working molecular dynamics program. While Matlab is a very uncommon language to use for scientific programming of this type due to its speed, it has been selected for this course as it is a very clear language, and it has been used in previous modules.

In practical situations for molecular dynamics a full code would be used which are generally freely available online, e.g. Gromacs, DL_POLY, etc. However, as these are highly optimised they are difficult to read as code, and to understand these programs it is important to have a good understanding of the basics of molecular dynamics.

[1] M. P. Allen and D. J. Tildesley. 2017. Computer Simulation of Liquids. Oxford University Press; 2 edition.

[2] D. Frenkel and B. Smit. 2001. Understanding Molecular Simulation: From Algorithms to Applications. Academic Press; 2 edition.

[3] A. Leach. 2001. Molecular Modelling: Principles and Applications Paperback. Prentice Hall; 2 edition.

Chapter 1

Background to Molecular Dynamics

Contents

1.1 Introduction

Computer experiments play an important role in science. Traditionally the physical sciences were characterised by linking experiments with theory. In experiments, a system is subjected to measurements, and results, expressed in numeric form are obtained. In theory, a model of the system is constructed, usually in the form of a set of mathematical equations. The model is then validated by its ability to describe the system.

In the past, theoretical models could only be easily tested in a few simple circumstances. For example, a model for intermolecular forces in a specific material could be verified in a diatomic molecule, or in a perfect crystal. Even then, approximations were often required to carry out the calculation. Most problems of interest fall outside of these simple circumstances.

The advent of high speed computers allowed a new joint element to be developed between experiments and theory; the computer experiment. In a computer experiment, a model is still provided by theory, but calculations are carried out by the computer following a standard procedure as in an experiment. These computer simulations altered the traditional relationship between theory and experiment in three key way. The first is, the demand for more accurate models. For example, molecular dynamics simulations allow the evaluation of the melting temperature of a material; but for this accurate interaction laws are needed. The second is, that simulations can come very close to experimental conditions, to the extent that then can be compared with experimental results. When this happens, simulations become a powerful tool to interpret the experiments at the molecular level. The third is, that computer simulations allow experiments all the investigation of things that are just impossible (or too expensive) to do in reality, but whose outcome greatly increases our understanding of a phenomena.

One of the computer experimental tools that is commonly used is molecular dynamics. Molecular dynamics is the imitation of the operation of a real-world process or system over time. It is one of the most widely used tools for decision making, usually performed on a computer with appropriate software. Molecular dynamics can be an analysis/descriptive tool (can answer what if questions) or a synthesis/prescriptive tool (can develop new systems). Simulations are generally applied to complex systems that are impossible to solve mathematically.

1.2 Molecular Dynamics

Molecular Dynamics simulation is a technique for computing the equilibrium and transport properties of a classical many-body system. In this context, classical means that the trajectory of the system is calculated by the solution of the classical equations of motion [1]. This is an excellent approximation for a wide range of materials, and since MD is time dependant it has the distinct advantage over Monte Carlo of being useful to measure time-dependant quantities.

In a Molecular Dynamics simulation, we select a model system consisting of N particles and we solve Newton's equations of motion, for each of the particles, i, for this system until the properties of the system no longer change with time (get to an equilibrium);

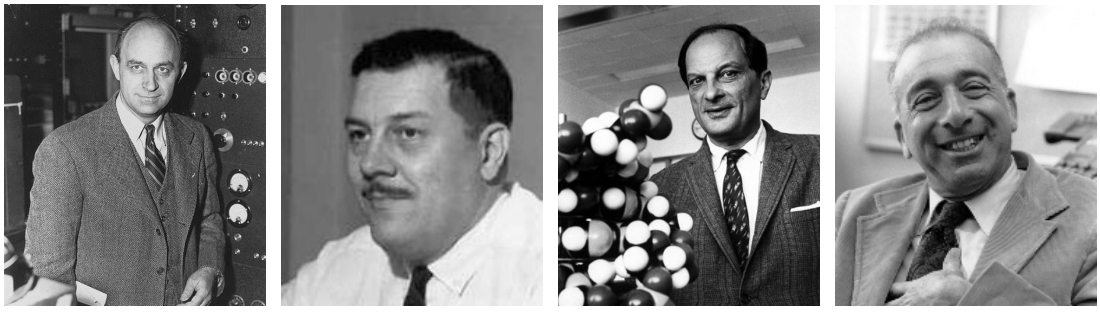
$$\mathbf{F}_i = m_i \mathbf{a}_i \tag{1.2.1}$$

Here m_i is the particle mass, $\mathbf{a}_i$ its acceleration, and $\mathbf{F}_i$ is the force acting upon it due to the interactions with other atoms. After equilibrium, we can then perform an actual measurement on the system. Therefore, in contrast with Monte Carlo methods, molecular dynamics is a deterministic technique: given an initial set of positions and velocities, the subsequent time evolution is in principle[1] completely determined.

The computer calculates a trajectory in a $6N$-dimensional phase space ($3N$ positions and $3N$ momenta). However, such trajectory is usually not particularly relevant by itself. Molecular dynamics is a statistical mechanics method, and it is a way to obtain a set of configurations distributed according to some statistical distribution function, or statistical ensemble. According to statistical physics, physical quantities are represented by averages over configurations distributed according to ensembles. A trajectory obtained by molecular dynamics provides such a set of configurations. Therefore, measurements of a physical quantity by simulations is simply obtained as an arithmetic average of the various instantaneous values assumed by that quantity during the simulation. Statistical mechanics is the link between the microscopic behaviour and the thermodynamics properties. More information is given in sections 1.3 and 1.4.

1.2.1 History

The techniques for Molecular Dynamics we first developed during the 1940s and 50s at the Los Alamos National Lab in California, where computers became avaliable as a fringe benefit of weapons work. Modelled on celestial mechanics, with molecules represented by mass points interacting with central forces, these calculations led to rapid advances in both equilibrium and nonequilibrium systems [2]. Figure 1.1 shows some of the key developers of Molecular Dynamics. These methods were then further developed by Alder and Wainwright in the late 1950's [1, 3] to study the interactions of hard spheres. Many important insights concerning the behavior of simple liquids emerged from their studies. The next major advance was in 1964, when Rahman carried out the first simulation using a realistic potential for liquid argon [4]. The first molecular dynamics simulation of a realistic system was done by Rahman and Stillinger in their simulation of liquid water in 1974 [5].



(a) (b) (c) (d)

Figure 1.1: Key developers of Molecular Dynamics (a) Enrico Fermi, (b) John Pasta, (c) Stanislaw Ulam, and (d) Berni Alder.

Although simulations of liquids are where molecular dynamics started, these are still

[1] In practice, the finiteness of the integration time step and arithmetic rounding errors will eventually cause the computed trajectory to deviate from the true trajectory

investigated now. Availability of new interaction models allows the study of new systems, and non-equilibrium techniques allow the study of transport phenomena such as viscosity and heat flow. Other key areas studied with molecular dynamics (though not an exhaustive list) are: Defects in crystals, fracture of solids, physics of surfaces, adhesion and friction between materials, surfactants and liquid crystal properties, and biomolecules.

1.2.2 Classical Mechanics

The molecular dynamics simulation method is based on Newton's second law for the equation of motion, $F = ma$, where F is the force exerted on the particle, m is its mass and a is its acceleration. From a knowledge of the force on each atom, it is possible to determine the acceleration of each atom in the system. Integration of the equations of motion then yields a trajectory that describes the positions, velocities and accelerations of the particles as they vary with time. From this trajectory, the average values of properties can be determined. Molecular dynamics simulations can be time consuming and computationally expensive. However, computers are getting faster and cheaper.

Newton's equations of motion for a simple system are given by equation 1.2.2, $\mathbf{r}_i$ is the atomic coordinates and m_i is the mass of particle i, t is time, and $\mathbf{V}$ is the potential energy.

$$m_i \frac{\partial^2 \mathbf{r}_i}{\partial t^2} = -\nabla \mathbf{V}_i \tag{1.2.2}$$

The velocity of each particle is therefore given by equation 1.2.3, the acceleration by equation 1.2.4, and the kinetic energy by equation 1.2.5.

$$\mathbf{v}_i = \frac{\mathrm{d}\,\mathbf{r}_i}{\mathrm{d}\,t} \tag{1.2.3}$$

$$\mathbf{a}_i = \frac{\mathrm{d}\,\mathbf{v}_i}{\mathrm{d}\,t} = \frac{\mathbf{F}_i}{m_i} \tag{1.2.4}$$

$$KE_i = \frac{m_i}{2}\mathbf{v}_i \cdot \mathbf{v}_i \tag{1.2.5}$$

In a classical many-body system, we can define the temperature using the equipartition of energy over all degrees of freedom, such that,

$$\left\langle \frac{1}{2} m_i \mathbf{v}_i^2 \right\rangle = \frac{1}{2} k_B T \tag{1.2.6}$$

In a practical simulation, we would measure the total kinetic energy of the system and divide this by the number of degrees of freedom, $N_f = 3N - 3$ for a system with N particles[1]. As the velocity of the particles fluctuates, so does the instantaneous temperature (from equation 1.2.6),

$$T(t) = \sum_{i=1}^{N} \frac{m_i \left|\mathbf{v}_i(t)\right|^2}{k_B N_f} = \frac{1}{3Nk_B} \sum_{i=1}^{N} \frac{\left|\mathbf{p}_i(t)\right|}{2m_i} \tag{1.2.7}$$

The first thing we need to build our molecular dynamics simulation is a model for the real systems, this is the focus of Chapter 2.

[1] With a large number of particles this is essentially $3N$.

1.3 Statistical Mechanics

When dealing with computer simulations only a small number of molecules are simulated. Chemical thermodynamics on the other hand deals with bulk properties of matter, typically of the order of 10^{23} particles. There should be some link between these two sets of particles. If 10^{23} particles are modelled then the number of interactions, positions, and velocities would be astronomical. Even if this were possible then a method to relate all of the molecular information to the bulk properties would be needed.

The pressure exerted by a gas on the container wall depends on the rate at which the particles collide with the wall. It is not necessary to know which particle underwent a particular collision. What is needed is the root mean square speed of the particles and their standard deviation about this mean. This is where statistical thermodynamics is useful. If the pressure exerted on the walls is measured many times at regular time intervals then the mean, $\langle P \rangle$, and standard deviation, σ_P, can be given by equation 1.3.1.

$$\begin{aligned} \langle P \rangle &= \frac{1}{n} \sum_{i=1}^{n} P\left(t_i\right) \\ \sigma_P &= \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(P\left(t_i\right) - \langle P \rangle\right)^2} \end{aligned} \tag{1.3.1}$$

The larger the number of samples the closer the sample mean will be to the true mean and the smaller the sample deviation will become.

Consider an ensemble of N^* cells, comprising the original cell and $N^* - 1$ replications. Energy may flow between the cells but the total ensemble energy is constant. Suppose that the possible total energies of the N particles contained in each cell are E_1^*, E_2^*, and so on. If a snapshot is taken there is a distribution of energies amongst the cells as follows:

N_1^* cells have energy E_1^*,

N_2^* cells have energy E_2^*, etc.

According to Boltzmann, the E^* and the N^* are related by equation 1.3.2.

$$\frac{N_i^*}{N^*} = \frac{\exp\left(-\frac{E_i^*}{kT}\right)}{\sum_i \exp\left(-\frac{E_i^*}{kT}\right)} \tag{1.3.2}$$

The denominator is called the partition function and is generally given by Q as in equation 1.3.3, this is a simplification and strict ensemble partition function is dependent on the thermodynamic ensemble as shown in Section 1.4.

$$Q = \sum_i \exp\left(-\frac{E_i^*}{kT}\right) \tag{1.3.3}$$

Some useful thermodynamic properties are:

Helmholtz energy:	$A = U - TS$
Gibbs free energy:	$G = H - TS = A + PV$
Enthalpy:	$H = U + PV$
Entropy:	$S = -\left(\frac{\partial A}{\partial T}\right)_{N,V}$
Pressure:	$P = -\left(\frac{\partial A}{\partial V}\right)_{N,T}$
Constant volume heat capacity:	$C_v = \left(\frac{\partial U}{\partial T}\right)_{N,V}$
Constant pressure heat capacity:	$C_p = \left(\frac{\partial H}{\partial T}\right)_{N,P}$

1.4 Thermodynamic Ensembles

Statistically speaking, the method by which macroscopic properties can be obtained from a simulation is by the averaging of (in theory) an infinite number of microscopic configurations, each of which can exist with certain probability. In statistical mechanics, this collection of micro-states is usually referred to as an ensemble. The ensemble also describes the probability of seeing each region in phase space with an accompanying partition function. Considered together, this distribution of micro-states can be used to model a system, averaging the properties of a large number of micro-states to find properties which could be correlated to the thermodynamic state of a real system. In practice however, it is impossible to simulate an infinite number of states, and so averages must be calculated from a finite sample of micro-states. Each ensemble can be defined from a real life experimental situation with particular rules and limits. This allows each ensemble of states to be connected with a thermodynamic property, which can be calculated as a function of the partition function.

1.4.1 Canonical (NVT)

The canonical ensemble, describes a system which cannot exchange mass (i.e. number of molecules N) or volume V with the outside world, but exchanges energy such that the temperature T of the system is uniform and constant. The canonical partition function, which describes the distribution of energies of the microstates is given by an expression taking into account both the kinetic and potential energies, equation 1.4.1, where Λ is the thermal de Broglie wavelength.

$$\begin{aligned} Q_{\text{NVT}} &= \frac{1}{N!\Lambda^{3N}} \int \mathrm{d}\,\mathbf{p}^N \mathrm{d}\,\mathbf{r}^N \exp\left[-\beta \mathcal{H}\left(\mathbf{p}^N, \mathbf{r}^N\right)\right] \\ &= \sum_i \exp\left(-\frac{E_i}{kT}\right) \end{aligned} \tag{1.4.1}$$

The relationship between the partition function and the thermodynamic potential of the ensemble is the Helmholtz free energy, A, equation 1.4.2.

$$A = U - TS = -kT \ln Q \tag{1.4.2}$$

1.4.2 Grand-canonical (μVT)

The grand-canonical ensemble is similar to the canonical ensemble in the assumption of a system of constant temperature and volume which can pass energy into and out of the system. Additionally however, a grand canonical system may pass mass into and out of the system to keep the chemical potential, μ, constant. As a result the partition function is now also dependent on the number of particles, N, in the system, equation 1.4.3.

$$\begin{aligned} Q_{\mu\mathrm{VT}} &= \sum_{N=1}^{\infty} \frac{1}{N!\Lambda^{3N}} \exp\left(\beta N\mu\right) \int \mathrm{d}\,\mathbf{p}^N \mathrm{d}\,\mathbf{r}^N \exp\left[-\beta\mathcal{H}\left(\mathbf{p}^N, \mathbf{r}^N\right)\right] \\ &= \sum_i \exp\left(-\frac{E_i - \mu N_i}{kT}\right) \end{aligned} \tag{1.4.3}$$

The thermodynamic potential of the grand-canonical ensemble can be described by equation 1.4.4.

$$\Omega_{\mu\mathrm{VT}} = A - N\mu = -PV = -kT\ln Q \tag{1.4.4}$$

1.4.3 Microcanonical (NVE)

The micro-canonical ensemble keeps system mass and volume constant. Where it differs from all other ensembles is that the system energy is controlled by changing the temperature of the system. The NVE ensemble's main advantage was that it does not require random numbers. Since modern everyday computers are capable of running effective random number generators in negligible time, this ensemble is hardly ever used for molecular simulation. It can however be useful during the preparation of an initial configuration for a simulation run in a different ensemble.

1.4.4 Isothermal-Isobaric (NPT)

The isothermal-isobaric ensemble is used to describe a system in which the temperature and number of molecules are constant as in the canonical ensemble. However, the important feature of the NPT ensemble is that the pressure of the system is maintained to a constant value by changing the volume of the system. This ensemble often provides a more realistic simulation environment than the canonical and grand-canonical ensembles. The partition function, which includes a pressure-volume relationship PV is described by equation 1.4.5.

$$Q_{\mathrm{NPT}} = \frac{\beta P}{N!\Lambda^{3N}} \int_0^{\infty} \mathrm{d}\,\mathbf{V}\mathbf{V}^N \exp\left[-\beta P\mathbf{V}\right] \int \mathrm{d}\,\mathbf{p}^N \mathrm{d}\,\mathbf{r}^N \exp\left[-\beta\mathcal{H}\left(\mathbf{p}^N, \mathbf{r}^N\right)\right] \tag{1.4.5}$$

1.4.5 Other Ensembles

Isobaric-Isenthalpic (NPH)

This ensemble is characterised by a constant number of atoms, N, the pressure, P, and the enthalpy, H.

Gibbs Ensemble

The Gibbs ensemble is used to simulate phase coexistence, usually two liquid phases or a liquid and a vapour phase. This is accomplished by the use of two simulation boxes, each held at a constant temperature. The boxes may exchange both mass and volume between each other, but in such a way that there is no change in mass or volume for the overall system.

Semi-grand Canonical Ensemble

The semi-grand canonical ensemble keeps N_1, μ_2, V, and T constant. Its main use is for the study of solvent-solute systems, where the number of solvent molecules N_1 is kept constant, but the amount of solute N_2 is allowed to fluctuate, keeping μ_2 constant.

1.5 Simulation Time and Length Scales

When considering the simulation of a physical system using computers it is important to know what the intended output of the simulation is i.e. What is the purpose of this process. Once the purpose is known, usually to study a particular phenomena, the length scale on which this phenomena is likely to occur must be identified in order to recognise which type of simulation is most likely to produce relevant results. For instance a simulation technique where the finest detail is on the micrometre scale would be unlikely to yield useful results if studying the kinetics of a chemical reaction. Similarly a nanoscale simulation would be extremely inefficient for studying the dynamics of bulk fluids. Figure 1.2 shows a representation of simulation detail. Figure 1.3 shows examples of soft matter simulation techniques and their relevant time- and size-scales.

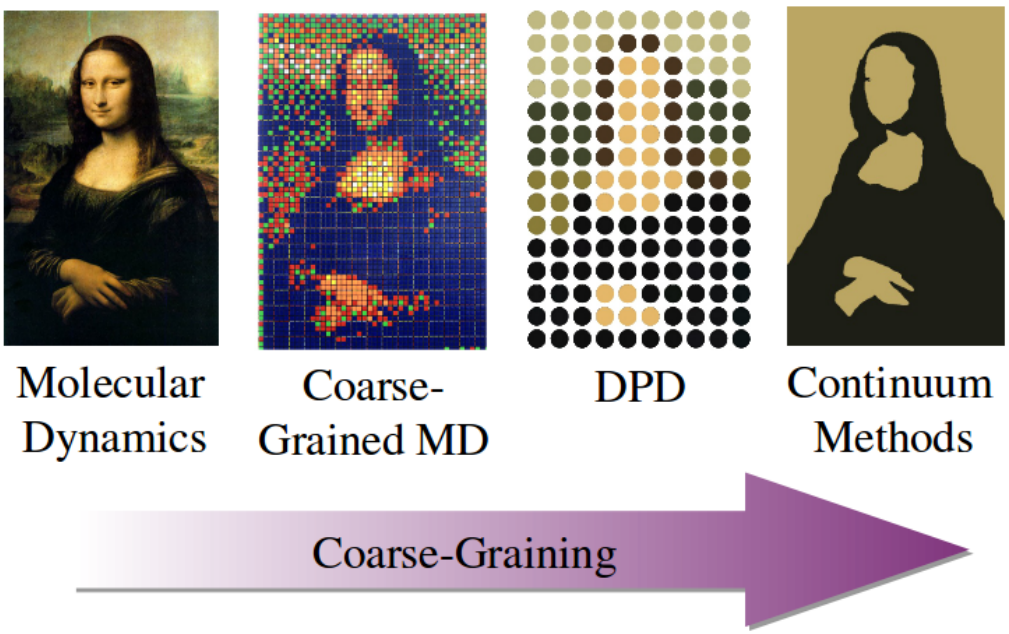


Figure 1.2: Schematic of simulation scales. The representation on the left has the most detail however would take a long time to paint, whereas the representation on the right still has the general features but would be a lot faster to produce.

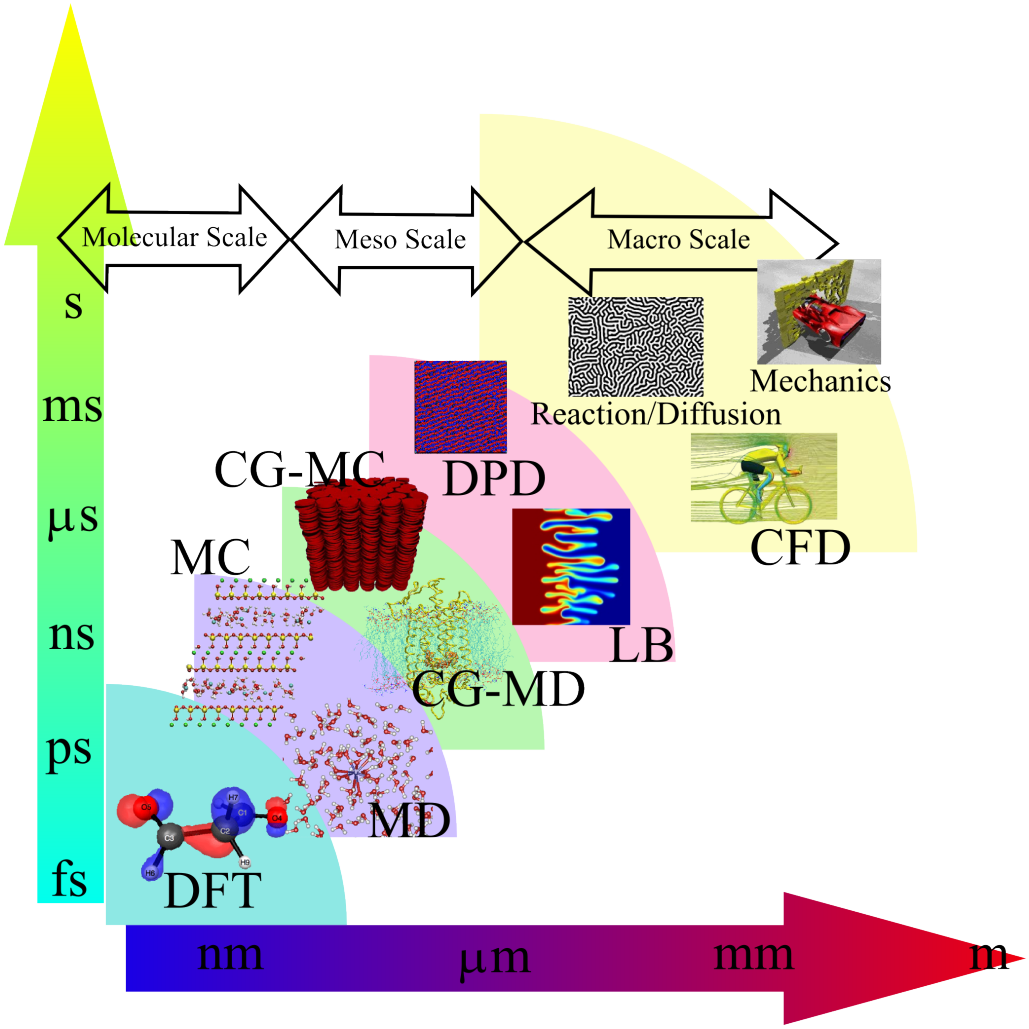


Figure 1.3: Examples of soft matter simulation techniques and their time- and size-scales.

1.6 References

[1] B. J. Alder and T. E. Wainwright. Studies in Molecular Dynamics. I. General Method. *Journal of Chemical Physics*, 31:459, 1959.

[2] E. Fermi, J. C. Pasta, and S. Ulam. Studies of Nonlinear Problems. Technical Report, LASL Report LA-1940, 1955.

[3] B. J. Alder and T. E. Wainwright. Phase Transition for a Hard Sphere System. *Journal of Chemical Physics*, 27:1208–1209, 1957.

[4] A. Rahman. Correlations in the Motion of Atoms in Liquid Argon. *Physical Review*, A136:405, 1964.

[5] F. H. Stillinger and A. Rahman. Improved Simulation of Liquid Water by Molecular Dynamics. *Journal of Chemical Physics*, 60:1545–1557, 1974.

Chapter 2

Molecular Interactions

Contents

2.1 Introduction

When molecules are near each other, they can influence one another. Therefore, the balance between the forces of attraction and repulsion needs to be understood. Such forces must exist in reality otherwise there would be nothing to bring molecules together to form liquid and solid states. People have speculated about these intermolecular forces since the concept of atoms and molecules first existed. The present theory is that molecules attract at long range but repel strongly at short range.

A system of interacting atoms is made up of nuclei and electrons which interact with each other. The true Hamiltonian for the system may be written as,

$$\mathcal{H} = \sum_i \frac{P_i^2}{2M_i} + \sum_n \frac{p_n^2}{2m} + \frac{1}{2}\sum_{ij} \frac{Z_i Z_j e^2}{|\mathbf{R}_i - \mathbf{R}_j|} + \frac{1}{2}\sum_{nn'} \frac{e^2}{|\mathbf{r}_n - \mathbf{r}_{n'}|} - \sum_{in} \frac{Z_i e^2}{|\mathbf{R}_i - \mathbf{r}_n|} \quad (2.1.1)$$

where indexes i, j run on the nuclei, n and n' on electrons, $\mathbf{R}$ and $\mathbf{P}$ are the positions and momenta of the nuclei, $\mathbf{r}$ and $\mathbf{p}$ of the electrons, Z is the atomic number of the nucleus, M its mass, m the electron mass, and e the unit charge. Coupled with the total wavefunction, $\Psi(\mathbf{R}_i, \mathbf{r}_n)$, calculated from the Schrödinger equation then everything about the system is known.

Of course, this is impossible to carry out in practice, and approximation schemes have to be employed. In 1923, Born and Oppenheimer noted that nuclei are much heavier than electrons, and thus move on a time scale which is about two orders of magnitude slower than that of the electrons. It is therefore sensible to think of the nuclei as fixed as far as the electronic part of the problem is concerned, thus factorize the total wavefunction as,

$$\Psi(\mathbf{R}_i, \mathbf{r}_n) = \Xi(\mathbf{R}_i)\,\Phi(\mathbf{r}_n; \mathbf{R}_i) \quad (2.1.2)$$

where $\Xi(\mathbf{R}_i)$ describes the nuclei, and $\Phi(\mathbf{r}_n; \mathbf{R}_i)$ the electrons (depending parametrically on the positions of the nuclei). With the assumptions, the problem is reformulated in terms of two separate Schrödinger equations,

$$\mathcal{H}_{el}\Phi(\mathbf{r}_n; \mathbf{R}_i) = V(\mathbf{R}_i)\,\Phi(\mathbf{r}_n; \mathbf{R}_i)$$
$$\text{where } \mathcal{H}_{el} = \sum_n \frac{p_n^2}{2m} + \frac{1}{2}\sum_{ij} \frac{Z_i Z_j e^2}{|\mathbf{R}_i - \mathbf{R}_j|} + \frac{1}{2}\sum_{nn'} \frac{e^2}{|\mathbf{r}_n - \mathbf{r}_{n'}|} - \sum_{in} \frac{Z_i e^2}{|\mathbf{R}_i - \mathbf{r}_n|} \quad (2.1.3)$$

and

$$\left[\sum_i \frac{P_i^2}{2M_i} + V(\mathbf{R}_i)\right] \Xi(\mathbf{R}_i) = E\Xi(\mathbf{R}_i) \quad (2.1.4)$$

Equation 2.1.3 is the electronic problem, considering the nuclei as fixed. The eigenvalue of the energy, $V(\mathbf{R}_i)$, will depend parametrically on the coordinates of the nuclei; we call this the interatomic potential. Once found, this quantity enters equations 2.1.4 which will give the motion of the nuclei. Note how in this equation there are no electronic degrees of freedom; it is all incorporated in $V(\mathbf{R}_i)$. For the nuclei, it is customary to replace this Schrödinger equation with a Newtonian equation[1].

[1] This is acceptable if quantum effects are small. For example light systems such as H_2 and He or systems at very low temperatures may have quantum effects and MD results should be interpreted with caution.

2.2 Molecular Force Fields

As actual interactions between atoms, interatomic potential, is difficult to calculate. For computational simulations, it is common practise to represent these real interactions as a computational force field. The choice of computational method is dependent on the size of the system and the time scales of calculations. For systems with up to several hundred to several thousand atoms, the molecules can be represented as individual atoms with each atom represented by a sphere. There are many types of force field, but they all follow a similar pattern:

- Nuclei and electrons are combined in an atom, represented by a ball.
- A ball has a radius and a constant charge. The ball can vary in softness.
- Bonds are represented by springs.
- Springs have an equilibrium length and can vary in stiffness.

The interactions between these spheres form the molecular force field. The interactions are described by pre-assigned parameters, that are system dependent. These parameters can be obtained from experiment and/or higher level computational calculations, such as *ab-initio* quantum mechanics. The force field is the set of parameters used in classical mechanical calculations. The potential energy of the system is described by the sum of all the interactions within the system, equation 2.2.1, where E_{str} is the energy function for stretching a bond between two atoms, E_{bend} is the energy required for bending an angle between three atoms, E_{tors} is the torsional energy for rotation around a bond, E_{VdW} is the Van der Waals energy, E_{el} is the electrostatic energy, and E_{Xterms} represents any coupling between the above terms.

$$E_{FF} = E_{str} + E_{bend} + E_{tors} + E_{VdW} + E_{el} + E_{Xterms} \tag{2.2.1}$$

The first three terms represent the bonded interactions. The electrostatic and Van der Waals energies are non bonded interactions between atoms. To be able to calculate the force field energy it is necessary to know atomic coordinates, geometries, and the relative energies of the atoms in the system.

The calculation of the total potential energy is dominated by the number of non-bonded interactions. The time for calculation of the bonded interactions increase on the order of $\sim N$, while the time for calculation of the non-bonded interactions increase on the order of $\sim N^2$.

2.3 Intra-Molecular Interactions

2.3.1 Stretching energy

The stretching energy of a bond is the energy of the extension of the spring, bonding two atoms, Figure 2.1. The shape of the energy curve is well described by a Taylor expansion, equation 2.3.1, where r_0 is an equilibrium bond length for the given system

and so corresponds to the minimal energy.

$$E_{str}\left(r^{AB}-r_0^{AB}\right)=E\left(r_0\right)+\frac{\mathrm{d}\,E}{\mathrm{d}\,r}\left(r^{AB}-r_0^{AB}\right) + \frac{1}{2}\frac{\mathrm{d}^2\,E}{\mathrm{d}\,r^2}\left(r^{AB}-r_0^{AB}\right)^2+\frac{1}{6}\frac{\mathrm{d}^3\,E}{\mathrm{d}\,r^3}\left(r^{AB}-r_0^{AB}\right)^3+\mathcal{O}\left(r^4\right) \quad (2.3.1)$$

At $r = r_0$, $E(r_0)$ is zero and is used as the zero point on the energy scale, the second term is a first order derivative that is also close to zero. The cubic term will drive the energy of the bond to negative value at infinity, so the formulation is often simplified to an harmonic interaction, equation 2.3.2, where k^{AB} is the force constant for stretching the bond between atoms A and B and will change depending on the system.

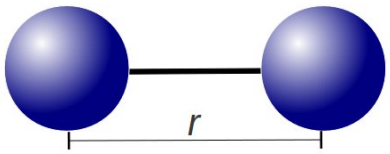


Figure 2.1: Model of diatomic molecule with bond length r_0.

$$E_{str}=\frac{1}{2}k^{AB}\left(r^{AB}-r_0^{AB}\right)^2 \quad (2.3.2)$$

Approximation to harmonic form is sufficient for most molecular dynamics calculations. For calculations of bond length dependent parameters, such as vibrational frequencies, or when the starting configuration is much different from equilibrium, terms up to the forth order may be included. The other common choice is the Morse potential [1], equation 2.3.3, where $\alpha=\sqrt{k/2D^{AB}}$ and D^{AB} is the bond dissociation energy, i.e. the depth of the well.

$$E_{str}=D^{AB}\left(1-\exp\left(-\alpha\left(r^{AB}-r_0^{AB}\right)^2\right)\right) \quad (2.3.3)$$

The Morse potential reproduces stretching well over a wide range of distances, at long distances tends to zero. Hence the Morse potential is slow in bringing atoms to equilibrium bond lengths, when the initial geometry has a large bond length. The comparison between harmonic and Morse potentials is shown in Figure 2.2.

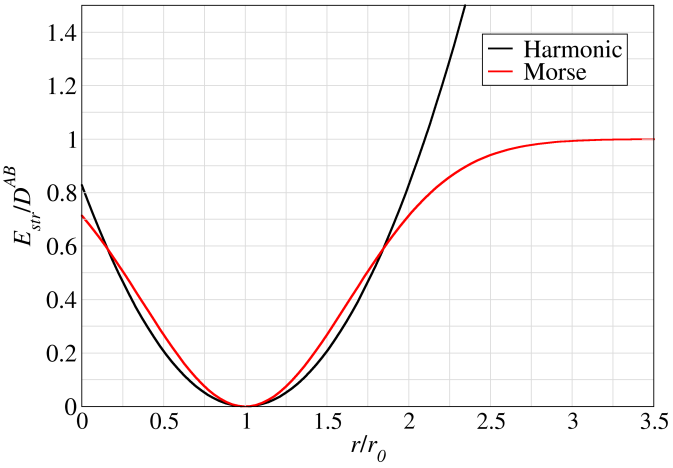


Figure 2.2: Bond stretching energy. Harmonic potential compared to the Morse potential.

The stretching energy has the highest contribution to overall energy of the system, followed by the bending energy.

2.3.2 Bending Energy

Similarly to the stretching energy, bending energy can also be expressed by a Taylor expansion, that can then be approximated to a harmonic potential, equation 2.3.4, where θ^{ABC} is an angle between three atoms *A*, *B* and *C*, as shown in Figure 2.3, and k^{ABC} is the force constant for the bending between three given atoms.

$$E_{bend} = \frac{1}{2}k^{ABC}\left(\theta^{ABC} - \theta_0^{ABC}\right)^2 \tag{2.3.4}$$

When a higher accuracy is required for the calculation, the next highest order term in the expansion can be included, or a series of harmonic potentials can be used.

Changes in the bending energy are usually smaller than the changes in stretching energy, so less energy is required to distort bond angles in comparison to bond length. Hence, force constants for bending are proportionally smaller than those for stretching.

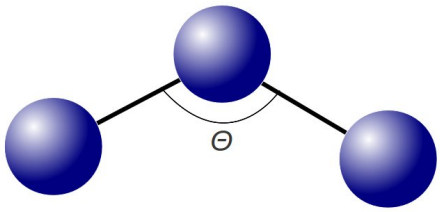


Figure 2.3: An angle between three bonded atoms.

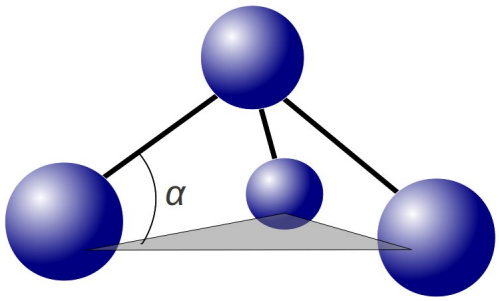


Figure 2.4: An illustration of out-of-plane bending, created by three atoms bonded to $sp2$ hybridised central atom.

When the central atom B is bonded to more than two other atoms, as shown on the Figure 2.4, a pyramidal structure is formed. This angle can no longer be calculated in the manner discussed above. A high force constant should be used to penalise out-of-plane bending, this will make planar structures very inflexible. By introducing a new term, the flexibility of the planar structure is maintained while out-of-plane movement is also taken into account. This bending is often referred to as the improper torsional energy. This also often takes the form of a harmonic potential, equation 2.3.5.

$$E_{improper} = \frac{1}{2}k^{B}(\alpha)^2 \tag{2.3.5}$$

2.3.3 Torsional Energy

Four aligned atoms frequently will rotate along the central bond, as shown in Figure 2.5.

Rotation along the bond is continuous, so when the bond rotates by $360°$ the energy should return to the initial value, making torsional energy periodic. The potential is well represented by the Fourier series, equation 2.3.6, where n is a periodicity term, depending on the allowed rotation it can be $n = 1$ for full $360°$ rotation, $n = 2$ for $180°$ periodicity, $n = 3$ for $120°$ periodicity, etc. V_n is a constant determining the barrier for the rotation,

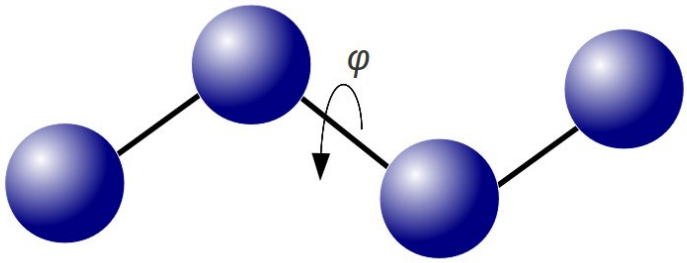


Figure 2.5: Torsional / dihedral angle, created by four linearly aligned atoms.

V_n is not equal to zero for allowed periodic rotations, φ is the torsional angle (also called the dihedral angle) and is shown on the Figure 2.5.

$$E_{tors} = \sum_n V_n \cos\left(n\varphi - \varphi_0\right) \tag{2.3.6}$$

Figure 2.6 shows a plot of the two torsional potentials with $n = 3$, $V_n = 4$ (dashed line), $n = 2$, $V_n = 3$ (solid line) and the corresponding total torsional potential in red if the two are combined.

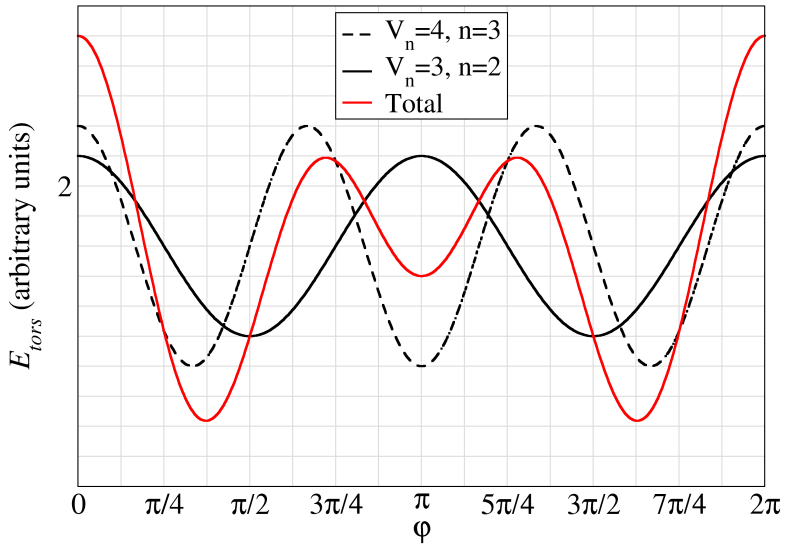


Figure 2.6: Variation of torsional potential for different values of n and V_n and the total potential (red).

2.3.4 Cross Terms

There are also cross terms available for some force fields. These can cover the coupling between the fundamental stretching, bending and torsional interactions.

2.4 Inter-Molecular Interactions

In addition to the bonded interactions described above, there are non bonded interactions in the system. These interactions appear between atoms in the same or neighbouring molecule and can be described by two potentials, representing the Van der Waals energy and the electrostatic energy.

2.4.1 Van der Waals

The Van der Waals energy describes the repulsion and attraction between non bonded atoms. At large interatomic distances the Van der Waals energy goes to zero, whereas at short distances it is positive and very repulsive. This mimics the overlap of the negatively charged electronic clouds of atoms. At intermediate distances there is a mild attraction between electronic clouds, due to electron-electron correlation. The attraction between two atoms arises because of an induced dipole–dipole moment created by the motion of electrons through the molecule, which effects the neighbouring molecule. The attraction between two fragments does not only depend on dipole–dipole interactions, but also on dipole–quadrupole, quadrupole–quadrupole etc. interactions. These interactions do not have a high contribution to the overall energy, so for simplicity they are neglected in calculations.

Hard-Sphere

Hard spheres are widely used as model particles in the statistical mechanical theory of fluids and solids. They are defined simply as impenetrable spheres that cannot overlap in space. They mimic the extremely strong repulsion that atoms and spherical molecules experience at very close distances. Hard spheres of diameter, σ, are particles with the following pairwise interaction potential:

$$V(\mathbf{r}_1, \mathbf{r}_2) = \begin{cases} 0 & \text{if } |\mathbf{r}_1 - \mathbf{r}_2| \geq \sigma \\ \infty & \text{if } |\mathbf{r}_1 - \mathbf{r}_2| < \sigma \end{cases} \tag{2.4.1}$$

Square-Well Potential

Square-Well potential is an update on the hard spheres model. They are defined simply as impenetrable spheres that cannot overlap in space, but they can have an attractive region around the hard sphere. For a spheres of diameter, σ, are particles with the following pairwise interaction potential:

$$V(\mathbf{r}_1, \mathbf{r}_2) = \begin{cases} 0 & \text{if } |\mathbf{r}_1 - \mathbf{r}_2| \geq \lambda\sigma \\ -V_i & \text{if } \sigma \leq |\mathbf{r}_1 - \mathbf{r}_2| < \lambda\sigma \\ \infty & \text{if } |\mathbf{r}_1 - \mathbf{r}_2| < \sigma \end{cases} \tag{2.4.2}$$

Where λ is the relative size of the attractive region. The sphere in the centre does not have to have an infinite repulsion, it can be set at a large positive value allowing some overlap of the spheres.

Mie Type Potentials

In 1903, Gustav Mie proposed an intermolecular pair potential[2], comprising two parts to represent attractive and repulsive forces, equation 2.4.3, where r is interatomic distance, $|\mathbf{r}_1 - \mathbf{r}_2|$, ϵ is the depth of the well at σ, the interatomic separation at which repulsive and attractive terms balance out and m, n can be adjusted.

$$V(\mathbf{r}_1, \mathbf{r}_2) = \left(\frac{n}{n-m}\right)\left(\frac{n}{m}\right)^{m/(n-m)} \epsilon \left[\left(\frac{\sigma}{r}\right)^n - \left(\frac{\sigma}{r}\right)^m\right] \tag{2.4.3}$$

There have been a number of models suggested, where m and n take different values and show an accurate representation of experimentally known interactions [3, 4]. Nevertheless, the most popular one is the 12–6 Lennard-Jones potential [5], equation 2.4.4, where σ is the interatomic separation at which repulsive and attractive terms balance out and ϵ is the depth of the well. The usage of 12 term reduces the amount of calculations needed, making it the lightest method available.

$$V^{\mathrm{LJ}}(\mathbf{r}_1, \mathbf{r}_2) = 4\epsilon^{AB}\left[\left(\frac{\sigma^{AB}}{r^{AB}}\right)^{12} - \left(\frac{\sigma^{AB}}{r^{AB}}\right)^{6}\right] \tag{2.4.4}$$

The 12–6 Lennard-Jones potential is shown on the Figure 2.7. Red shows the repulsive term, blue the attractive term and the overall potential is shown in black. Position of σ and ϵ is also shown on the graph.

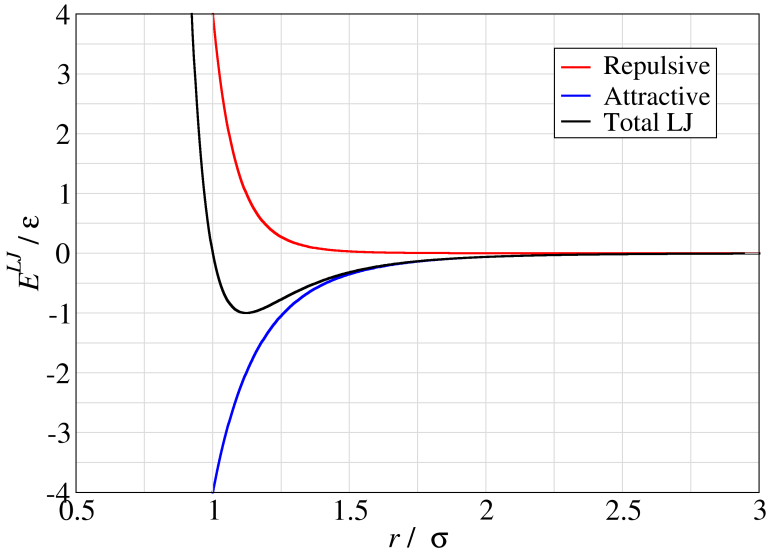


Figure 2.7: The Lennard-Jones Potential (black) is a combination of attractive (blue) and repulsive (red) terms.

? Activity

The energy between two particles can be given by the Lennard-Jones potential, equation 2.4.4. Produce an expresion for the force between these two particles. Using this expression write a program in matlab to calculate the force between two particles based on the distance between them and to plot this variation in force with distance.

Use the example system of Lennard-Jonesium, $\sigma = 3.405\,\text{Å}$ and $\epsilon/k_B = 119.8\,\text{K}$

. [6].

It is possible to mix two Lennard-Jones potentials for single type atoms to obtain a potential describing interaction between two different atoms. There are three main approaches:

- Arithmetic or Lorentz-Berthelot rules, equations 2.4.5 and 2.4.6 [7].

$$\sigma_{ij} = \frac{\sigma_{ii} + \sigma_{jj}}{2} \tag{2.4.5}$$

$$\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}} \tag{2.4.6}$$

- Geometric rules equation 2.4.7, with ϵ_{ij} calculated as given in equation 2.4.6 [8].

$$\sigma_{ij} = \sqrt{\sigma_{ii}\sigma_{jj}} \tag{2.4.7}$$

- Fender-Halsey rule, equation 2.4.8 [9].

$$\epsilon_{ij} = \frac{2\epsilon_i\epsilon_j}{\epsilon_i + \epsilon_j} \tag{2.4.8}$$

Buckingham Potential

A slightly more accurate potential, but up to 4 times more computationally demanding, is the Buckingham or Hill potential [4]. Repulsive forces take the form of an exponential, as the repulsion arises due to electron correlation, and the electron density reduces exponentially with the distance, equation 2.4.9, where A, B, and C are suitable constants.

$$V^{\text{Hill}}(\mathbf{r}_1, \mathbf{r}_2) = A\exp(-Br_{AB}) - Cr_{AB}^6 \tag{2.4.9}$$

2.4.2 Electrostatic Energy

The electrostatic energy calculates non bonded interactions that appear due to an uneven internal distribution of electrons. This leads to positively and negatively charged parts in molecule. The simplest way to model this behaviour is to place charges on atoms. The law of the interaction between two point charges was investigated by Charles Augustin Coulomb in 1780s and can be expressed by the Coulomb potential, equation 2.4.10, where ϵ_0 is the dielectric constant, the Q values are partial electronic charges on atoms A and B, and r is a distance between them.

$$V_{el}(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{4\pi\epsilon_0}\frac{Q^A Q^B}{r^{AB}} \tag{2.4.10}$$

The atomic charges are mainly taken from electrostatic potential calculations carried out using higher precision methods. Another approach arises from assigning bond dipole moments. This gives similar results to the partial charge method, but these two methods will only give identical results for interactions at larger distances.

2.4.3 Potential Truncation and Long-Range Corrections

The potentials given have an infinite range. In practical applications, it is customary to establish a cutoff radius, R_c, and disregard the interactions between atoms separated by more than R_c. This results in simpler programs and large savings of computer resources, because the number of atomic pairs separated by a distance r grows as r^2 and becomes quickly huge.

A simple truncation of the potential creates a new problem through; whenever a particle pair crosses the cutoff distance, the energy makes a little jump. A large number of these events is likely to disrupt the energy conservation in a simulation. To avoid this problem, the potential is often shifted in order to vanish at the cutoff radius[1]:

$$V(r) = \begin{cases} \phi_{LJ}(r) - \phi_{LJ}(R_c) & \text{if } r \leq R_c \\ 0 & \text{if } r > R_c \end{cases} \tag{2.4.11}$$

Commonly used truncation radii for the Lennard-Jones potential are 2.5σ and 3.2σ. Physical quantities are of course affected by the potential truncation. The effects of truncating a full-ranged potential can be approximately estimated by treating the system as a uniform continuum beyond R_c. For a bulk system this usually amounts to a constant additive correction.

2.5 Choice of Force Field

The correct choice of the force field is essential for performing an accurate simulation. A large number of force fields have been developed through last decades. The force field consists of a set of parameters that depend on a particular the atom and the interaction.

Ideally force fields are designed to be transferable between a number of molecular systems. Nevertheless, some are more suitable for particular systems and states, then others. Comparison of the performance of the force filed should be done with a specific study in mind. Typical tests include the comparison to the experimental physical properties, such as density, boiling and melting temperatures, secondary protein structure, as well as the agreement with vapour-liquid coexistence, and the reproduction of thermodynamic properties.

Most force fields represent all atom systems, with a few being united atom where hydrogens are united to the neighbouring heavy atoms. United atom representation has less interactions, by such speeding up the calculations. Not every system can be accurately represented as united atom, for example systems having hydrogen bonding, benzene rings, polar and charged systems.

All atom force fields:

- OPLS-AA(Optimised Potentials for Liquid Simulations) [10]
- CHARMM (Chemistry at HARvard Macromolecular Mechanics) [11]

[1] Shifting eliminates the energy discontinuity, but not the force discontinuity. Some researchers altered the form of the potential near R_c to obtain a smoother junction, but there is no standard way to do that.

- AMBER (Assisted Model Building with Energy Refinement) [12]
- UFF (Universal Force Field) [13]

United atom force fields:

- OPLS (Optimised Potentials for Liquid Simulations)[14]
- CHARMM (Chemistry at HARvard Macromolecular Mechanics) [11]
- GROMOS (GROningen MOlecular Simulation) [15]
- TraPPE (Transferable Potentials for Phase Equilibria) [16]

2.5.1 Universal Force Field (UFF)

UFF [13] was developed by Rappe *et. al.* in 1992. It is an unusual force field, unlike others it covers the full periodic table, from hydrogen to lawrencium. The parameters are obtained by estimating from given general rules, that convert individual parameters for the element into the intermolecular and intramolecular parameters for calculations.

The performance of this force field with respect to the experimental results [17], is not very consistency, so it is not really suitable for high accuracy condensed state calculations. This force field is however useful for calculations of exotic molecules, where other force field have not been created.

2.5.2 Assisted Model Building with Energy Refinement (AMBER)

AMBER is both a molecular dynamics package and a set of force fields [12, 18]. The force field uses 12-6 Lennard-Jones, equation 2.4.4, with parameters computed by Lorentz-Berthelot mixing rules, equations 2.4.5 and 2.4.6, 1-4 interactions are scaled by $\frac{1}{2}$ and Columbic interactions are represented by point charges and are scaled by $\frac{5}{6}$. Bond and angle interactions are expressed as harmonic potentials and torsional interactions are represented as a cosine series.

AMBER force fields are all atom only (though the newest force field, ff12sb, does include some united atom parameter), parametrised for biomolecular simulations and contains the 20 common amino acids as functional groups. More recently there have been some work put in to extend the AMBER force field to a wider range of systems through the development of the general AMBER force field (GAFF)[19].

2.5.3 Chemistry at HARvard Molecular Mechanics (CHARMM)

CHARMM is also molecular dynamics program [11], as well as a set of force fields. There are all atom CHARMM22 [20, 21], CHARMM27 [22], CgenFF [23], and united atom CHARMM19 [24] force fields. The potential function is of the same form as AMBER; however, non-bonded parameters are not scaled.

All of the force fields, apart from more recent CgenFF, were parametrised for biomolecular simulations, similarly to AMBER, containing the 20 common amino acid groups. CgenFF was created as a general force field for drug-type molecules, to accompany the rest of the force fields.

2.5.4 Optimised Potentials for Liquid Simulations (OPLS)

OPLS force field was developed by group of William L. Jorgensen and contains several sets of parameters, for both all atom and united atom calculations that can be mixed [10, 14]. The potential functions are the same as in AMBER, Lennard-Jones is computed by geometrical mean mixing rules, equation 2.4.7. Both, Lennard-Jones and Coulombic, parameters are scaled by a $1/2$.

OPLS at the beginning was parametrised not only for biomolecular systems, but also organic liquids. The parameters were optimized to fit experimental densities and heats of vaporisation of liquids, as well as gas-phase torsional profiles. The force field has being updated and many independent modifications to the force field have been developed [25, 26]. OPLS is, arguably, the most widely parametrised force field available.

2.5.5 GROningen MOlecular Simulation (GROMOS)

GROMOS is also both a molecular dynamics package [15] and the complementary force field. The first was also the base for GROMACS package [27]. GROMOS is a united atom force field, with ongoing improvements being done, currently most widely used version of the force field is GROMOS-96 [28] and its latest update [29]. This force field is also aimed at biomolecular calculations and is parametrised for proteins.

Unlike AMBER, the Lennard-Jones parameters for heteroatomic interactions are readily provided by the force field and should not be calculated via mixing rules. The force field uses a quartic expression for the bond length, a harmonic cosine potential for angle bending and a single cosine series for torsional angle potential.

2.5.6 Transferable Potentials for Phase Equilibria (TraPPE)

TraPPE force field evolved from the previous SKS (Smit-Karaborni-Siepmann) force field [30]. SKS was developed in Shell laboratories in the 1990s and is the first force field, aimed at reproducing properties of n-alkanes, the main components in fuels. The TraPPE force field contains parameters for alkanes, aromatics, some oxygen, sulphur and nitrogen containing compounds.

TraPPE force field is mainly united atom [16, 31–36] with two explicit hydrogen models [37, 38]. Unlike most of united atom force fields, in TraPPE hydrogen atoms are also united in aromatic interactions.

The functional form is the same as in AMBER, including mixing rules, Coulombic terms are scaled by a $\frac{1}{2}$. Since the force field was developed with Monte Carlo calculations, there is no improper torsional angle term what can lead to incorrect conformations when performing molecular dynamics. The force field is parametrised to fit the vapour-liquid coexistence curve for small organic molecules.

2.6 Water Models

Water molecular models have been developed in order to help discover the structure of water. They are useful given the basis that if the (known but hypothetical) model can

successfully predict the physical properties of liquid water then the (unknown) structure of liquid water is determined. There are two general forms for these models of water:

Explicit Water, where each water molecule is simulated directly.

Implicit Water, where the general effects of water are taken into account but the molecules are not directly simulated.

2.6.1 Explicit Water

When simulating water explicitly it is important to use an accurate but efficient model. This is due to the computational expensiveness of simulating water in a solvated system, e.g. in simulating a protein in periodic box full of water at aqueous density about 90% of the computational time is used in simulating water-water interactions.

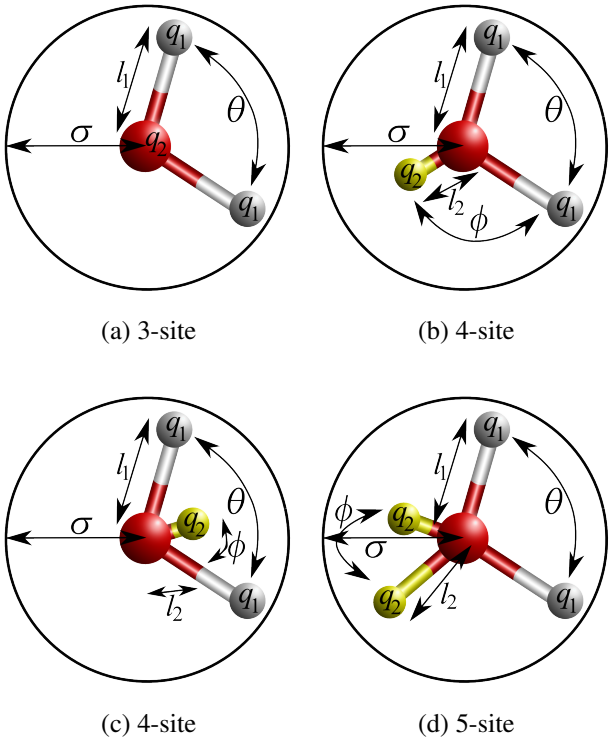
The original atomic-scale computational model for liquid water was proposed by Bernal and Fowler in 1933 [39]. This model took into account the position and the charge on the hydrogens and the position of the negative charge in the molecule. It was a highly insightful model and similar to the general view in Figure 2.8(c). At the time of the earliest developments of protein force fields the ST2 model of Stillinger and Rahman [40] was in wide use, which lead to the ST2 model being for some of the first protein simulations.

In the early 1980s two new force fields were developed, the SPC [41] and TIP3P [42] models. They involve orienting electrostatic effects and Lennard-Jones sites. The Lennard-Jones interaction accounts for the size of the molecules. It is repulsive at short distances, a ensuring that the structure does not completely collapse due to the electrostatic interactions. At intermediate distances it is significantly attractive but non-directional and competes with the directional attractive electrostatic interactions. This competition ensures a tension between an expanded tetrahedral network and a collapsed non-directional network (for example, similar to that found in liquid noble gases).

Since then many models for water have been created around these general principles, although the charge and Lennard-Jones sites don't always coincide. The four main general model types are shown in Figure 2.8. Generally each model is developed to fit well with one particular physical structure or parameter (for example, the density anomaly, radial distribution function or the critical parameters) and it comes as no surprise when a model developed to fit certain parameters it gives good compliance with these same parameters [43]. In spite of the heavy computational investment in the calculations, the final agreement with experimental data is often 'by eye' and not statistically tested or checked for parametric sensitivity. Also, tests for 'quality' often use the radial distribution fit with diffraction data in spite of the major fitted peaks being derived from the tetrahedral nature of water that is built into every model and overpowers any disagreement in the fine detail. In particular, the O—O radial distribution function seems to be a poor discriminator between widely differently performing models [44]. Table 2.1 shows the values predicted for some key water properties by some of the most popular water force fields.

2.6.2 Implicit Water

There are many circumstances in molecular modeling studies in which a simplified description of solvent effects has advantages over the explicit modelling of each solvent



(a) 3-site (b) 4-site

(c) 4-site (d) 5-site

Figure 2.8: Models for water molecules, (a) is a 3-site model, (b) and (c) are alternative 4-site model, and (d) is a 5-site model. Red spheres are oxygen atoms, white spheres are hydrogen atoms, and yellow spheres are dummy charge locations.

molecule. Most of these are where the saving of computer time are needed over using explicit water. Most of the simulation techniques have models in which the solute degrees of freedom are treated explicitly but the solvent degrees of freedom are not. This requires that the energy surface used for the protein degrees of freedom be a potential of mean force (PMF) in which the solvent degrees of freedom are implicitly averaged over [68]. Assuming that the full potential energy function consists of a term, U_{vac} for the interactions within the protein, depending only on the protein degrees of freedom, $\mathbf{r}$, and an additional term for the protein-solvent and solvent-solvent interactions, the PMF is ideally equation 2.6.1, where $DeltaG_{\text{sol}}(\mathbf{r})$ is the free energy of transferring the protein from vacuum to the solvent with its internal degrees of freedom fixed at $\mathbf{r}$.

$$U_{\text{PMF}}(\mathbf{r}) = U_{\text{vac}}(\mathbf{r}) + \Delta G_{\text{sol}}(\mathbf{r}) \tag{2.6.1}$$

Some of the main models are:

COSMO Model [69] treats the solvent as a high dielectric continuum, interacting with charges that are embedded in solute molecules of lower dielectric. In spite of the severity of the approximation, this model often gives a good account of equilibrium solvation energetics.

Poisson-Boltzmann Model [70] describes electrostatic interactions in a multiple-dielectric environment and are typically solved by finite-difference or boundary element numerical methods. These can be efficiently solved for small molecules but may become expensive for proteins or nucleic acids. Although progress continues to be made in numerical solutions, there is a clear interest in exploring more efficient, if approximate, approaches to this problem.

Generalized Born Model [71, 72] computes the electrostatic work required to move a charged sphere from a vacuum environment into a continuous dielectric region. The result is proportional to the square of the charge and is inversely proportional to the size of the ion. The basis of generalized Born theory is to extend these ideas to non-spherical molecules by casting the electrostatic contribution to solvation. The Born model have not traditionally considered salt effects, but the model has be extended to low-salt concentrations [73]. The key to making Generalized Born calculations more accurate (in the sense of agreement with Poisson-Boltzmann calculations) is improved estimation of the effective Born radii [74].

2.7 References

[1] Philip M. Morse. Diatomic Molecules According to the Wave Mechanics. II. Vibrational Levels. *Phys. Rev.*, 34:57–64, Jul 1929.

[2] Gustav Mie. Zur kinetischen Theorie der einatomigen Körper. *Annalen der Physik*, 316(8):657–697, 1903.

[3] A. Warshel and S. Lifson. Consistent force field calculations. II. Crystal structures, sublimation energies, molecular and lattice vibrations, molecular conformations, and enthalpies of alkanes. *The Journal of Chemical Physics*, 53:582, 1970.

[4] David N.J. White. A computationally efficient alternative to the Buckingham potential for molecular mechanics calculations. *Journal of Computer-Aided Molecular Design*, 11:517–521, 1997. 10.1023/A:1007911511862.

[5] J. E. Jones. On the Determination of Molecular Fields. I. From the Variation of the Viscosity of a Gas with Temperature. *Proceedings of the Royal Society of London. Series A*, 106(738):441–462, 1924.

[6] J.-P. Hansen and L. Verlet. Phase Transitions of the Lennard-Jones System. *Physical Review*, 184:151–161, 1969.

[7] H. A. Lorentz. Ueber die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase. *Annalen der Physik*, 248(1):127–136, 1881.

[8] Robert J. Good and Christopher J. Hope. New Combining Rule for Intermolecular Distances in Intermolecular Potential Functions. *The Journal of Chemical Physics*, 53(2):540–543, 1970.

[9] B. E. F. Fender and Jr. G. D. Halsey. Second Virial Coefficients of Argon, Krypton, and Argon-Krypton Mixtures at Low Temperatures. *The Journal of Chemical Physics*, 36(7):1881–1888, 1962.

[10] William L. Jorgensen, David S. Maxwell, and Julian Tirado-Rives. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *Journal of the American Chemical Society*, 118(45):11225–11236, 1996.

[11] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis,

J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry*, 30(10):1545–1614, 2009.

[12] Wendy D. Cornell, Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz, David M. Ferguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell, and Peter A. Kollman. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *Journal of the American Chemical Society*, 117(19):5179–5197, 1995.

[13] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American Chemical Society*, 114(25):10024–10035, 1992.

[14] William L. Jorgensen and Julian. Tirado-Rives. The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin. *Journal of the American Chemical Society*, 110(6):1657–1666, 1988.

[15] Markus Christen, Philippe H. Hunenberger, Dirk Bakowies, Riccardo Baron, Roland Burgi, Daan P. Geerke, Tim N. Heinz, Mika A. Kastenholz, Vincent Krautler, Chris Oostenbrink, Christine Peter, Daniel Trzesniak, and Wilfred F. van Gunsteren. The GROMOS software for biomolecular simulation: GROMOS05. *Journal of Computational Chemistry*, 26(16):1719–1751, 2005.

[16] M.G. Martin and J.I. Siepmann. Transferable potentials for phase equilibria. 1. United-atom description of n-alkanes. *The Journal of Physical Chemistry B*, 102(14):2569–2577, 1998.

[17] Marcus G. and Martin. Comparison of the AMBER, CHARMM, COMPASS, GROMOS, OPLS, TraPPE and UFF force fields for prediction of vapor–liquid coexistence curves and liquid densities. *Fluid Phase Equilibria*, 248(1):50 – 55, 2006.

[18] Yong Duan, Chun Wu, Shibasish Chowdhury, Mathew C. Lee, Guoming Xiong, Wei Zhang, Rong Yang, Piotr Cieplak, Ray Luo, Taisung Lee, James Caldwell, Junmei Wang, and Peter Kollman. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *Journal of Computational Chemistry*, 24(16):1999–2012, 2003.

[19] Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. Development and testing of a general amber force field. *Journal of Computational Chemistry*, 25(9):1157–1174, 2004.

[20] A. D. MacKerell, D. Bashford, Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins†. *The Journal of Physical Chemistry B*, 102(18):3586–3616, 1998.

[21] A.D. Mackerell Jr, M. Feig, and C.L. Brooks III. Extending the treatment of backbone energetics in protein force fields: Limitations of gas-phase quantum mechanics

in reproducing protein conformational distributions in molecular dynamics simulations. *Journal of computational chemistry*, 25(11):1400–1415, 2004.

[22] A.D. MacKerell Jr, N. Banavali, and N. Foloppe. Development and current status of the CHARMM force field for nucleic acids. *Biopolymers*, 56(4):257–265, 2000.

[23] K. Vanommeslaeghe, E. Hatcher, C. Acharya, S. Kundu, S. Zhong, J. Shim, E. Darian, O. Guvench, P. Lopes, I. Vorobyov, et al. CHARMM general force field: A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *Journal of computational chemistry*, 31(4):671–690, 2010.

[24] III WH Reiher. *Theoretical studies of hydrogen bonding*. PhD thesis, Harvard University, 1985.

[25] S.V. Sambasivarao and O. Acevedo. Development of OPLS-AA force field parameters for 68 unique ionic liquids. *Journal of Chemical Theory and Computation*, 5(4):1038–1050, 2009.

[26] Z. Xu, H.H. Luo, and D.P. Tieleman. Modifying the OPLS-AA force field to improve hydration free energies for several amino acid side chains using new atomic charges and an off-plane charge model for aromatic residues. *Journal of computational chemistry*, 28(3):689–697, 2007.

[27] Berk Hess, Carsten Kutzner, David van der Spoel, and Erik Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*, 4(3):435–447, 2008.

[28] Wilfred F. van Gunsteren, S. R. Billeter, A. A. Eising, Philippe H. Hünenberger, P. Krüger, Alan E. Mark, W. R. P. Scott, and Ilario G. Tironi. *Biomolecular Simulation: The GROMOS96 manual and user guide*. 1996.

[29] Markus Christen, Philippe H. Hünenberger, Dirk Bakowies, Riccardo Baron, Roland Bürgi, Daan P. Geerke, Tim N. Heinz, Mika A. Kastenholz, Vincent Krautler, Chris Oostenbrink, Christine Peter, Daniel Trzesniak, and Wilfred F. van Gunsteren. The GROMOS software for biomolecular simulation: GROMOS05. *Journal of Computational Chemistry*, 26(16):1719–1751, 2005.

[30] Berend Smit, Sami Karaborni, and J. Ilja Siepmann. Computer simulations of vapor–liquid phase equilibria of n-alkanes. *The Journal of Chemical Physics*, 102(5):2126–2140, 1995.

[31] M.G. Martin and J.I. Siepmann. Novel configurational-bias Monte Carlo method for branched molecules. Transferable potentials for phase equilibria. 2. United-atom description of branched alkanes. *The Journal of Physical Chemistry B*, 103(21):4508–4517, 1999.

[32] C.D. Wick, M.G. Martin, and J.I. Siepmann. Transferable potentials for phase equilibria. 4. United-atom description of linear and branched alkenes and alkylbenzenes. *The Journal of Physical Chemistry B*, 104(33):8008–8016, 2000.

[33] B. Chen, J.J. Potoff, and J.I. Siepmann. Monte Carlo calculations for alcohols and their mixtures with alkanes. Transferable potentials for phase equilibria. 5. United-atom description of primary, secondary, and tertiary alcohols. *The Journal of Physical Chemistry B*, 105(15):3093–3104, 2001.

[34] J.M. Stubbs, J.J. Potoff, and J.I. Siepmann. Transferable potentials for phase equilibria. 6. United-atom description for ethers, glycols, ketones, and aldehydes. *The Journal of Physical Chemistry B*, 108(45):17596–17605, 2004.

[35] C.D. Wick, J.M. Stubbs, N. Rai, and J.I. Siepmann. Transferable potentials for phase equilibria. 7. Primary, secondary, and tertiary amines, nitroalkanes and nitrobenzene, nitriles, amides, pyridine, and pyrimidine. *The Journal of Physical Chemistry B*, 109(40):18974–18982, 2005.

[36] N. Lubna, G. Kamath, J.J. Potoff, N. Rai, and J.I. Siepmann. Transferable potentials for phase equilibria. 8. United-atom description for thiols, sulfides, disulfides, and thiophene. *The Journal of Physical Chemistry B*, 109(50):24100–24107, 2005.

[37] B. Chen and J.I. Siepmann. Transferable potentials for phase equilibria. 3. Explicit-hydrogen description of normal alkanes. *The Journal of Physical Chemistry B*, 103(25):5370–5379, 1999.

[38] N. Rai and J.I. Siepmann. Transferable potentials for phase equilibria. 9. Explicit hydrogen description of benzene and five-membered and six-membered heterocyclic aromatic compounds. *The Journal of Physical Chemistry B*, 111(36):10790–10799, 2007.

[39] J. D. Bernal and R. H. Fowler. A Theory of Water and Ionic Solution, with Particular Reference to Hydrogen and Hydroxyl Ions. *Journal of Chemical Physics*, 1:515–548, 1933.

[40] F. H. Stillinger and A. Rahman. Improved Simulation of Liquid Water by Molecular Dynamics. *Journal of Chemical Physics*, 60:1545–1557, 1974.

[41] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, and J. Hermans. *Intermolecular Forces*. Reidel, Dordrecht, 1981.

[42] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. Comparison of simple potential functions for simulating liquid water. *Journal of Chemical Physics*, 79:926–935, 1983.

[43] J. L. F. Abascal and C. Vega. A general purpose model for the condensed phases of water: TIP4P/2005. *Journal of Chemical Physics*, 123:234505, 2005.

[44] P. E. Mason and J. W. Brady. "Tetrahedrality" and the relationship between collective structure and radial distribution functions in liquid water. *Journal of Physical Chemistry B*, 111:5669–5679, 2007.

[45] M-L. Tan, J. T. Fischer, A. Chandra, B. R. Brooks, and T. Ichiye. A temperature of maximum density in soft sticky dipole water. *Chemical Physics Letters*, 376:646–652, 2003.

[46] K. Kiyohara, K. E. Gubbins, and A. Z. Panagiotopoulos. Phase coexistence properties of polarizable water models. *Molecular Physics*, 94:803–808, 1998.

[47] D. van der Spoel, P. J. van Maaren, and H. J. C. Berendsen. A systematic study of water models for molecular simulation: Derivation of water models optimized for use with a reaction field. *Journal of Chemical Physics*, 108:10220–10230, 1998.

[48] M. W. Mahoney and W. L. Jorgensen. Diffusion constant of the TIP5P model of liquid water. *Journal of Chemical Physics*, 114:363–366, 2001.

[49] C. Vega and J. L. F. Abascal. Relation between the melting temperature and the temperature of maximum density for the most common models of water. *Journal of Chemical Physics*, 123:144504, 2005.

[50] H. Yu and W. F. van Gunsteren. Charge-on-spring polarizable water models revisited: from water clusters to liquid water to ice. *Journal of Chemical Physics*, 121:9549–9564, 2004.

[51] H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma. The missing term in effective pair potentials. *Journal of Physical Chemistry*, 91:6269–6271, 1987.

[52] L. A. Baez and P. Clancy. Existence of a density maximum in extended simple point-charge water. *Journal of Chemical Physics*, 101:9837–9840, 1994.

[53] Y. Wu, H. L. Tepper, and G. A. Voth. Flexible simple point-charge water model with improved liquid state properties. *Journal of Chemical Physics*, 124 :024503, 2006.

[54] I. M. Svishchev, P. G. Kusalik, J. Wang, and R. J. Boyd. Polarizable point-charge model for water. Results under normal and extreme conditions. *Journal of Chemical Physics*, 105:4742–4750, 1996.

[55] M. W. Mahoney and W. L. Jorgensen. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *Journal of Chemical Physics*, 112:8910–8922, 2000.

[56] H. W. Horn, W. C. Swope, J. W. Pitera, J. D. Madura, T. J. Dick, G. L. Hura, and T. Head-Gordon. Development of an improved four-site water model for biomolecular simulations. *Journal of Chemical Physics*, 120:9665–9678, 2004.

[57] S. W. Rick. Simulation of ice and liquid water over a range of temperatures using the fluctuating charge model. *Journal of Chemical Physics*, 114:2276–2283, 2001.

[58] P. J. van Maaren and D. van der Spoel. Molecular dynamics of water with novel shell-model potentials. *Journal of Physical Chemistry B*, 105:2618–2626, 2001.

[59] M. A. González and J. L. F. Abascal. A flexible model for water based on TIP4P/2005. *Journal of Chemical Physics*, 135:224516, 2011.

[60] A.-P. E. Kunz and W. F. van Gunsteren. Development of a nonlinear classical pPolarization model for liquid water and aqueous solutions: COS/D. *Journal of Physical Chemistry A*, 113:11570–11579, 2009.

[61] P. Paricaud, M. Predota, A. A. Chialvo, and P. T. Cummings. From dimer and condensed phases at extreme conditions: Accurate predictions of the properties of water by a Gaussian charge polarizable model. *Journal of Chemical Physics*, 122:244511, 2005.

[62] G. Lamoureux, E. Harder, I. V. Vorobyov, B. Roux, and A. D. MacKerell Jr. A polarizable model of water for molecular dynamics simulations of biomolecules. *Chemical Physics Letters*, 418:241–245, 2005.

[63] S. W. Rick. A reoptimization of the five-site water potential (TIP5P) for use with Ewald sums. *Journal of Chemcal Physics*, 120:6085–6093, 2004.

[64] G. S. Fanourgakis and S. S. Xantheas. The flexible, polarizable, Thole-type interaction potential for water (TTM2-F) revisited. *Journal of Physical Chemistry A*, 110:4100–4106, 2006.

[65] H. A. Stern, F. Rittner, B. J. Berne, and R. A. Friesner. Combined fluctuating charge and polarizable dipole models: Application to a five-site water potential function. *Journal of Chemical Physics*, 115:2237–2251, 2001.

[66] H. Nada and J. P. J. M. van der Eerden. An intermolecular potential model for the simulation of ice and water near the melting point: A six-site model of H_2O. *Journal of Chemical Physics*, 118:7401–7413, 2003.

[67] S. Y. Liem, P. L. A. Popelier, and M. Leslie. Simulation of liquid water using a high-rank quantum topological electrostatic potential. *International Journal of Quantum Chemistry*, 99:685–694, 2004.

[68] B Roux and T. Simonson. *Biophysical Chemistry*, 78:1–20, 1999.

[69] A. Klamt and G. Schüürmann. *J Chem. Soc. Perkin Trans*, 2:799–805, 1993.

[70] C. J. Cramer and D. G. Truhlar. *Chemical Reviews*, 99:2161–2200, 1999.

[71] M. Born. *Z. Phys.*, 1:45–48, 1920.

[72] D. Bashford and D. Case. *Annu. Rev. Phys. Chem.*, 51:129–152, 2000.

[73] J. Srinivasan, M. W. Trevathan, P. Beroza, and D. A. Case. *Theor. Chem. Ace.*, 101:126–434, 1999.

[74] M. S. Lee, F. R. Salsbury Jr, and C. L. Brooks III. *Journal of Physical Chemistry*, 116,:10606–10614, 2002.

Table 2.1: Calculated physical properties of the water models at 25 °C and 1 atm.

Model	Type	Dipole Moment	Dielectric constant	Self Diffusion / 10^{-5}cm^2 s^{-1}	Average Configurational Energy / kJ mol^{-1}	Density Maximum Temperature /°C	Expansion Coefficient, 10^{-4} °C^{-1}
SSD [45]	-[1]	2.35 [45]	72 [45]	2.13 [45]	-40.2 [45]	-13 [45]	-
SPC [41]	a	2.27 [46]	65 [47]	3.85 [48]	-41.0 [47]	-45 [49]	7.3[2][50]
SPC/E [51]	a	2.35 [51]	71 [51]	2.49 [48]	-41.5 [51]	-38 [52]	5.14 [53]
SPC/Fw[4][53]	a	2.39 [53]	79.63 [53]	2.32 [53]	-	-	4.98 [53]
PPC[4][51]	b	2.52 [51]	77 [51]	2.6 [51]	-43.2 [51]	4 [54]	-
TIP3P [42]	a	2.35 [55]	82 [51]	5.19 [48]	-41.1 [55]	-91 [49]	9.2 [55]
TIP3P/Fw[4][53]	a	2.57 [53]	193 [53]	3.53 [53]	-	-	7.81 [53]
TIP4P [55]	c	2.18 [51, 55]	53 [51]	3.29 [48]	-41.8 [55]	-25 [55]	4.4 [55]
TIP4P-Ew [56]	c	2.32 [56]	62.9 [56]	2.4 [56]	-46.5 [56]	1 [56]	3.1 [56]
TIP4P-FQ [57]	c	2.64 [57]	79 [57]	1.93 [57]	-41.4 [58]	7 [57]	-
TIP4P/2005 [43]	c	2.305 [43]	60 [43]	2.08 [43]	-	5 [43]	2.8 [43]
TIP4P/2005f [59]	c	2.319 [59]	55.3 [59]	1.93 [59]	-	7 [59]	-
SWFLEX-AI[4][58]	c	2.69 [58]	116 [58]	3.66 [58]	-41.7 [58]	-	-
COS/G3 [50]	c	2.57[2][50]	88[2][50]	2.6[2][50]	-41.1[2][50]	-	7.0[2][50]
COS/D [60]	c	2.43 [60]	69.8 [60]	2.5 [60]	-41.8 [60]	-	-
GCPM[4][61]	c	2.723 [61]	84.3 [61]	2.26 [61]	-44.8 [61]	-13 [61]	-
SWM4-NDP[4][62]	c	2.461 [62]	79 [62]	2.33 [62]	-41.5 [62]	-	-
TIP5P [55]	d	2.29 [55]	81.5 [55]	2.62 [48]	-41.3 [55]	4 [55]	6.3 [55]
TIP5P-Ew [63]	d	2.29 [63]	92 [63]	2.8 [63]	-	8 [63]	4.9 [63]
TTM2-F [64]	c	2.67 [64]	67.2 [64]	1.4 [64]	-45.1 [64]	-	-
POL5/TZ[4][65]	d	2.712 [65]	98 [65]	1.81 [65]	-41.5 [65]	25 [65]	-
Six-site [66]	c+d	1.89[3][66]	33[3][66]	-	-	14[3][66]	2.4[3][66]
QCT [67]	a	1.85[2][67]	-	1.5[2][67]	-42.7[2][67]	10[2][67]	3.5[2][67]
Experimental	-	2.95	78.4	2.30	-41.5 [55]	3.984	2.53

[1]Has only a single, center of mass, interaction site with a tetrahedrally coordinated sticky potential that regulates the tetrahedral coordination of neighboring molecules

[2]Calculated at 27 °C

[3]Calculated at 20 °C

[4]Polarizable model

Chapter 3

Practical Aspects of Molecular Dynamics

Contents

3.1 Introduction

The best way to examine the practical aspects of Molecular Dynamics is to consider a simple program. This program is kept as simple as possible to allow illustration of the key features of a Molecular Dynamics simulaion, algorithm 3.1.

The key steps in the program are:

1. Read in the parameters that specify the conditions of the simulation, e.g. initial temperature, number of particles, density, and time step.
2. Initialize the system, i.e. select the initial position and velocities.
3. Conpute the forces on all particles.
4. Integrate Newton's equations of motion.
5. Steps 3 and 4 are the core part of the simulation and are repeared until the time of the simulation has reached the desired time.
6. After completion, we compute and print the averages of the measured quantities we want.

3.2 Reduced Units

In simulations it is often convenient to express quantities such as the temperature, density, or pressure in reduced units. This means that we choose a convenient unit of energy, length, and mass to express all these quantities in. If using the Lennard-Jones potential (equation 2.4.4) it is typical to use σ as the unit of lenght, ϵ as the unit of energy, and m (the mass of the atoms in the system) as the unit of mass. This means that,

$$t^* = \frac{t}{\sigma\sqrt{m/\epsilon}} \tag{3.2.1}$$

$$r^* = \frac{r}{\sigma} \tag{3.2.2}$$

$$u^* = \frac{u}{\epsilon} \tag{3.2.3}$$

$$T^* = \frac{k_B T}{\epsilon} \tag{3.2.4}$$

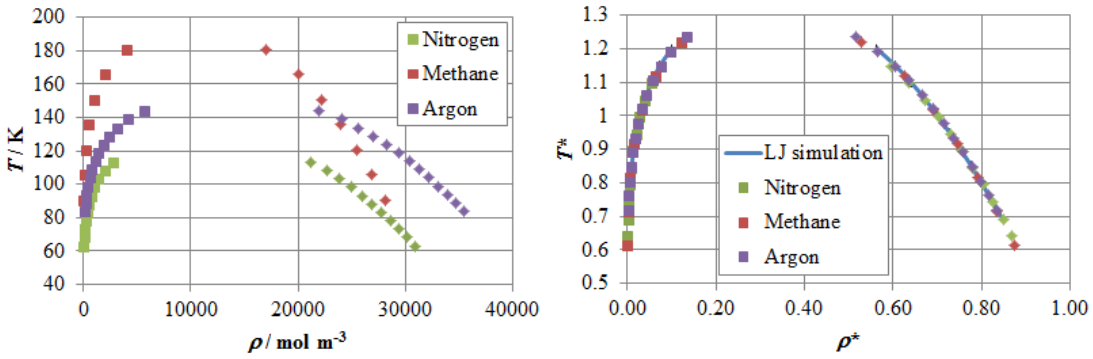
$$P^* = \frac{P\sigma^3}{\epsilon} \tag{3.2.5}$$

$$\rho^* = \rho\sigma^3 \tag{3.2.6}$$

The most important reason for this is that there are many combinations of ρ, T, ϵ, and σ that correspond to the same state in reduced units, Figure 3.1 provides an example for some components. Another practical reason is that if we were working with SI units the absolute values would be very small meaning that computational errors could appear in the simulations.

```
% Simple MD program

% Read in initial parameters
MDparam;

% Initialize the system
[coords,L,velSTD,vels] = init(Temp,mass,density,nPart,nDim);

% Set initial time
time = 0;

for step = 1:nSteps

    % Calculate the forces
    [forces,virial,en] = calcforce(coords,L);

    % Intergrate the equations of motion
    [coords,vels] = integrate(coords,forces,vels,mass,L,dt,nDim,velSTD,nu);

    % Step time forward
    time = time + dt;

    % Sample measured qualities
    samples(step,nSteps,sampleFreq,vels,coords,L);

end
```

Algorithm 3.1: A simple outline Molecular Dynamics program



(a) (b)

Figure 3.1: Variation of the vapour-liquid equlibria for some example components, squares are saturated vapour and diamonds are saturated liquid. (a) The plots in S.I. units and (b) the plots in reduced units. LJ parameters: Nitrogen $\epsilon/k = 99\,\mathrm{K}$ and $\sigma = 360\,\mathrm{pm}$, Methane $\epsilon/k = 148\,\mathrm{K}$ and $\sigma = 373\,\mathrm{pm}$, Argon $\epsilon/k = 117\,\mathrm{K}$ and $\sigma = 339\,\mathrm{pm}$.

3.3 Initialization

To start the simulation we need to assign initial positions and velocities to all particles in the system. The particles should be positioned so that the structure is compatible with the structure we are aiming to simulate, and should be picked so that the positions don't result in an overlap of the molecular cores. Often the most convenient option for this is to place the particles on a simple cubic lattice, as in Algorithm 3.2. Assuming that the values of the density and temperature are chosen such that the simple cubic lattice is unstable and melts rapidly.

After placing each particle on its lattice site we need to attribute a velocity to each particle. To allow each particle to have a velocity which fits to the expected ditribution of velocities we can attribute each particle a velocity based on a Gaussian distribution with a mean

```
function [coords, L] = initCubicGrid(nPart,density)
% Set initial configuration of particles on a cubic grid.
%
% Parameters:     nPart    Number of particles to assign on the grid
%                 density  The density of the particles
%
% Return values:  coords   An array of particles' coordinates
%                 L        Box length along each side

% Initialize with zeroes
coords = zeros(3,nPart);

% Get the cooresponding box size
L = (nPart/density)^(1.0/3);

% Find the lowest perfect cube greater than or equal to the number of
% particles
nCube = 2;

while (nCube^3 < nPart)
    nCube = nCube + 1;
end

% Start positioning - use a 3D index for counting the spots
index = [0,0,0]';

% Assign particle positions
for part=1:nPart
    % Set coordinate
    coords(:,part) = (index+[0.5,0.5,0.5]')*(L/nCube);

    % Advance the index
    index(1) = index(1) + 1;
    if (index(1) == nCube)
        index(1) = 0;
        index(2) = index(2) + 1;
        if (index(2) == nCube)
            index(2) = 0;
            index(3) = index(3) + 1;
        end
    end
end

end
```

Algorithm 3.2: Code to produce a set of cooridinates based on a simple cubic lattice

value of 0 and a standard deviation of $\sqrt{(T/m)}$[1], as in Algorithm 3.3. This process should make the total momentum zero; however, as the distribution is random, the resulting velocities should be adjusted so that the total momentum is zero,

$$\mathbf{v}_i = \mathbf{v}_i - \frac{\sum_{i=1}^{N} \mathbf{v}_i}{N} \tag{3.3.1}$$

The velocities also need to be adjusted so that the mean kinetic energy is the desired value for the temperature of the simulation equation 1.2.7. We can adjust the initial temperature to match the desired temperature, T, by scaling all velocities with the factor $\sqrt{T/T_{init}}$,

$$\mathbf{v}_i = \mathbf{v}_i \sqrt{\frac{T}{T_{init}}} \tag{3.3.2}$$

[1] as at thermal equlibrium $\langle v_i^2 \rangle = k_B T/m$

This initial setting of the temperature is not particularly critical, as the temperature will change anyway during the equilibration; however, this speeds up the equilibration process.

? Activity

Create the function for init to produce the inital coordinates and velocities for the MD program. Algorithms 3.2 and 3.3 maybe useful for this.

Check your code by calculating the mean and standard deviation of the velocities. What does the distrubution of velocities look like?

3.4 Force Calculation

The force calculation is the most time-consuming part of almost all Molacular Dynamics simulations. This is because not only do we need to perform this calculation every step, but we also need to consider all the pairwise additive interactions, i.e. the contribution to the force on particle i due to all its neighbors.

If we consider only the interacton between a particle and the nearest image of another particle, this implies that, for a system of N particles, we would need to evaluate $N\left(N-1\right)/2$ pair distances. This imples that the time needed for evaluation of the forces scales as N^2 [1].

We first must compute the current distance between each pair of particles i and j in the x, y, and z directions. Due to the limited size of the simulation we need to check this distance with the periodic boundary conditions, see Section 3.5. Typically we use a cutoff distance, r_c where outside this distance we have no interaction between particles. The values of r_c is chosen to be less than half the diameter of the periodic box, which means that we limit the interactions between i and j to the interaction between i and the nearest periodic image of j.

Having computed all the cartesian components of $\mathbf{r}_{ij}$ we compute r_{ij}^2. We can test if this is less than r_c^2; if not we can skip to the next value of j[2]. If the given pair of particles is close enough to interact, we compute the force between them. The x-component of the force is,

$$f_x\left(r\right) = -\frac{\partial\, u\left(r\right)}{\partial\, x} = -\left(\frac{x}{r}\right)\left(\frac{\partial\, u\left(r\right)}{\partial\, r}\right) \tag{3.4.1}$$

? Activity

Create the function calcforce based on the interaction between the particles being given by the Lennard-Jones potential, equation 2.4.4. Remember we are using reduced units. Algorithm 3.4 may be useful.

Check that this gives the correct shape of the force.

[1] There exist effcient techniques to speed up this evaluation, to scale with N; however, these are beyond the scope of this course

[2] Note that we don't compute $|\mathbf{r}_{ij}|$ because it is computationally expensive due to the calculation of a square root

```
function randNums = randGauss(mu,sigma,nDim)
% Generate random numbers from a Gaussian distribution.
%
% This function uses the Matlab function randn() to produce numbers
% distributed normallly with mean 0 and std 1. It then shifts the numbers
% to match the given mean and standard deviation.
%
% Parameters:     mu        The distribution's mean value
%                 sigma     The distribution's standard deviation
%                 nDim      Number of random numbers to generate and size
%                           of returned array
%
% Return values:  randNums  An array of size 1,nDim of normally
%                           distributed numbers with mean 'mu' and
%                           standard deviation 'sigma'

% Generate normally distributed random numbers
randNums = randn(nDim,1);

% Shift to match given mean and std
randNums = mu + randNums * sigma;

end
```

Algorithm 3.3: Code to produce a set of scaled velocities based on a Gaussian distribution.

3.5 Simulation Box

System sizes accessible to atomistic simulations are of the order of thousands of atoms. These can be typically enclosed in a cubic box with 15 nm sides. The molecules on the surface of the box will experience different forces to the ones in the centre. Unless we want to simulate a cluster of molecules then this situation is not realistic. No matter how large the simulated system, the number of molecules N would be negligible compared with the number of atoms contained in a macroscopic system (of the order of 10^{23}).

This is typically solved by imposing periodic boundary condition (PBC), i.e. surrounding the simulation box with 'pseudo' simulation boxes, being a clone of the original one. If one of the particles is located at position $\mathbf{r}$ in the box, we assume that this particle really represents an infinite set of particles located at,

$$\mathbf{r} + l\mathbf{a} + m\mathbf{b} + n\mathbf{c}, \qquad \{\forall\, l, m, n \in \mathbb{Z}\} \tag{3.5.1}$$

where $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are the vectors corresponding to the edges of the box. All the image molecules move together, and infact only one is actually represented in the program. This results in a simulation with bulk properties across the whole original box. When the molecule leaves the box on one side, as shown on the Figure 3.2, it will enter the neighbouring 'pseudo' box, and by such re-enter the initial box from the opposite side.

To be able to perform calculations at the reasonable speed, the interaction of the molecule is only calculated with it's direct neighbours. The standard way is to use a Verlet list [1] that, for every particle, defines a surrounding shell at a distance slightly larger than the cut-off radius for the interactions. The cut-off distance should be less or equal to the half of the box size.

This means when we calculate the distance between particles we need to check to see if the image of the particle over the periodic boundary condition is closer, Algorithm 3.4.

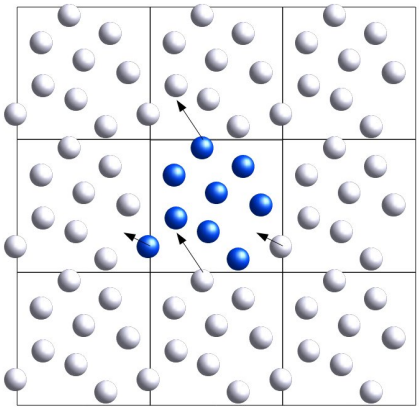


Figure 3.2: Periodic boundary condition, two-dimensional illustration: the box containing blue atoms is the simulation box, while the boxes containing white atoms are 'pseudo' boxes, that mimic the behaviour of the main simulation box.

```
function vec = distPBC3D(vec,L)
% Correct a distance vector accoring to periodic boundary conditions.
%
% This function is used for applying PBC (Periodic Boundary Conditions)
% on a distacne vector in 3D.
%
% A distance vector is a vector that is a result of subtracting the
% positions of two particles.
%
% Parameters:      vec    A distance vector (3D array)
%                  L      The box size along all directions
%
% Return values:   vec    The corrected distance vector

% Calculate the half box size in each direction
hL = L/2.0;

% Distance vector should be in the range -hLx -> hLx and -hLy -> hLy
% Therefore, we need to apply the following changes if it's not in
% this range:

for dim=1:3
    if (vec(dim) > hL)
        vec(dim) = vec(dim)-L;
    elseif (vec(dim) < -hL)
        vec(dim) = vec(dim)+L;
    end
end

end
```

Algorithm 3.4: Code to check distance between particles with cubic periodic bondary conditions.

3.6 Integration Algorithms

In a system with a large number of particles the coupled differential equations 1.2.2 and 1.2.3 cannot be solved analytically, therefore must be solved numerically. Time integration algorithms are based on finite difference methods, where time is discretised on a finite grid, the time step δt being the distance between consecutive points on the grid. Knowing the positions and some of their time derivatives at time t (the exact details depend on the type of algorithm), the integration scheme gives the same quantities at a later time $t + \delta t$. By iterating the procedure, the time evolution of the system can be followed for long times.

Several different algorithms have been developed to integrate the equation of motion, but as these are all approximate there are errors associated with them. These two types of error are: truncation errors, related to the accuracy of the finite difference method with respect to the true solution; and round-off errors, related to errors associated to a particular implementation of the algorithm. Both errors can be reduced by decreasing δt.

3.6.1 Verlet algorithm

This algorithm was developed by Loup Verlet in 1967 [2] and became the base for most algorithms used in modern simulations. The algorithm calculates positions forward and backward in time allowing a better calculation of the new positions than a simple single step numerical integration. This is performed via a third order Taylor expansion, equation 3.6.1.

$$\begin{aligned}\mathbf{r}\left(t+\delta t\right) &= \mathbf{r}\left(t\right) + \mathbf{v}\left(t\right)\delta t + \frac{1}{2}\mathbf{a}\left(t\right)\delta t^2 + \frac{1}{6}\dddot{\mathbf{r}}\left(t\right)\delta t^3 + \mathcal{O}\left(\delta t^4\right) \\ \mathbf{r}\left(t-\delta t\right) &= \mathbf{r}\left(t\right) - \mathbf{v}\left(t\right)\delta t + \frac{1}{2}\mathbf{a}\left(t\right)\delta t^2 - \frac{1}{6}\dddot{\mathbf{r}}\left(t\right)\delta t^3 + \mathcal{O}\left(\delta t^4\right)\end{aligned} \tag{3.6.1}$$

Adding two above expressions together yields, equation 3.6.2.

$$\mathbf{r}\left(t+\delta t\right) = 2\mathbf{r}\left(t\right) - \mathbf{r}\left(t-\delta t\right) + \mathbf{a}\left(t\right)\delta t^2 + \mathcal{O}\left(\delta t^4\right) \tag{3.6.2}$$

The main problem with this algorithm is that the velocities are not directly generated, though it makes it computationally cheaper as lower data storage is required. The Verlet algorithm can be seen graphical as Figure 3.3(a).

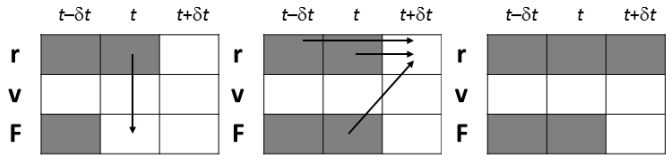
3.6.2 The Leap-Frog Algorithm

The leap-frog algorithm is a modification of the classical Verlet algorithm. It calculates velocities, $\mathbf{v}$, at a half time $\delta t/2$ step in advance of the reference time, t, equation 3.6.3.

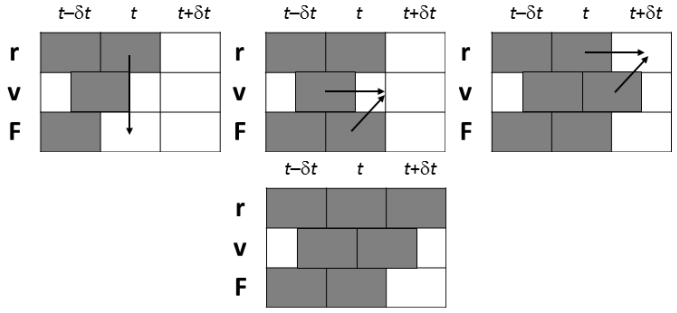
$$\mathbf{v}\left(t+\frac{\delta t}{2}\right) = \mathbf{v}\left(t-\frac{\delta t}{2}\right) + \mathbf{a}\left(t\right)\delta t \tag{3.6.3}$$

These are then used to calculate the position at the full time step, δt, equation 3.6.4.

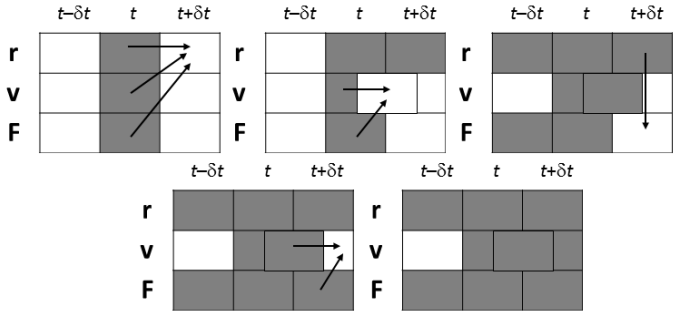
$$\mathbf{r}\left(t+\delta t\right) = \mathbf{r}\left(t\right) - \mathbf{v}\left(t+\frac{\delta t}{2}\right)\delta t \tag{3.6.4}$$



(a) Verlet



(b) Leap-Frog



(c) Velocity Verlet

Figure 3.3: Graphical representation of the key integration algorithms.

In this method the velocities leap over positions, which in turn leap over velocities. The advantage of this method is that velocities are calculated accurately. However, they are not calculated at the same time as positions. So if velocities at a given time, t, are required they must be averaged as equation 3.6.5.

$$\mathbf{v}(t) = \frac{1}{2}\left[\mathbf{v}\left(t - \frac{\delta t}{2}\right) + \mathbf{v}\left(t + \frac{\delta t}{2}\right)\right] \tag{3.6.5}$$

This algorithm is easily applied to complex molecules and may also be combined with constraint algorithms. The speed and simplicity of this algorithm means it often makes it the first choice for molecular simulation. The leap-frog algorithm can be seen graphical as Figure 3.3(b).

3.6.3 The velocity Verlet algorithm

This is another modification of the Verlet algorithm, similar to the leap-frog algorithm, but allowing the calculation of positions, velocities and accelerations at the given time

[3],

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \mathbf{v}(t)\,\delta t + \frac{1}{2}\mathbf{a}(t)\,\delta t^2 \tag{3.6.6}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \frac{\mathbf{a}(t) + \mathbf{a}(t+\delta t)}{2}\delta t \tag{3.6.7}$$

This results in more accurate trajectories and less drift of the total energy. The increase in precision also comes with increase in computational cost. The standard implementation scheme of this algorithm is:

1. Calculate $\mathbf{r}(t+\delta t)$ from equation 3.6.6.
2. Calculate the half step velocity, $\mathbf{v}(t+0.5\delta t) = \mathbf{v}(t) + 0.5\mathbf{a}(t)\,\delta t$.
3. Derive $\mathbf{a}(t+\delta t)$ from the interaction force using $\mathbf{r}(t+\delta t)$.
4. Calculate the final velocity, $\mathbf{v}(t+\delta t) = \mathbf{v}(t+0.5\delta t) + 0.5\mathbf{a}(t+\delta t)\,\delta t$

The velocity verlet algorithm can be seen graphical as Figure 3.3(c).

3.6.4 Beeman's algorithm

Beeman's algorithm is also related to Verlet algorithm. Beeman's algorithm [4] uses more accurate expressions, that allows for more accurate velocities and provides better energy conservation. This method is computationally more expensive and is only used when high accuracy is required.

? Activity

Create the function integrate based on the velocity verlet algorithm. Your Molecular Dynamics simulation can now be ran. Try the parameters below:

```
% Set configuration parameters
nPart = 125;          % Number of particles
density = 0.85;       % Density of particles
mass = 1;             % Particles' mass
nDim = 3;             % The dimensionality of the system

% Set simulation parameters
dt = 0.0005;          % Integration time
Temp = 2.0;           % Simulation temperature

nSteps = 1000;        % Total simulation time (in integration steps)
```

Monitor the temperature and total energy to check your simulation; Algorithms 4.2 to 4.4 will be useful. How do these vary with time?

3.7 Temperature Coupling

As we have presented it so far, the Molecular Dynamics simulation is performed in the microcanonical ensemble of constant NVE variables (and technically also: total momentum p). We can calculate the temperature using thermodynamic averages, equation 1.2.7. It is often desirable, however, to specify the temperature *a priori* and perform a simulation

in the canonical ensemble, NVT. There are several approaches to doing this and we must consider whether the methods:

- preserve the correct thermodynamics, i.e. the correct microstate distribution in the canonical ensemble.
- preserve realistic dynamics in that the equations of motion can be used to compute transport quantities accurately.

As the temperature of the system is related to the average kinetic energy, the temperature can be controlled by scaling velocities.

3.7.1 Velocity Rescaling

The idea behind this is that we rescale the velocities at each step (or after a preset number of steps) so that the kinetic energy gives the desired target temperature,

$$\mathbf{v}_{new} = \lambda \mathbf{v} \tag{3.7.1}$$

as in equation 3.3.2 where $\lambda = \sqrt{T/T\,(t)}$.

Although simple, this approach does not generate the correct thermodynamic properties of the canonical ensemble. Consider the limit in which we rescale the velocities at every time step. In this case, the kinetic energy will remain constant in time, with zero fluctuations. This is not correct, as the statistical mechanics of the canonical ensemble says that $\sigma^2_{K.E.} = 3/2Nk_B^2T^2$. In other words, velocity rescaling does not capture the correct energy fluctuations in the system.

3.7.2 Berendsen Thermostat

The Berendsen thermostat [5] is similar to the velocity rescaling approach, but assigns a time scale for the updating of the velocities, rather than assuming they are completely scaled to the target temperature at each time step. Underlying this approach is the assumption that the system is weakly coupled to a heat bath whose coupling constant, or time scale of heat transfer, is τ, equation 3.7.2.

$$\lambda^2 = 1 + \frac{\delta\, t}{\tau}\left(\frac{T}{T\,(t)} - 1\right) \tag{3.7.2}$$

Here, $\delta\, t$ is the time step in the molecular dynamics simulation. Typically, $\tau = 0.1 - 0.4\,\text{ps}$ for $\delta\, t = 1\,\text{fs}$. λ is the velocity rescaling factor as before.

The Berendsen thermostat, however, suffers from the same problems as velocity rescaling in that the energy fluctuations are not captured correctly. There are also some notable pathologies for specific systems.

3.7.3 Andersen Thermostat

The Andersen thermostat [6] introduces a stochastic element to the temperature by having random collisions of molecules with an imaginary heat bath at the desired temperature.

In the single-particle approach, a random particle is chosen and its velocity is reassigned randomly from a Maxwell-Boltzmann distribution at the desired temperature,

$$\mathscr{P}\left(\mathbf{v}_i\right) = \left(\frac{m_i}{2\pi k_B T}\right)^{1/2} \exp\left(-\frac{m_i \mathbf{v}_i^2}{2k_B T}\right) \tag{3.7.3}$$

In the Andersen scheme, one does not perform a collision during each molecular dynamics time step, but rather it is customary to adopt a collision frequency ν or collision time $\tau = 1/\nu$. The collision frequency should be chosen so as not to be too short to the time scales of molecular motions, Algorithm 3.5.

In the limit of an infinitely long trajectory averaged over many heat bath collisions, it can be shown that the Andersen thermostat rigorously generates the correct canonical ensemble probabilities. That is, the distribution of kinetic and potential energies, and the microstate probabilities of different configurations of the system, all rigorously approach their true form in the canonical ensemble.

However, the presence of random collisions causes the velocities of particles to decorrelate ("lose memory" of their initial values from some previous point in time) much faster than the NVE dynamics. As a result, true molecular kinetics are not preserved by the Andersen thermostat. For example, the computed diffusion constants for particles would give erroneous values.

3.7.4 Nose-Hoover Thermostat

This algorithm has the advantage that it can reproduce a correct canonical ensemble [7, 8]. The other thermostats above are weak coupling and are used for relaxing systems to a desired temperature, whereas this thermostat can be used for the final simulations. In this method, a thermal reservoir and friction coefficient are introduced into the equation of motion, and thereby allow kinetic energy changes in the system.

3.7.5 Practical use of Thermostats

In general, if one is interested in microscopic dynamics and transport coefficients, it is recommended to use the microcanonical ensemble, which is the only ensemble that correctly reproduces the true dynamics of the simulation. A common approach is to equilibrate a

```
function [vels]=Andersen(vels,nu,dt,velSTD,nDim)
% calculate number of particles
    nPart = size(vels,2);
% Implement the Andersen thermostat
    for part =1:nPart
        % Test for collisions with the Andersen heat bath
        if (rand < nu*dt)
            % If the particle collided with the bath, draw a new velocity
            % out of a normal distribution
            vels(:,part) = randGauss(0,velSTD,nDim);
        end
    end
end
```

Algorithm 3.5: Code to rescale the particle velocities based on the Andersen Thermostat.

system using a thermostat, but then to allow the production period to correspond to NVE dynamics.

? Activity

Add the Andersen Thermostat, Algorithm 3.5, into the integrate function and compare the temperature produced in the simulation with that without the thermostat. Use the parameters below:

```
% Set simulation parameters
nu = 1;              % Thermostat parameter - frequency of collisions
                     % with the heat bath
```

Why is there this different in temperature?

3.8 Pressure coupling

Similarly to temperature coupling, the system can be coupled to a pressure bath. There are a number of methods developed for this purpose.

3.8.1 Berendsen Barastat

This algorithm rescales the coordinates and box vectors at every step towards a given reference pressure, P_0, equation 3.8.1.

$$\frac{\mathrm{d}\,P}{\mathrm{d}\,t} = \frac{P_0 - P}{\tau_P} \tag{3.8.1}$$

Rescaling is performed by applying the scaling matrix, μ, equation 3.8.2, where β is the isothermal compressibility of the system.

$$\mu = \delta_{ij} - \frac{\Delta t}{3\tau_P}\beta_{ij}\left[P_{0ij} - P_{ij}\left(t\right)\right] \tag{3.8.2}$$

Like the Berendsen thermostat, the barastat is a weak coupling so does not generate a proper thermodynamic ensemble.

3.8.2 Parrinello-Rahman Barastat

This approach is similar to the Nose-Hoover for temperature coupling, where equation of motion for particles is updated at every step [9], and a correct thermodynamic ensemble is produced.

3.9 References

[1] Loup Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.*, 159:98–103, Jul 1967.

[2] Loup Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.*, 159:98–103, Jul 1967.

[3] W. C. Swope, H. C. Anderson, P. H. Berens, and K. R. Wilson. A Computer Simulation Method for the Calculation of Equlibrium Constants for the Formation of Physical Clusters of Molecules: Applications to Small Water Clusters. *Journal of Chemical Physics*, 76:637–649, 1982.

[4] D. and Beeman. Some multistep methods for use in molecular dynamics calculations. *Journal of Computational Physics*, 20(2):130 – 139, 1976.

[5] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *The Journal of Chemical Physics*, 81(8):3684–3690, 1984.

[6] H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *Journal of Chemical Physics*, 72:2384–2393, 1980.

[7] Shuichi Nosé and M.L. Klein. Constant pressure molecular dynamics for molecular systems. *Molecular Physics*, 50(5):1055–1076, 1983.

[8] Shuichi Nosé. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*, 81(1):511–519, 1984.

[9] M. Parrinello and A. Rahman. Polymorphic transitions in single crystals: A new molecular dynamics method. *Journal of Applied Physics*, 52(12):7182–7190, 1981.

Chapter 4

Analysis and Applications of Molecular Dynamics

Contents

4.1 Introduction

Often we start our simulation with initial velocities and positions that are not representative of the state condition of interest (e.g. as specified by the temperature and density). As such, we must equilibrate our system by first running the simulation for an amount of time that lets it evolve to configurations representative of the target state conditions. Once we are sure we have equilibrated, we then perform a production period of simulation time that we used to study the system and/or compute properties at the target state conditions.

One approach to check if we have equilibrated our system is to monitor the time-dependence of simple properties, like the potential energy or pressure.

Measuring quantities in Molecular Dynamics usually means performing time average of physical properties over te system trajectory. Physical properties are usually a function of the particle coordinates and velocities. So for instance we can define the instantaneous value of a generic physical property A at time t,

$$A(t) = f(\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t), \mathbf{v}_1(t), \ldots, \mathbf{v}_N(t)) \tag{4.1.1}$$

and then obtain its average,

$$\langle A \rangle = \frac{1}{M} \sum_{t=1}^{M} A(t) \tag{4.1.2}$$

where t is an index which runs over the time steps from 1 to the total number of steps, M. This can be be added in the sampling function of the Molecular Dynamics simultion, Algorithm 4.1.

```
function [h] = samples(step,nSteps,sampleFreq,vels,coords,L,h)

    % Initialisation
    if step==1
        % Any code needed to set up the sampling
    end
    % Sampling
    if mod(step,sampleFreq) == 0
        % Code for sampling on given steps
    end

    % End
    if step==nSteps
        % Code at end of simulation
    end
end
```

Algorithm 4.1: Function forrunning samples at designated steps.

4.2 Energies

4.2.1 Kinetic Energy

The instantaneous kinetic energy is given by,

$$K(t) = \frac{1}{2} \sum_i m_i |\mathbf{v}_i(t)|^2 \tag{4.2.1}$$

Which can be added to the Molecular Dynamics simulation as Algorithm 4.2.

```
sumv2=0;
for part=1:nPart
    % Calculate total v2
    sumv2=sumv2+dot(vels(:,part),vels(:,part));
end
KE=0.5*m*sumv2/nPart
```

Algorithm 4.2: Code to calculate the instantaneous kinetic energy.

4.2.2 Potential Energy

The potential energy, V, is obtained by based on the force field, in the case of two-body interactions,

$$V(t) = \sum_i \sum_{j>i} \phi \left| \mathbf{r}_i(t) - \mathbf{r}_j(t) \right| \tag{4.2.2}$$

This is often calculated along with the force calculation for convenience with Algorithm 4.3 for the Lennard-Jones potential.

```
en=0;
% cut off energy
ecut=(1/(L/2)^12)-(1/(L/2)^6);
% Loop over all particle pairs
for partA = 1:nPart-1
    for partB = (partA+1):nPart
        % Calculate particle-particle distance
        dr = coords(:,partA) - coords(:,partB);
        % Fix according to periodic boundary conditions
        dr = distPBC3D(dr,L);
        % Get the distance squared
        dr2 = dot(dr,dr);
        % Check if the particles are within the cutoff
        if dr2 < (L/2)^2
            % Lennard-Jones potential:
            % U(r) = 4*epsilon* [(sigma/r)^12 - (sigma/r)^6]
            %
            % U(r) = 4 * [(1/r)^12 - (1/r)^6]
            invDr2 = 1.0/dr2; % 1/r^2
            invDr6 = invDr2^3;
            en=en+invDr6*(invDr6-1)-ecut;
        end
    end
end
% Multiply energy by 4
en=en*4;
```

Algorithm 4.3: Code to calculate the instantaneous potential energy for a Lennard-Jones system.

The term $ecut$ is the value of the potential as $r = r_c$ (the cut off), for the Lennard-Jones potential we have,

$$ecut = 4\left(\frac{1}{r_c^{12}} - \frac{1}{r_c^6}\right) \tag{4.2.3}$$

4.2.3 Total Energy

The total energy, $E = K + V$, is a conserved quantity in Newtonian dynamics. However, it is common practice to compute it at each time step in order to check that it is indeed constant with time. During the simulation energy flows back and forth between the kinetic and potential energy causing them to fluctuate while their sum remains fixed.

In practice there could be small fluctuations in the total energy. These fluctuations are usually caused by errors in the time integration and can be reduced in magnitude by reducing the time step. Slow drifts of the total energy are also sometimes observed in very long runs. Such drifts could also originate from large time steps. Drifts are more disturbing than fluctuations because the thermodynamic state of the system is also changing with the energy. If drifts over long runs tend to occur, they can be prevented, for instance by breaking the long run into smaller pieces and restoring the energy to the nominal value between one piece and the next. A common mechanism to adjust the energy is to modify the kinetic energy via the rescaling of velocities[1].

4.3 Temperature

There is no rigorous microscopic definition of the temperature in the microcanonical ensenble. Instead we can use the macroscopic thermodynamic results,

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_{V,N} = k_B \left(\frac{\partial \ln \Omega}{\partial E}\right)_{V,N} \tag{4.3.1}$$

As mentioned in Chapter 1, it can be shown using the equipartition theorem that the average kinetic energy relates to the temperature via equation 1.2.6. This means that we can calculate the temperature for a time step as,

$$T(t) = \frac{2E_K(t)}{k_B N_f} \tag{4.3.2}$$

Which can be coded as Algorithm 4.4 with use of Algorithm 4.2.

```
1 temp=m*sumv2/(3*nPart);
```

Algorithm 4.4: Code to calculate the instantaneous temperature.

Because the kinetic energy fluctuates during a simulation the temperature also fluctuates. This is an estimator of the temperature and we must perform an average to find the true temperature.

Although it is not fequently used, we can also compute a configurational estimate of the temperature (rather than kinetic estimate) [1],

$$k_B T = \left\langle \frac{\mathbf{F}^N \cdot \mathbf{F}^N}{-\nabla \cdot \mathbf{F}^N} \right\rangle \tag{4.3.3}$$

This estimate depends on the forces and their derivatives. Both the kinetic and configurational temperatures are equal in the limit of infinite simulation time and equilibrum.

[1]Temperature scaling will of course change the total energy of the system as the velocities are being rescaled.

4.4 Pressure

To compute the pressure, we can often use the virial theorem,

$$P = \frac{1}{3V}\left\langle 3Nk_BT + \sum_i \mathbf{F}_i \cdot \mathbf{r}_i \right\rangle \tag{4.4.1}$$

This expression is derived for the canonical ensemble (NVT), but it is often applied to Molecular Dynamics simulations regardless. Though for large systems the difference between NVE and NVT is very small.

The expression above is not generally used for systems of pairwise-interacting molecules subject to periodic boundary conditions. Instead, we can rewrite the force sum,

$$\begin{aligned}
\sum_i \mathbf{F}_i \cdot \mathbf{r}_i &= \sum_i \left(\sum_j \mathbf{F}_{ij} \right) \cdot \mathbf{r}_i = \sum_{i,j} \mathbf{F}_{ij} \cdot \mathbf{r}_i \\
&= \sum_{i<j} \mathbf{F}_{ij} \cdot \mathbf{r}_i + \sum_{i>j} \mathbf{F}_{ij} \cdot \mathbf{r}_i \\
&= \sum_{i<j} \mathbf{F}_{ij} \cdot \mathbf{r}_i + \sum_{i<j} \mathbf{F}_{ji} \cdot \mathbf{r}_j \\
&= \sum_{i<j} \mathbf{F}_{ij} \cdot (\mathbf{r}_i - \mathbf{r}_j) \\
&= -\sum_{i<j} \frac{\mathrm{d}\, u(r_{ij})}{\mathrm{d}\, r} r_{ij}
\end{aligned} \tag{4.4.2}$$

This term is called the virial and contains the sum of pairwise interactions. It is therefore convenient to compute it in the pairwise loop with the forces and energies. For an example with the Lennard-Jones Potential (equation 2.4.4),

$$\sum_i \mathbf{F}_i \cdot \mathbf{r}_i = \sum_{i<j} -24\left(2r_{ij}^{-12} - r_{ij}^{-6}\right) \tag{4.4.3}$$

The virial can be implemented in the Molecular Dynamics simulation as Algorithm 4.5.

```
virial=0;
% Loop over all particle pairs
for partA = 1:nPart-1
    for partB = (partA+1):nPart
        % Calculate particle-particle distance
        dr = coords(:,partA) - coords(:,partB);
        % Fix according to periodic boundary conditions
        dr = distPBC3D(dr,L);
        % Get the distance squared
        dr2 = dot(dr,dr);
        % Check if the particles are within the cutoff
        if dr2 < (L/2)^2
            % Lennard-Jones potential:
            % Fx(r) = 4 * (x/r) * [12*(1/r)^13 - 6*(1/r)^7]
            %       = 48 * x * (1/r)^8 * [(1/r)^6 - 0.5]
            invDr2 = 1.0/dr2; % 1/r^2
            invDr6 = invDr2^3;
            forceFact = invDr2 * invDr6 * (invDr6 - 0.5);
            virial=virial+forceFact*dr2;
        end
    end
end
```

```
% Multiply virial by 48
virial = virial*48;
```

Algorithm 4.5: Code to calculate the instantaneous virial.

Thus the pressure can be calculated as Algorithm 4.6.

```
press = (3*nPart*temp + virial)/(3*L^3);
```

Algorithm 4.6: Code to calculate the instantaneous pressure.

Activity

Add the sampling of the pressure to your Molecular Dynamics program using Algorithms 4.5 and 4.6. How does the pressure change with temperature from $T^* = 0.8$ to $T^* = 1.1$?

How is the pressure variation different if $\rho^* = 0.005$?

Why?

4.5 Diffusion

Diffusion is the process whereby an initially nonuniform concentration profile (e.g. an ink drop in water) is smoothed in the absence of flow. Diffusion is caused by the molecular motion of the particles in the fluid. The macroscopic law that describes diffusion is known as Fick's law, which stats that the flux $\mathbf{j}$ of the diffusing species is proportional to the negative gradient in the concentration of that species,

$$\mathbf{j} = -D\nabla c \tag{4.5.1}$$

where D is the diffusion coefficient. Diffusion of a species in an otherwise identical solvent molecules is called self-diffusion. The concentration profile of the species, under the assumption that at time $t = 0$ the concentration is the initial frame, with time can be given by Fick's law coupled with the conservation of the total amount is,

$$\frac{\partial c(r,t)}{\partial t} + \nabla \cdot \mathbf{j}(r,t) = 0 \tag{4.5.2}$$

Combining with equation 4.5.1 gives,

$$\frac{\partial c(r,t)}{\partial t} - D\nabla^2 c(r,t) = 0 \tag{4.5.3}$$

What we need to link with simulations is actually not an expression for $c(r,t)$ itself but the time dependence of its second moment, $\int \mathrm{d}\mathbf{r} c(r,t) r^2$. Thus we can multiply equation 4.5.3 by r^2 and integrate over all space,

$$\frac{\partial}{\partial t}\int \mathrm{d}\mathbf{r} r^2 c(r,t) = D\int \mathrm{d}\mathbf{r} r^2 \nabla^2 c(r,t) \tag{4.5.4}$$

Defining the fact that $\int \mathrm{d}\mathbf{r} c\left(r,t\right)=1$, i.e. a total amount of material equal to 1, then the left hand side of equation 4.5.4 reduces to $\partial\left\langle r^{2}\left(t\right)\right\rangle /\partial t$, thus,

$$\begin{aligned}\frac{\partial\left\langle r^{2}\left(t\right)\right\rangle }{\partial t} &= D\int \mathrm{d}\mathbf{r} r^{2}\nabla^{2}c\left(r,t\right)\\ &= D\int \mathrm{d}\mathbf{r}\nabla\cdot\left(r^{2}\nabla c\left(r,t\right)\right)-D\int \mathrm{d}\mathbf{r}\nabla r^{2}\cdot\nabla c\left(r,t\right)\\ &= D\int \mathrm{d}\mathbf{S}\left(r^{2}\nabla c\left(r,t\right)\right)-2D\int \mathrm{d}\mathbf{r}\mathbf{r}\cdot\nabla c\left(r,t\right)\\ &= 0-2D\left[\int \mathrm{d}\mathbf{r}\left(\nabla\cdot\mathbf{r}c\left(r,t\right)\right)-\int \mathrm{d}\mathbf{r}\left(\nabla\cdot\mathbf{r}\right)c\left(r,t\right)\right]\\ &= 0+2dD\int \mathrm{d}\mathbf{r}c\left(r,t\right)\\ &= 2dD \end{aligned} \tag{4.5.5}$$

where d is denotes the dimensionality of the system. Equation 4.5.5 relates the diffusion coefficient D to the width of the concentration profile. This relation was first derived by Einstein. It should be noted that whereas D is a macroscopic transport coefficient, $\left\langle r^{2}\left(t\right)\right\rangle$ has a microscopic interpretation; it is the mean-squared distance over which the molecules have moved in a time interval t. Therefore, to find D for every particle i, we measure the distance travelled in time t, $\Delta\,\mathbf{r}_{i}\left(t\right)$,

$$\left\langle r^{2}\left(t\right)\right\rangle =\frac{1}{N}\sum_{i=1}^{N}\Delta\,\mathbf{r}_{i}^{2}\left(t\right) \tag{4.5.6}$$

4.6 References

[1] B. D. Butler, G. Ayton, O. G. Jepps, and D. J. Evans. Configurational temperature: Verification of Monte Carlo simulations. *Journal of Chemical Physics*, 109:6519–6522, 1998.

Appendix A

Translating Instructions to Code

To think about how we translate instructions, be them mathematical equations or written directions, into code let's take the example of making a cake. These instructions can be converted into blocks of code as functions containing actions, e.g. for loops, if statements, etc.

Original Recipe

Wet Ingredients

150g margarine
300ml soya milk
1 tbsp cider vinegar
0.5 tsp vanilla extract

Dry Ingredients

300g plain flour
200g golden caster sugar
4 tbsp cocoa powder
1.5 tbsp baking power
1 tsp bicarbonate of soda

Method

1. Preheat the oven to 190 °C (170 °C for fan).
2. Grease the base and sides of a 25 cm diameter circular tin with margarine, then line the base with baking parchment.
3. Add the wet ingredients together in a bowl and mix together well.
4. In a second bowl, add the dry ingredients together and mix.
5. Add the mixed dry ingredients to the wet ingredients and mix until smooth.
6. Add the cake batter to the prepared tin and bake for 30 mins or until a skewer inserted into the middle of the cakes comes out cleanly.
7. Leave to cool in the tin for 10 mins then turn out onto a wire rack to cool completely.

Translating to Code

To translate the above instructions into code, we can start by taking the steps and defining a function[1] for the steps[2],

```
% Program to bake a cake

% Read in ingredient list
wet_ingredients = [150g margarine, 300ml soya milk, 1 tbsp cider vinegar,
    0.5 tsp vanilla extract];
dry_ingredients = [300g plain flour, 200g golden caster sugar, 4 tbsp cocoa
    powder, 1.5 tbsp baking power, 1 tsp bicarbonate of soda];

% Step 1
[T] = set_oven(oven);

% Step 2
[ready_tin] = prepare_tin(tin,marge,parchment);

```

[1]Functions are "self contained" modules of code that accomplish a specific task. Functions usually take in data, process it, and return a result. Once a function is written, it can be used over and over and over again.

[2]NB. Anything following a % sign is a comment only and is not used when the program is ran.

```
% Step 3-5
[mix_bowl] = make_cake(wet_ingredients,dry_ingredients)

% Step 6-7
[cake] = bake_cake(mix_bowl,ready_tin,T);

% Eat
Eat
```

Algorithm A.1: Bake a cake.

We can now work through the steps in order and pick out the key information to fill in the functions we need,

1 Preheat the oven to 190 °C (170 °C for fan).

The key things here are to set the temperature based on the type of oven; therefore, we check the oven type an use this test to allow us to define the correct temperature,

```
function [T] = set_oven(oven)

    % Check the oven type
    if oven=="Fan"
        % If the oven is a fan oven
        T=170;
    else
        % If the oven is not a fan oven
        T=190;
    end
end
```

Algorithm A.2: Step 1.

2 Grease the base and sides of a 25 cm diameter circular tin with margarine, then line the base with baking parchment.

In this step, we are adding items together; the tin, the margarine, and the baking parchment. The key thing we want here is variables that make sense so we remember what they are,

```
function [ready_tin] = prepare_tin(tin,marge,parchment)

    grease_tin = tin + marge;
    ready_tin = grease_tin + parchment;

end
```

Algorithm A.3: Step 2.

3 Add the wet ingredients together in a bowl and mix together well.

4 In a second bowl, add the dry ingredients together and mix.

5 Add the mixed dry ingredients to the wet ingredients and mix until smooth.

The next 3 steps have been grouped together due to their similarity. We need to set up the size of the bowls, then add the ingredients. We also need to notice that the first bowl for the wet ingredients needs to be large enough for all the ingredients,

```
function [mix_bowl] = make_cake(wet_ingredients,dry_ingredients)

    % Find the number of wet and dry ingredients
    nWet = size(wet_ingredients);
    nDry = size(dry_ingredients);

    % Get the bowls and set to the right size with zeros, i.e. empty bowls
    % wet bowl, but large enough for all
    mix_bowl = zeros(nWet+nDry);
```

```
    % dry bowl
    dry_bowl = zeros(nDry);

    % Add wet ingredients to bowl
    [mix_bowl] = mix_ingredients(wet_ingredients,mix_bowl);
    % Add dry ingredients to bowl
    [dry_bowl] = mix_ingredients(dry_ingredients,dry_bowl);

    % Combine into one bowl
    for i = 1:nDry
        mix_bowl(nWet+i) = dry_bowl(i);
    end

    Mix
end
```

Algorithm A.4: Step 3-5.

For steps 3 and 4 we can actually use the same function, and as we have to do the same activity several times we can use a for loop,

```
function [bowl] = mix_ingredients(ingredients,bowl)

    % Add all the ingredients to the bowl
    for i in 1:size(ingredients)
        bowl(i) = ingredients(i);
    end

    Mix
end
```

Algorithm A.5: Step 3-4.

6 Add the cake batter to the prepared tin and bake for 30 mins or until a skewer inserted into the middle of the cakes comes out cleanly.

7 Leave to cool in the tin for 10 mins then turn out onto a wire rack to cool completely.

Now we place the cake mix into the prepared tin and bake. The important information to take here is that we need to wait 30 minutes, unless our test for the cake being baked is already true,

```
function [cake] = bake_cake(mix_bowl,ready_tin,T);

    % Put the cake and the tin together
    to_bake.cake_tin = ready_tin;
    to_bake.batter = mix_bowl;

    % Put the cake in the oven at the correct temperature
    Oven.T = T;
    Oven.bake = to_bake;

    % Set the initial time, i.e. 0 minutes, for starting to bake
    t = 0;

    % Check if the time is over 30 minutes
    while t < 30
        % Time increase by 1 minute
        t=t+1;

        % Test cake readiness, skewer will return either "dirty" or "clean"
        [skewer] = test_cake;

        % If the skewer is clean the cake is ready, so exit the loop
        if skewer == "clean"
            break
        end
    end

    % Take the cake out of the oven
    done_cake = Oven.bake;

```

```
31      % Cool for 10 minutes
32      % Reset time to 0
33      t = 0;
34
35      % Check if time is over 10 minutes
36      while t < 10
37          % Time increase by 1 minute
38          t=t+1;
39      end
40
41      % Take the cake out the tin
42      cake = done_cake.batter;
43
44  end
```

Algorithm A.6: Step 6-7.

Appendix B

Some Coding Tips

Modularization

- Programs should use good modular design - the program should be broken into suitable functions and sections
- Each function should perform a single well-defined operation and be designed thinking of reuse of that function e.g. handling of variables into and out of the function
- If functions are associated with m-files it must be obvious what m-file is what function

Design

- Program should be written in a clear and logical manner
- The most appropriate algorithms should be used e.g. differentials of functions explicitly used, constants taken out of loops
- Avoid slow operations e.g. square roots, high powers
- Pre-allocation of arrays is faster than building them within loops
- Do not use global variables - pass variables into functions if needed. Global constants are acceptable if needed

Readability

- Use indentation appropriately and consistently to delineate code blocks
- Use meaningful identifier names, e.g., function and variables names
- Make sure variable names are consistent
- Use appropriate white space between code blocks
- Use white space within lines of code. Operators such as $=$ and $+$ and $<<$ should have a space on either side
- Lines of code should be a reasonable length - Break long lines into shorter lines when possible
- Use a non-proportional font, so indentation is meaningful (each character takes up same amount of space).

Comments

- The contents of each file are listed if multiple functions are in each file
- The contents of each function are described at the beginning of the function - what is the purpose of each function and any assumptions made
- Comments are used to explain key parts of the code

Function

- Programs must work correctly
- It needs to be clear where input parameters are given to the program and what these input parameters are
- The code produces sensible outputs, which can be easily used for analysis