4 Introduction to Molecular Dynamics

In this set of exercises, you will be introduced to the basic concepts of Molecular Dynamics which you will apply to a simple test system, a water box. You will then quantify the averaged structural properties of the system using its pair radial distribution function.

4.1 The Ergodic Hypothesis in Statistical Mechanics

The question whether some properties obtained by averaging over a thermodynamical ensemble (the *ensemble average*) are equal to a *time average* of said properties - *i.e.* whether a system is *ergodic* or not - poses one of the fundamental problems of statistical mechanics. Unfortunately, there exists no complete proof of an *ergodic theorem* applied to thermodynamic ensembles. However, it can be shown that, if the only constants of motion of the system are constant functions over phase space (*i.e.* constants independent of coordinates and momenta), then the ensemble and time averages are identical for $t \to \infty$. Since a more general ergodic theorem has not been proven yet, a system is usually considered ergodic as long as *all* regions of phase space are accessible during the time t for which the system is sampled. In practice, the *ergodic hypothesis*,

$$\langle O \rangle = \int_\Gamma \frac{1}{Z} O(\Gamma) e^{-\beta \mathrm{H}(\Gamma)} \mathrm{d}\Gamma = \lim_{t\to\infty} \frac{1}{t} \int_0^\tau O(\tau) \mathrm{d}\tau, \tag{1}$$

is thus assumed - but not guaranteed - to hold.

4.2 Sampling Phase Space using Molecular Dynamics

Under the ergodic hypothesis, a direct sampling of phase space configurations can be replaced by a sampling of the dynamic evolution of the system for long enough times t. For an NVT ensemble, an observable of a state s will then be explored with a probability corresponding to its Boltzmann weight, and for a sufficiently long t - *i.e.* for a sufficiently high occurence of all events to be representative - the time-average over the $O(s)$ will reproduce the ensemble average. The longer the simulation time t is chosen, the better the convergence of the time average towards the ensemble average (it is evident that too short t do not lead to a converged dynamics, *i.e.* phase space is not properly sampled).

In this spirit, a (suitable) starting conformer of a system can be propagated in time according to Hamilton's equations of motions:

$$\frac{\mathrm{d}\mathbf{p}_i}{\mathrm{d}t} = -\frac{\partial H}{\partial \mathbf{q}_i}, \tag{2}$$

$$\frac{\mathrm{d}\mathbf{q}_i}{\mathrm{d}t} = \frac{\partial H}{\partial \mathbf{p}_i}. \tag{3}$$

This time-evolution is completely general. If the system is represented by a set of classical point particles, the Newtonian formulation of classical mechanics can be applied to the problem, and the particles can be propagated by evaluating the force acting on them:

$$\mathbf{F}_I = \nabla_I E \tag{4}$$

$$= m_I \mathbf{a}_I \tag{5}$$

For small time steps $\Delta t = \tau$, the particles are accelerated according to a which is determined from $\frac{\nabla_I E}{m_I}$ at $t = 0$. At time $t = \tau$, the forces and the acceleration are re-evaluated, and the system is moved according to the updated forces that act on it. These forces may be obtained from quantum mechanical calculations (*first principles* dynamics, *cf.* the discussion of the Hellmann-Feynman theorem) or from a parametrised form for $E(\mathbf{r})$ (*classical dynamics*, which will be discussed in detail in the following lecture). Since the evolution of the system is well-defined at every point based on the forces acting on it, its dynamics will be *deterministic*. Given an initial set of positions and momenta, every point ever visited by the system at time t is pre-determined. Applying this approach to molecular systems results in either *classical* or *first-principles molecular dynamics* (MD).

4.3 Structural Properties from MD

4.3.1 Periodic Boundary Conditions

Simulating for long times t ensures that the ensemble average can be approached, however, it is impossible to sample in the limit of the ergodic theorem $t \to \infty$. Additionally, for bulk-property calculation it is necessary to use a sufficiently large number of molecules to ensure that regions of phase space are sampled representatively, such that one may be confident that the ensemble average is properly reconstructed from the time average. In practice there is a relatively small and finite number of molecules for which simulation is computationally feasible, hence, compared to a macroscopic system ($\sim N_A$ molecules), the ratio of molecules near the surface of the simulation box is often too large to be representative. Computational modelling of molecular systems could therefore have an artificially imposed doping of surface effects which negatively impacts the calcuation of any bulk property of interest. To remedy this, surface effects can be disposed of for all system sizes if periodic boundary conditions (PBC) are imposed. In this regime, the simulation box is replicated through space to form an infinite lattice. When a molecule moves during simulation its periodic images move with the exact same displacement, thus, if a molecule leaves the central box, one of its images will enter through the opposite face. This is illustrated in Figure 1: there are no walls at the boundary of the central box and the system has no surface.

It is not necessary to store the coordinates of all images in a simulation (this would require infinte space). When a molecule leaves the box by crossing a boundary, attention may be switched to the identical molecule entering from the opposite side.

4.3.2 Pair Radial Distribution Functions $g(r)$

Radial distribution (Pair correlation) functions are of fundamental importance in thermodynamics, since macroscopic thermodynamic properties can usually be calculated directly from $g(r)$. In short, they simply describe how probability density varies as a function of distance from a reference particle.

The partition function Z can be evaluated, in principle, by carrying out the integrations for a substance with a known potential function. However, this task is very difficult due to the very large number of molecules involved in real systems. A more convenient

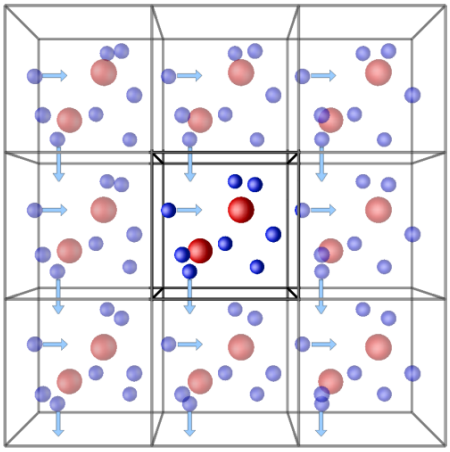

Figure 1: Simulation box and its periodic images. When a molecule leaves the central box, it is wrapped around to the opposite side.

formulation is based on the concept of distribution functions. The probability $P(N)$ of finding molecule 1 in volume element $\mathrm{d}\mathbf{r}_1$ at $\mathbf{r}_1$, molecule 2 in volume element $\mathrm{d}\mathbf{r}_2$ at $\mathbf{r}_2$, ..., and molecule N in volume element $\mathrm{d}\mathbf{r}_N$ at $\mathbf{r}_N$ is given by:

$$P(N)\mathrm{d}\mathbf{r}_1 \dots \mathrm{d}\mathbf{r}_N = \frac{1}{Z_c(N,V,T)} e^{-\frac{U_N}{k_B T}} \mathrm{d}\mathbf{r}_1 \dots \mathrm{d}\mathbf{r}_N, \tag{6}$$

where $U_N(\mathbf{r}_1, \dots, \mathbf{r}_N)$ is the potential energy due to the interaction between particles and $Z_c(N,V,T)$ is the configurational integral,

$$Z_c(N,V,T) = \int \dots \int e^{-\frac{U_N}{k_B T}} \mathrm{d}\mathbf{r}_1 \dots \mathrm{d}\mathbf{r}_N, \tag{7}$$

taken over all possible combinations of atomic particle positions. Generally the total number of particles is massive enough such that $P(N)$ is not particularly useful. It is more informative to consider the relative position of two molecules irrespective of the location of other molecules in the system. Integrating equation 6 over all coordinates except those pertaining to the two molecules of interest, one obtains the definition of the second-order disitribution function $p^{(2)}(\mathbf{r}_1, \mathbf{r}_2)$, which gives the probability of finding molecule 1 in volume element $\mathrm{d}\mathbf{r}_1$ at $\mathbf{r}_1$ and molecule 2 in volume element $\mathrm{d}\mathbf{r}_2$ at $\mathbf{r}_2$:

$$p_{ij}^{(2)}(\mathbf{r}_1, \mathbf{r}_2) = N_i(N_j - \delta_{ij}) \frac{1}{Z_c(N,V,T)} \int \dots \int e^{-\frac{U_N}{k_B T}} \mathrm{d}\mathbf{r}_3 \dots \mathrm{d}\mathbf{r}_N, \tag{8}$$

where δ_{ij} is the Kronecker delta. Note that $p^{(2)}(\mathbf{r}_1, \mathbf{r}_2)$ depends on temperature, density and composition additionally to $\mathbf{r}_1$ and $\mathbf{r}_2$. For molecules which interact with radially symmetric potential functions $p^{(2)}(\mathbf{r}_1, \mathbf{r}_2)$, in the fluid state, depends only on the distance between centres of masses $r_{12} = |\mathbf{r}_1 - \mathbf{r}_2|$. In the limit of ideal gas ($\frac{U}{k_B T} \to 0$), the distribution function $p^{(2)}(\mathbf{r}_1, \mathbf{r}_2)$ approaches the value $N_i(N_j - \delta_{ij})/V^2$. This suggests

defining the pair radial distribution function, $g_{ij}^{(2)}(r)$ by:

$$g_{ij}^{(2)}(r) = \frac{p_{ij}^{(2)}(r)V^2}{(N_i N_J)}, \tag{9}$$

which approaches $1 - \delta_{ij}/N_j$ in the above limit. Combining equations 8 and 9 gives:

$$g_{ij}^{(2)}(r) = V^2 \left(1 - \frac{\delta_{ij}}{N_j}\right) \frac{1}{Z_c(N,V,T)} \int \dots \int e^{\frac{-U_N}{k_B T}} \mathrm{d}\mathbf{r}_3 \dots \mathrm{d}\mathbf{r}_N, \tag{10}$$

which is the second-order correlation function (pair radial distribution function). If the system consists of spherically symmetric particles $g_{ij}^{(2)}$ depends only on the relative distance between them $r_{ij} = r_j - r_i$. Taking particle 0 as fixed at the origin of the coordinate system, $\rho g(r)\mathrm{d}r = \mathrm{d}n(r)$ is the number of particles (among the remaining $N-1$) to be found in the volume $\mathrm{d}r$ around the position r. These particles can then be formally counted as:

$$\mathrm{d}n(r) = \left\langle \sum_{i \neq 0} \delta(r - r_i) \right\rangle \mathrm{d}r \tag{11}$$

where $\langle \dots \rangle$ denotes the ensemble average, yielding:

$$g(r) = \frac{1}{\rho} \left\langle \sum_{i \neq 0} \delta(r - r_i) \right\rangle = V \frac{N-1}{N} \langle \delta(r - r_1) \rangle \tag{12}$$

where the second equality requires the equivalence of particles $1, \dots, N-1$.

4.4 Tinker

Tinker is a complete package for performing empirical force field molecular dynamics calculations, and will be used throughout the remaining lab sessions. The following section provides a brief introduction while those thereafter provide instructions and exercises.

4.4.1 Introduction

Tinker is maintained by Professor Jay William Ponder at the Washington University School of Medicine. Tinker is written in Fortran95 and works on Windows, Mac, and Linux. Its source code is available free of charge under a restrictive license. It is intended to serve as a platform for algorithm development and parameterisation, while still being sufficient for most production work. Rather than incorporating all the functionality in one monolithic program, Tinker provides a set of relatively small programs that interoperate to perform complex computations. New programs can be easily added by developers with only limited programming experience. The central component of the Tinker package is a modular set of routines which allow the manipulation of coordinates and evaluation of potential energy and derivatives in a straightforward fashion.

The series of major programs included in the distribution perform the following core tasks: (1) energy minimisation over Cartesian coordinates, torsional angles or rigid bodies, (2) molecular, stochastic and rigid body dynamics with periodic boundaries and temperature/pressure control, (3) multiple time step RESPA integration for efficient MD simulation, (4) building protein and nucleic acid structures from sequence, (5) analysis and breakdown of single point potential energies, (6) potential energy surface search, (7) free energy calculations, (8) analysis of and comparison to electrostatic potentials, (9) fitting of intra-and intermolecular potential parameters to structural and thermodynamic data, and (10) global optimization via simulated annealing, Monte Carlo minimisation, and energy surface smoothing methods.

4.4.2 Using Tinker

Tinker has been pre-installed on the computer room machines. If you wish to use your own laptop or desktop instead, please refer to section 4.8 for information about compiling Tinker from source.

To run Tinker calculations you need two files: the Tinker `.xyz` file and the Tinker `.key` file. The first file contains coordinate information of the system of interest while the second file contains the instructions and properties necessary for the calculation being performed such as: cutoff distance, potential/force-field parameters and thermostat information *etc.* It is worthwhile to note that the `.xyz` file contains the progressive number of atoms in the system, atomic name, X-, Y-, Z- coordinates, type and connectivity between atoms.

In this exercise, you are asked to run a molecular dynamics simulation of a box of water molecules [24.622, 24.622, 24.622 ($Å^3$)]. The files necessary for this exercise are provided on the exercise webpage. Those are `waterbox.xyz`, `waterbox.key` and `waterbox.dyn`, which are the inputs to the molecular dynamics simulation. The `waterbox.dyn` file is simply a restart file from a previous molecular dynamics run. Create a new folder named `Exercise_4/` within the `Desktop/` directory and copy those necessary files.

To run the simulation use the following command:

```
/opt/tinker/dynamic waterbox.xyz -k waterbox.key 10000 1.0 0.1 2 298.0
> md.log &
```

This command specifies that you wish to perform molecular dynamics upon the `waterbox.xyz` system using parameters defined within the `waterbox.key` file. The additional parameters passed by command-line are:

a) Number of steps: 10000.

b) Time step length: 1.0 femtoseconds.

c) Record the output coordinates at every 0.1 picosecond.

d) Sample the canonical ensemble (Option 2).

e) Temperature held constant at 298.0K.

And finally the output of the simulation is piped to the file `md.log` which you can track the progress of by typing the following command:

```
tail -10 md.log
```

where the `-10` flag will make `tail` print the last 10 lines of the `md.log` file to your terminal.

4.5 Creating the Trajectory

The above dynamics will produce 100 snapshots of the system, saved in the files named 'waterbox.001...waterbox.100'. These files contain the coordinates of the trajectory at every 0.1 picosecond. To visualise this trajectory, you must first compile them together into a single archive using the following program:

```
/opt/tinker/archive
```

You will be asked to enter '1' to indicate that you are compressing a set of frames and then the name of the coordinate archive file. Enter 'waterbox'. You will now need to enter the frames over which the dynamics was recorded, so enter '001 100'. This will produce a `waterbox.arc` file which can be loaded into `VMD`.

4.6 VMD: Pair Radial Distribution Function

To calculate the pair radial distribution function $g(r)$ from the waterbox you have just simulated, first open the waterbox.arc file with VMD using the following command:

```
vmd waterbox.arc
```

If the `waterbox.arc` file produced above cannot be loaded into VMD immediately, you will need a small amount of editting. Open the file in `vi` as follows:

```
vi waterbox.arc
```

then type the following command:

```
:%s/    24.662000.*\n//ge
```

and press enter. Note that there are 4 spaces between the first `/` and the numeral `2`. This command will remove the second line (containing simulation box information) from every frame in the archive.

Once the waterbox.arc file can be automatically recognised as a Tinker file, you can calculate the pair radial distribution function:

a) Click on Extensions → Analysis → Radial Distribution Function $g(r)$. This will open a new window.

b) Select your system in "Use Molecule". Click on Utilities → set unit cell dimensions and modify Length [a, b, c] to [24.662, 24.662, 24.662 (Å^3)]. The Angles $[\alpha, \beta, \gamma]$ should be left at [90, 90, 90] respectively. Finally click 'Set unit cell'.

c) Enter 'name O' into the Selection 1 field, and 'name O' into the Selection 2 field. This will calculate $g(r)$ between oxygen atoms solely.

d) Select the following boxes to true: use PBC, Display $g(r)$, Display $\int g(r)dr$ and Save to file.

e) Click compute $g(r)$.

You will be presented with two graphs, $g(r)$ and $\int g(r)dr$. You will also be asked to save the data to file; enter an appropriate name and save this file if you wish to replot using `gnuplot`. Additionally you can save both of these graphs to an image by selecting `File` → `Export to PostScript`.

4.7 Exercises

4.7.1 Sampling Phase Space using Molecular Dynamics

a) How does an MD program work? Describe schematically how you would perform a molecular dynamics simulation. Point out the main differences between your scheme and Monte Carlo methods. **Bonus:** Give your answer in pseudocode.

b) Describe a possible implementation of periodic boundary conditions. **Bonus:** Give your answer in pseudocode.

c) Why are most MD and MC simulations based on periodic systems? Explain the main purpose of periodic boundary conditions in these schemes.

4.7.2 Structural Properties from Molecular Dynamics

a) Provide the $g(r)$ plot in your report, indicating the locations of the solvation shells, noting that r has the units of Ångstroms (Å).

b) Provide the $\int g(r)dr$ plot in your report, what can you specifically infer from this graph?

c) Figure 2 presents various $g(r)$ pairs determined experimentally for bulk water at 298.0K. Comment upon and list reasons for any major differences between this and the $g_{OO}(r)$ you have calculated.

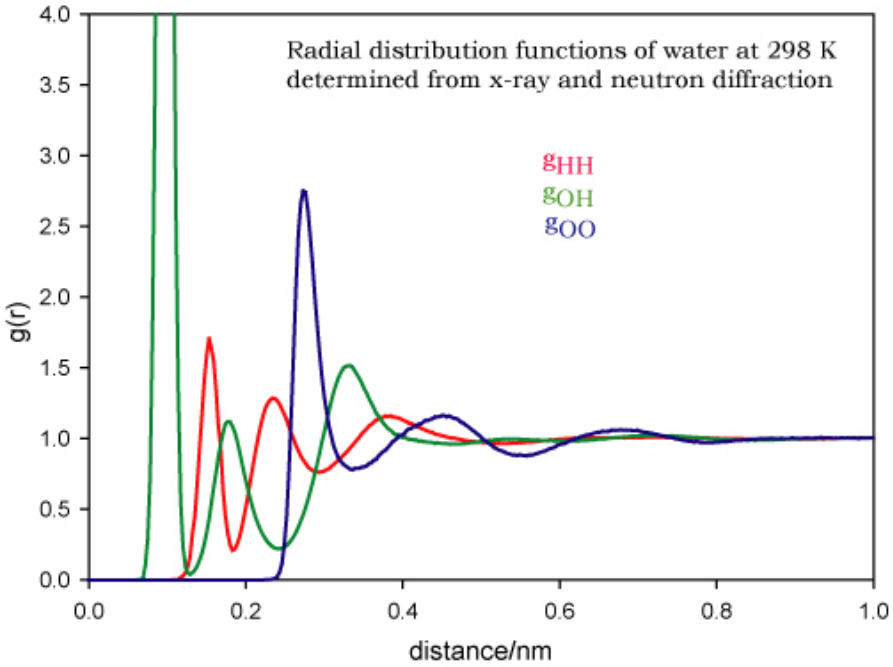

Figure 2: Experimental $g(r)$ of water at 298.0K.

d) Recompute $g(r)$ with the PBC condition turned off and provide the plot in your report. Explain the changes in the $g(r)$ graph.

e) Perform the MD for 5000 and 20,000 steps using the previous conditions. Calculate and provide in your report the corresponding $g(r)$ and $\int g(r)dr$ graphs, and comment upon the effect of increased sampling time upon $g(r)$ calculation.

4.8 Tinker: Installation

If you wish to install Tinker onto your laptop or machine, you will find the following information useful.

4.8.1 Requirements

This section assumes you are using MacOSX. If you are using a Linux or Windows machine, the following information is still relevant, but in addition you will find section 4.9.1 useful.

4.8.2 Download

To download Tinker 6.3.3, navigate to http://dasher.wustl.edu/tinker/ and scroll down to the `Tinker Downloads` section. Here you will find a file labelled `Previous Release (Tinker 6.3.3, GNU gzip)`. Download and move this file to your desktop.

Open a terminal instance, navigate your terminal focus to `Desktop/` and decompress the archive file using the following command:

```
tar -zxf tinker-6.3.3.tar.gz
```

This will produce a folder called `tinker/` which contains the entire source code of Tinker 6.3.3: including multiple examples, test scripts, parameters for force fields...*etc.* Now type the following commands to navigate to the `tinker/` folder, create a directory called `bin/` and finally move to the `source/` directory:

```
cd tinker/
mkdir bin
cd source
```

4.9 Compilation

The following commands will be needed to compile Tinker 6.3.3. You must ensure that you are launching these scripts from the `source/` directory.

```
../mac-osx/gfortran/compile.make
../mac-osx/gfortran/library.make
../mac-osx/gfortran/link.make
```

The `compile.make` script must be run first and will take approximately 10 minutes to finish. Once this has completed, run the `library.make` script to produce the necessary library files, and finally `link.make` to link all object files to create each Tinker executeable (`*.x`). It is customary to separate source from executeable files, thus you should move the `.x` files to the `bin/` folder you created earlier as follows:

```
mv *.x ../bin
```

You have now succesfully installed Tinker.

4.9.1 Compilation: Windows and Linux

Tinker supports a multitude of Fortran compilers and has provided individual make scripts for the common distributions, however, it is recommended that you use gfortran for consistancy with this script. If you do not have gfortran already installed, you will find the instructions on the GCC Wiki website most useful (https://gcc.gnu.org/wiki/GFortran).

You will find operating system-specific compile, library and link scripts in the `tinker/` folder. This process is identical to that mentioned in section 4.9.