Statistical methods in atomistic computer simulations

Prof. Michele Ceriotti, michele.ceriotti@epfl.ch
Piero Gasparotto, piero.gasparotto@epfl.ch

Hands-on exercises on a Lennard-Jones 38 cluster

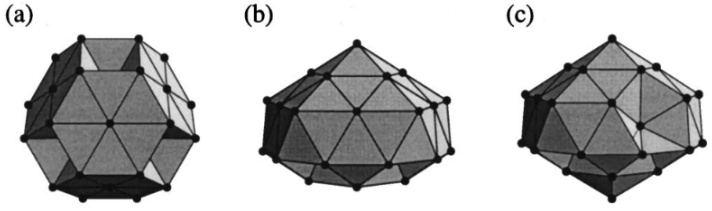
The system that will be examined is a clusters of 38 atoms interacting with a simple Lennard-Jones (LJ) potential:

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

where ϵ is the well depth and $2^{\frac{1}{6}}\sigma$ is the equilibrium separation for a diatomic molecule. This potential describes dipole fluctuation attractive interactions that decay as r^{-6}, and a somewhat arbitrary r^{-12} repulsive wall at short inter-atomic separations, that models the Pauli repulsion between electron clouds. The Lennard-Jones potential is a good model for the interaction between noble gases atoms, but here we will use it just as an inexpensive model of an isotropic pair-wise interaction between atoms. In all the exercises reduced units will be used, that correspond to measuring energies in units of ϵ, distances in units of σ, time in units of $t^* = \sqrt{m\sigma^2/\epsilon}$, and temperature in units of $T^* = (\epsilon/k_B)$. In practice this amounts at setting to one most constants: in principle all results can be scaled to the physical values for a particular system by setting the appropriate mass, well depth and equilibrium distance.

A LJ cluster provides a particularly useful model system since for small LJ clusters a complete enumeration of the minima and transition states allows a detailed view of the potential energy landscape [1]. In particular the LJ_{38} , the cluster which we study here, has a double-funnel landscape: the global minimum is a face-centered-cubic (fcc) truncated octahedron (Fig. 1(a)) and the second lowest energy minimum is an incomplete Mackay icosahedron (Fig. 1(b)).

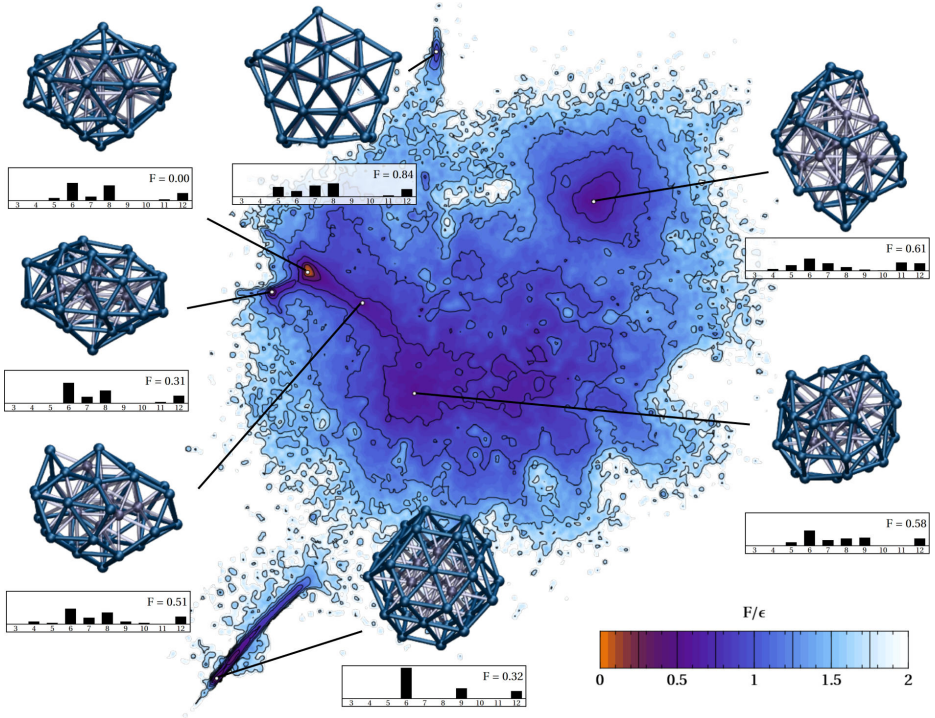
Figure 1: (a) The LJ_{38} global minimum, an fcc truncated octahedron. (b) and (c) Second lowest energy minimum of LJ_{38}. Figure adapted from [2].



There is thus a solid–solid transition at moderate temperatures and a subsequent solid–liquid transition at higher temperatures. The solid–solid transition occurs because the energy landscape in the high-temperature phase is flatter, resulting in a larger entropic contribution to the free energy of this structure, and to its stabilization at moderate temperatures relative to the *fcc* minimum [3]. Figure 2 shows a few selected configurations for this system, together with the free energy computed close to the solid-liquid transition

temperature. The stability of different configurations depends dramatically on the simulation temperature, and the time scales for transitions is long, but accessible to direct simulation thanks to the small size and inexpensive potential. This makes LJ_{38} an ideal example to show the strength and pitfalls of different sampling techniques in atomistic simulations.

Figure 2: A number of representative configurations of the LJ_{38} cluster are projected together with the free energy surface computed at $0.18T^*$. Figure adapted from [3]



General remarks

These exercises are structured as a hands-on tutorial, in which you are given simple, rudimentary programs that (should) already work. You are encouraged to look at the source code to understand how things work, and you are more than welcome if you want to clean up the code, include better comments or more explanatory error messages.

You should be able to finish most of the exercises in two hours. Please keep your results in order, as you will be asked to put them together in a short report in which you explain what you learned from each part of the tutorial, and since at times you will have to compare results obtained from different exercises.

Always **change the random number seed** in the example input files before you run: in this way each one will have statistically independent results, and it will be possible to combine the results of different groups to get more converged statistics. It is expected for you to be familiar with running programs from a UNIX shell. If this is not the case, it is highly recommended that you follow one of the many excellent tutorials that can be found on-line before you start.

Utility programs

The programs used to *run* simulations have been developed specifically for this course, and are distributed as a package together with these notes. However, a few utilities will be used to post-process the simulation output – for instance to compute histograms and autocorrelation functions. These should be downloaded from their `git` repositories, and then compiled:

```
$ git clone https://github.com/cosmo-epfl/toolbox.git
$ cd toolbox/src
$ make
$ cd ..
```

You may need to create a `make.in` file based on `make.in.example`, and to install lapack and fftw3 to have access to all the features of this set of utilities. On a Debian/Ubuntu system, you should be able to do so by running

```
$ cp make.example.in make.in
$ sudo apt-get install libfftw3-dev liblapack-dev
```

For the last exercise, you will also need the `sketchmap` suite of dimensionality reduction programs. You should be able to get it from the repository

`https://github.com/cosmo-epfl/sketchmap.git`

Lastly, in some steps you will be asked to visualize your structure trajectories using the `vmd` command. In order to use it, you'll have to install it from its webpage

`http://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD`

1 Monte Carlo

This exercise is meant to be a simple introduction to Metropolis Monte Carlo (MC), and a way to get familiar with the LJ_{38} system and with the programs we will use to post-process the simulation data. You will find the source code in the `src/` directory, that contains

`mcnvt.f90`	The main file, containing the initialization of the system and the Metropolis MC algorithm
`routines.f90`	The module containing the principal functions and subroutines used in the `mcnvt.f90` file, such as the calculation of the energy, the i/o routines, ...
`random.f90`	Routines to sample a uniform and Gaussian random numbers
`Makefile`	The file containing the compilation parameters

First, try to familiarize yourself with the source, which is heavily commented and should allow you to easily follow the algorithms. Compilation is straightforward:

```
$ pwd
~/exercises/ex1
$ cd src
$ make
$ cd ..
```

This will generate a binary file called `mcnvt`, that you will use to run the exercises. The input file for `mcnvt` contains the following options:

`&inp`	
`seed = 1357`	Initial seed for the random number generator
`temp = 0.23`	Target temperature
`dataxyz = 'lj38.xyz'`	Starting configuration (xyz format)
`nstep = 10000000`	Number of steps to be performed
`stridetrj = 10000`	Output stride for the trajectory
`stridelog = 100`	Output stride for the simulation logs
`mcstep = 0.001`	Variance of the Gaussian random numbers used to generate trial configurations
`outputf = 'out.xyz'`	Output trajectory file
`&end`	

1.1 MC equilibration

In this exercise we will study the equilibration of the system in the NVT ensemble. The proposed target temperature is $0.23\,T^*$. The starting initial configuration contained in the file `lj38.xyz` is the *fcc* truncated octahedron relaxed to $0\,T^*$.

Launch the program with the following command:

```
$ ../src/mcnvt input > log &
```

This will output the statistics in the file `log` and will run the calculation in background so you can monitor the equilibration of the system using `gnuplot`. For instance, to plot the potential energy as a function of the number of MC steps:

```
$ gnuplot
```

```
gnuplot> p 'log' u 1:2 w l
(alternatively: gnuplot> plot 'log' using 1:2 with line)
```

- Observe how the potential energy converges to the equilibrium value
- Look at the trajectory file using

  ```
  $ vmd out-023.xyz
  ```

  and experiment with the visualization options of `vmd` (particularly in the menu graphics -> representations).
- Check how the equilibration varies when changing the input parameters, such as `mcstep` and the target temperature
- The transient that is observed at the beginning of the equilibration is related to the sampling efficiency (because of the connection between relaxation to equilibrium and fluctuations in the equilibrium ensemble). The sampling efficiency can be estimated by computing the autocorrelation time.
  - use the `autocorr` tool to compute the autocorrelation function from one of the logs

    ```
    $ tail -n +10000 log | awk '!/#/{print $2}'
    | autocorr -maxlag 10000 -timestep 100 > acf
    ```

    The `-maxlag` option specifies the time window in which to calculate the autocorrelation function, while the `-timestep` flag is used to set the unit of time based on the stride and timestep used for the trajectory, so that the time scale output from `autocorr` corresponds to the correct simulation time. Note that we use `tail` to remove the first part of the trajectory (which consists of the equilibration, so the number of dropped lines should be adjusted depending on the length of the transient!), and `awk` to pick the column that corresponds to the potential energy.
  - the program returns extensive statistical data about the trajectory. The header contains the average and standard deviation of the data set, the autocorrelation time and an estimate of the error in the mean obtained from the variance of the data, the autocorrelation time and the the length of the trajectory.

    ```
    $ head acf
    ...
    #  computed with time unit:    1.0000000e+02     #
    #  mean:      -1.5525889e+02 +- 2.9949957e-01    #
    #  sigma:     2.4443721e+00                      #
    #  a.c. time: 4.5166879e+05                      #
    ...
    ```

    Then, a series of time-dependent diagnostics follows, where the first line indicate the time and the second column the value of the autocorrelation function – you can ignore the other columns. Plot the autocorrelation function using gnuplot, and observe how the decay to zero varies with the MC step.
  - compute the autocorrelation function of trajectories of different length. See how the autocorrelation function requires very long trajectories to reach convergence, and how the tail of the function tends to be noisy. Since `autocorr` evaluates the autocorrelation time as the integral up to the maximum time lag, the `-maxlag` option should be set to be long enough for the function to converge to zero, but not much longer, to avoid picking noise from the tail, which would affect the estimate of the correlation time.

1.2 Optimization of the MC step

Since the autocorrelation time directly relates to the statistical efficiency of sampling, it is crucial to choose a MC step that minimizes the autocorrelation time. Here we will perform a systematic study using many different values in order to understand the trend of τ_A as a function of the the `mcstep`. We will also see how the optimal correlation time compares with the acceptance ratio $\alpha = \frac{\text{accepeted moves}}{\text{total moves}}$.

- Perform a series of separate runs changing the value of the `mcstep` parameter, and compute for each of the runs the autocorrelation time. Start with the examples provided. Make sure that the simulation is long enough to converge the autocorrelation function, and that the maximum lag is not much longer than the autocorrelation time itself. Also, keep track of the acceptance ratio, which is printed out at the end of each run as a comment in the log file:

```
$ tail -n 6 log-0.05
##
## Total moves:        50000
## Accepteed moves:        11277
##
## Ratio:   0.22553999999999999
##
```

- Write down in a text file the autocorrelation time and the acceptance ratio as a function of the `mcstep`. You can then use `gnuplot` to generate a plot and store it in an image that you can use when you write your report. To export graphs produced by `gnuplot`, you can just specify a terminal, which also determines the format of output file. `gnuplot` supports terminals for various formats including png, jpg, gif and postscript. For instance,

```
gnuplot> set terminal png size 400,300 enhanced font "Sans,20"
gnuplot> set output "mcstep.png"
```

  In this example we used the png output format, but you can either use jpg, ps and pdf. The other options used – size, enhanced, font – let you to control the appearance of the exported plot.
- Which value of `mcstep` corresponds to the shortest autocorrelation time? What is the corresponding α?

1.3 Exploiting local moves

[OPTIONAL] The program is implemented using the detail balance Metropolis condition as transition rule :

$$\eta < e^{-\beta(V^{\text{new}} - V^{\text{new}})}$$

where η is a number between zero and one sampled from a uniform distribution, while $V^{\text{new}} - V^{\text{new}}$is the energy difference of the entire system after and before the move. Since we are using a simple pairwise LJ potential and since we move just one particle per move, this is a wasteful way to proceed: the potential is evaluated as a sum of pair terms

$$V\left(\{\mathbf{r}_1, \ldots, \mathbf{r}_N\}\right) = \frac{1}{2}\sum_{ij} v\left(|\mathbf{r}_i - \mathbf{r}_j|\right),$$

and so moving just one particle leaves most of the pair-wise terms untouched. In fact, one can show that if the k-th particle has been moved, the difference between the potential

before and after the move is just

$$V^{\rm new} - V^{\rm old} = \sum_i v\left(|\mathbf{r}_i - \mathbf{r}_k^{\rm new}|\right) - \sum_i v\left(\left|\mathbf{r}_i - \mathbf{r}_k^{\rm old}\right|\right).$$

You can try to modify the code to use this trick to reduce the cost of performing each MC step, and see how the wall-clock execution time changes.

2 Molecular dynamics.

This exercise focuses on using Molecular Dynamics (MD) to sample the constant temperature canonical ensemble. Enclosed is a simple MD program that will give you the possibility to run simulations in the NVE and NVT ensemble for a Lennard-Jones cluster. In order to couple the system with a thermal bath, three thermostat are implemented: Andersen, white-noise Langevin (WNL) and colored-noise Generalized Langevin Equation (GLE). All the source code files are provided in the `src/` directory, that contains

`mdcode.f90`	The main file, containing the initialization of the system and the MD loop
`routines.f90`	The module containing the principal subroutines used in the `mdcode.f90` file, such as the calculation of energies and forces, i/o routines and basic thermostats
`glecn.f90`	The module containing routines for the colored-noise Langevin thermostat
`random.f90`	Routines to sample a random number from a Gaussian distribution with zero mean.
`Makefile`	The file containing the compilation parameters

To compile the program just type `make` as already seen for the MC code. This will generate a binary file called `mdcode`. The input file is as follows:

`&inp`	
`seed = 1357`	Initial seed for the random number generator
`temp = 0.20`	Target temperature
`dataxyz = 'lj38.xyz'`	Starting configuration
`dt = 0.001`	Integration time step
`nstep = 20000`	Number of steps to be performed
`langevinWNtau = 10`	Relaxation time of the white-noise Langevin thermostat. If set to 0, or if the keyword is not present, this thermostat will not be applied
`gleafile = 'gle-A.dat'`	The file containing the drift matrix A required by the colored-noise Langevin thermostat. If set to an empty string or if the keyword is not present this thermostat will not be applied
`mstep = 1`	Apply the Langevin thermostats in a multiple time step fashion
`andersentau = 10`	Relaxation time of the Andersen thermostat If set to 0, or if the keyword is not present, this thermostat will not be applied
`stridetrj = 50000`	Stride outputting the trajectory
`stridelog = 1`	Stride outputting the simulation logs
`outputf = 'out.xyz'`	Output trajectory file
`&end`	

Note that using `langevinWNtau = 0, gleafile = "` and `andersentau = 0` one can perform constant-energy microcanonical dynamics in the NVE ensemble.

2.1 NVE dynamics and energy conservation.

In MD, Hamiltonian systems of differential equations are numerically integrated to evolve the system in time following the prescriptions of classical, Newtonian dynamics. The integrator implemented in `mdcode` is the simple (but effective) velocity-Verlet algorithm. Such a method does not conserve energy exactly along the trajectory, thus leading to a discretized trajectory that diverges from the one that would be obtained by exact integration of the dynamics. However, although it does not conserve the energy, when using a sufficiently small time step, the velocity-Verlet integrator maintains the system's total energy in a narrow band around the true energy and yields remarkably stable trajectories with essentially no significant energy drift.

A word of caution: while one can achieve near-perfect energy conservation by reducing the time step, working with an excessively small time step may result in waste of computer time. A practical compromise would allow for small energy fluctuations and slow energy drifts, as a price to pay to work with a reasonably large `dt`. Concerns about the conservation of total energy are less serious when using a thermostat, as it will be discussed below.

As a first example of molecular dynamics, we will run NVE simulations starting from a configuration frozen in the *fcc* ground state structure, and try to melt it as we did with Monte Carlo. Use the example file called `input` in the directory `pt1/` .

To run the program:

```
$ ../src/mdcode input > log &
```

Before starting the exercise take a brief look at the `log` file:

```
$ head log
## MD code.
## Natoms:           38
## Timestep:   1.0000000000000002E-002
## Temperature:   0.23000000000000001
##
## Step, Temp, Ekin, Epot, Etot, Conserved quantity
 0  0.2397 0.1330 -173.9225 -160.6201 -160.6201
 10 0.8079 4.4842 -165.0655 -160.5812 -160.5812
 20 0.1119 6.2140 -166.8071 -160.5931 -160.5931
```

- Use `gnuplot` to monitor the potential, kinetic and total energies:

  ```
  gnuplot> p 'log' u 1:3 w l
  gnuplot> p 'log' u 1:4 w l, 'log' u 1:5 w l
  ```

- Try running with increasing values of the time step. What happens? Visualize the trajectory using `vmd`.
  - What is the highest usable timestep?
  - How do energy fluctuations and drift vary with `dt`?

Now, choose the best timestep you found and run a simulation at T^*=0.20 lasting for 3 million steps. Look at the trajectory using `vmd`. Plot the potential energy and compare the results with those from MC simulation at the same target temperature, starting from the same configuration.

- What can you say about the equilibration and energy trend comparing the MD plot with the MC one?
- Compute the average temperature and the average configurational energy.

```
$ tail -n +5000 log | awk '!/#/{t+=$2;pot+=$4; n++}
 END{print"T=",t/n;print"Epot=",pot/n}'
```

  What is the value of $\langle T^* \rangle$? And $\langle V \rangle$? Are they comparable with the chosen target temperature and with $\langle V \rangle$ obtained from the MC simulation?

2.2 Constant temperature MD

NVE molecular dynamics does not yield sampling consistent with the target temperature – particularly if the simulation is initialized from a configuration which is not compatible with the ensemble. Three thermostat are implemented in `mdcode` to perform ergodic trajectories within the NVT ensemble. Go into the directory `pt2` and run a first simulation using the provided example file (do not change the target temperature).
You should also run the GLE simulation from (2.4) since the simulation should last for an hour.

- Plot the total energy (or the configurational energy) and compare it with the one from the NVE run. What happens switching on the thermostat? Plot also the conserved quantity (column 6). Is it (approximately) conserved? See that its fluctuations decrease if you reduce the time step further.
- Run a new simulation using as starting configuration the final snapshot from the previous trajectory:

```
$ tail -n 40 out.xyz > ljnew.xyz &
```

  Remember to change the input file setting `dataxyz = 'ljnew.xyz'`.

```
$ sed "s/lj-oct-ok/ljnew/" input > input.new &
```

  - Is the equilibration transient changed? If yes, why?

- Now try to increase a bit `dt` (let's say `dt=0.01`) and, again, plot the conserved quantity and the total energy together.
  - What happens to the conserved quantity increasing `dt`?
  - Does the mean total energy change significantly (use `autocorr` to compute a mean value with errorbars)?
- Compute the autocorrelation function for the potential energy and plot it. Can you see the multiple time scales at play here? Compute the autocorrelation time and compare it to the efficiency of Monte Carlo in terms of energy evaluations needed to obtain an uncorrelated sample. Is this a fair comparison?
- [OPTIONAL] Test the Andersen thermostat and compare the resulting total energy with the one using the WNL from the previous simulation. What the difference between the two? How does the Andersen's one look like?
  - What happens choosing a big relaxation time for the Andersen thermostat? And a really small one?

2.3 Assessing the sampling efficiency of the WNL thermostat

The sampling efficiency of a thermostat can be assessed quantitatively evaluating the autocorrelation times relative to the observables we are interested in. Since the behavior of the WNL thermostat is controlled tuning its relaxation time (`langevinWNtau`), we will now perform a systematic study in order to discover the relaxation time that minimizes the (configurational) energy autocorrelation time (τ_V).

- Go into the directory `pt3` and run different simulations at $T^* = 0.20$, changing the value of the `langevinWNtau` (0.01,0.1,1,10,100).
  N.B.: Start using configuration extracted from a previously equilibrated run at $T^* = 0.20$. Use the following command:

  ```
  $ tail -n 40 out.xyz > lj.xyz &
  ```

  Make sure that the number of steps that you will choose for the simulations is long enough to converge the autocorrelation functions (the a.c. function should converge to zero and should not be too much noisy); use the given input file as a prototype for your runs. Since you have to run several simulations and to analyze them, you would probably want your simulation lasting not too much. Also avoid to output the logs every steps, otherwise you will rapidly fill up all your disk free space. An example of the parameters you should use in this example is: `nstep=20000000` and `stridelog=50`.
- Compute for each run the autocorrelation time using `autocorr`. Plot the a.c. functions in a comparative graph and save it to a file (that will be part of your final report). Be extremely careful here: as already said, you must choose a `maxlag` value assuring you that the a.c. functions will converge to zero, to get reliable results. You might need to choose different values depending on the simulation parameters
- Plot the trend of τ_V (as already seen in the MC part, you can find the a.c. time in the `autocorr` output header!) as a function of the WNL relaxation time and save it in a file for your report. Use the logarithmic scale to better see the trend of the variation.
- What is the optimal relaxation time for the WNL thermostat at the chosen condition?
- [OPTIONAL] What do you think will happen when changing the target temperature? Will the `langevinWNtau` you found still be the best at any T^*? Compare with correlation times found at another temperature ($T^\star = 0.23$ for example).

2.4 Colored-noise Langevin thermostat

You should now know that finding the best parameters for the WNL thermostat is definitely not trivial. Multiple time scales are at play, and applying a different WNL thermostat to different molecular coordinates would require in-depth knowledge of the dynamics of the system, which is often impractical and computationally demanding. A good solution is to use the so called colored-noise, Generalized Langevin Equation (GLE) thermostat. This thermostat is based on a stochastic technique that automatically adapts and enforces ergodic sampling on all the normal modes of the system. In this way the GLE thermostat assure an "optimal sampling" avoiding you the effort to find the best relaxation time for the particular system you're studying at. Keep in mind that for this particular set-up, with simple L.J. potential, the numerical cost of the GLE thermostat is not negligible and the simulation should run for about an hour.

- Go into the directory `pt4` and run a simulation using the example file.

- Compute the energy autocorrelation functions and compare it with the ones you obtained using the WNL thermostat.
- What about the GLE thermostat a.c. time compared with conventional Langevin? Consider that you did not need to optimize any parameter here!

3 Free energy and transition state theory

The program is a modified version of the simple MD code of the previous exercise. It only implements white-noise Langevin (WNL), but contains additional routines to evaluate collective variables, and a separate program to perform a committor analysis

`mdcode.f90`	The main file, containing the initialization of the system and the MD loop
`mdcommitt.f90`	Computes the committor for a number of atomic configurations, given in input
`routines.f90`	The module containing the principal subroutines used in the `mdcode.f90` file, such as the calculation of energies and forces, i/o routines and basic thermostats
`structure.f90`	The module with the routines to evaluate coordination numbers
`random.f90`	Routines to sample a random number from a Gaussian distribution with zero mean.
`Makefile`	The file containing the compilation parameters

Both programs can be compiled in the usual way. This will generate two files: `mdcode`, that you will use to run constant-temperature MD trajectories, and `mdcommitt`, that you will use for the subsequent analysis. The modified MD program also outputs two collective variables that make it possible to distinguish different structures:

$$n_c = \sum_{i=1}^{N} e^{-\frac{(c_i - c)^2}{2\sigma^2}},$$

where c_i is the coordination number of atom i, defined in turn as

$$c_i = \sum_j \mathcal{C}\left(|\mathbf{q}_i - \mathbf{q}_j|\right), \qquad \mathcal{C}(d) = \begin{cases} 0 & d > r_0 \\ 1 & d < r_1 \\ (y-1)^2 (2y+1) & r_1 < d < r_0, \quad y = \frac{d - r_1}{r_0 - r_1} \end{cases}.$$

The variable n_c corresponds approximately to the number of atoms in the structure with a coordination number around the value c.

Note that each run in exercise 3.2 and 3.3 should last about 30 minutes so you might want to launch most of them in advance. The input file for the `mdcode` program is as follows:

`&inp`	
`seed = 1357`	Initial seed for the random number generator
`temp = 0.168`	Target temperature
`dataxyz = 'lj38.xyz'`	Starting configuration
`dt = 0.01`	Integration time step
`nstep = 100000000`	Number of steps to be performed
`langevinWNtau = 10`	Relaxation time of the white-noise Langevin thermostat. If set to 0, or if the keyword is not present, this thermostat will not be applied
`mstep = 10`	Apply the Langevin thermostats in a multiple time step fashion
`stridetrj = 20000`	Stride outputting the trajectory
`stridelog = 2000`	Stride outputting the simulation logs
`r0=1.5, r1=1.25, sig=0.5`	Parameters for the evaluation of CVs (r_0, r_1, σ)
`stridecv = 200`	Stride outputting the collective variables
`outputf = 'out.xyz'`	Output trajectory file
`outputcv = 'out.cv'`	File to output CV information
`clist(1)=6, clist(2)=8`	Chooses two coordination numbers a and b for which to compute n_c.
`sela(1)=18, sela(2)=25`	Intervals for selecting structures with sela(1) $< n_a <$ sela(2). Does not perform a check if sela(1)<0
`selb(1)=-1, selb(2)=-1`	Intervals for selecting structures with selb(1) $< n_b <$ selb(2). Does not perform a check if selb(1)<0
`seloutf = 'sel.xyz'`	File to output selected configurations
`&end`	

3.1 Computing free energies

In the first part of this exercise we will run a constant-temperature simulation, compute free energies relative to different collective coordinates and verify that they can distinguish between different meta-stable structures. Example input files are given in `pt1/`, and can be run with

```
$ ../src/mdcode input > log &
```

Besides the usual diagnostics and a series of snapshots collected along the trajectory, the program will also print out a `outputcv` file containing the values of n_6, n_8 and the values of the one-dimensional flux factors $\phi_8(\mathbf{q})$ and $\phi_6(\mathbf{q})$ computed for the corresponding configurations. It will also print out a file with configurations selected according to their value of (n_6, n_8).

One can then compute the free energy associated with the two collective variable separately, by computing the histograms of $h(n_8)$ and $h(n_6)$. The corresponding free energy is then just $F(n_c) = -\frac{1}{\beta} \ln h(n_c)$. A script is provided that computes $F(n_8)$ and $F(n_6)$, as well as the free energy relative to the *joint* distribution of n_6 and n_8 and the average flux factors

$$\phi(n_c) = \frac{\langle \delta(n_c(\mathbf{q}) - n_c)\, \phi_c(\mathbf{q}) \rangle}{\langle \phi_c(\mathbf{q}) \rangle}.$$

A script is provided that computes all of these quantities using `(nd)histogram` at once (assuming that the file names have not been changed!):

```
$ ../src/mkfes.sh 0.168 &
```

- Run the simulation, monitoring the value of the collective variables with gnuplot.
- Compute the free energies (make sure to understand what the script is doing!)
- Look at the free energy for n_6 and see that there is a meta-stable state for $n_6 \gtrsim 18$. By inspecting the selected atomic configurations using `vmd`, see what structure this corresponds to. Note that there are also some configurations that look quite different. This is a sign that n_6 is not sufficient to identify the *fcc* cluster.
- You might want to have a look at $F(n_6, n_8)$ that clarifies what is going wrong. However, we will keep using just n_6 for a while...

```
gnuplot> set pm3d map; sp 'n6n8.fes' w pm3d
```

- Estimate the transition state theory rate by combining the free energy information with the average flux factor at $n_6 = 18$

3.2 Committor analysis

To assess the quality of the collective variable n_6 in studying the *fcc*-liquid transition, let us try to single out the transition state ensemble and to perform a committor analysis. To this aim, we will use the `mdcommitt` code, that is just a simplified version of the MD code with the following options

`&inp`	
`seed = 1357`	Initial seed for the random number generator
`temp = 0.168`	Target temperature
`dataxyz = 'ts.xyz'`	Sequence of configurations for which the committor is to be computed
`dt = 0.01`	Integration time step
`nstep = 250`	Number of steps to be performed for each trajectory
`nshoot = 100`	Number of trajectory that must be generated from each configuration
`r0=1.5, r1=1.25, sig=0.5`	Parameters for the evaluation of CVs (r_0, r_1, σ)
`stridecv = 200`	Stride outputting the collective variables
`cvoutf = 'out.finalcv'`	Print out the final CVs for each trajectory
`clist(1)=6, clist(2)=8`	Chooses two coordination numbers a and b for which to compute n_c.
`sela(1)=0, sela(2)=1.8`	Interval defining the product region (sela(1) $<n_6<$ sela(2)). Does not perform a check if sela(1)<0
`selb(1)=-1, selb(2)=-1`	Interval defining the product region (selb(1) $<n_8<$ selb(2)). Does not perform a check if selb(1)<0
`&end`	

The committor is computed by running multiple trajectories from each configuration, and counting how many times the trajectory ends up in the defined product region. The code will print out for each configuration the initial value of the two CVs and the value of the committor. Example input files are given in `pt2/`.

- Start a (long) trajectory using the `input.md` file. The `sela` parameters should pick a region in the vicinity of the maximum of the free energy relative to n_6. This should be something around $n_6 = 18$.

```
$ ../src/mdcode input.md > log.md &
```

- The simulation will print out configurations that fulfill these constraints, into the `ts.xyz` file. Look at the values of the collective variables along the trajectory, making sure that there actually are transitions between the *fcc* minimum and the molten states of the cluster. Inspect the file with `vmd`.
- When enough snapshots have been accumulated (it might take a long time before the MD simulation is over, so you may want to start playing around before the run is finished. Make sure not to overwrite files while the MD is still running!) you can run the actual committor analysis.

```
$ ../src/mdcommit input.comm > log.comm &
```

  It will take some time before there is any output in the file – this is because the output is buffered, and so lines will appear only when the buffer is flushed.
- Look at the position of the initial configurations and the value of the committor. This can be done with gnuplot using

```
gnuplot> p 'log.comm' u 5:7:2 w p pt 7 lt pal
```

- Compute the histogram of committors for the transition state ensemble

```
$ awk '!/#/{print $2}' log.comm
| histogram -xi 0 -xf 1 -n 100 -t 0.1 -whard > histo.comm
```

  Is this a good collective variable?

3.3 Committor analysis in 2D

Look carefully at the 2D free energy surface computed on one of the MD runs. It should be clear why n_6 is not too bad to distinguish reactants and products, but is a terrible CV to describe the transition state. You can try to repeat the committor analysis using both n_6 and n_8 to define the transition state region. Example input files are given in `pt3/`.

- Run the MD trajectory outputting the configurations that conform to the 2D definition of the transition state ensemble (tentative ranges for n_6 and n_8 are already given, but feel free to experiment).
- Run the committor analysis on this new set of transition states. It is somewhat better than before, but not much better. LJ_{38} is much harder than it looks like!

4 Reweighing and biased sampling

Statistical reweighing promises to extract much more information from a single MD run, by gathering thermodynamic averages at different temperatures, or by correcting the distortion of the ensemble statistics which is due to the use of a biasing potential. To do so, one has to weight observables by the ratio of the target probability distribution and the distribution that was sampled during the simulation. The code for this exercise has been modified to output the energy and bias together with the values of collective variables, i.e. the file specified in `outputcv` contains

```
Step.Index   Na    PhiA   Nb    PhiB     V      Bias
```

Note that Metadynamics (4.2) and Well-tempered metadynamics (4.3) would need one hour to run so you might want to launch them in advance.

4.1 Temperature reweighing

Perform two runs at $T^\star = 0.17$ and $T^\star = 0.19$.

```
$ ../src/mdcode input.17 > log.17 &
$ ../src/mdcode input.19 > log.19 &
```

- Compare runs at different temperatures. Use re-weighting to recover evaluate the mean potential energy at temperatures that differ from those of the actual run. You can use the script given in the `src/` directory, but make sure to understand what it is doing.

  ```
  $ ../src/mkrwavg.sh 0.17 < log.17 > vavg.17
  $ ../src/mkrwavg.sh 0.19 < log.19 > vavg.19
  $ ../src/mkrwavg.sh 0.17 0.19 < log.17 > vavg.17to19
  $ ../src/mkrwavg.sh 0.19 0.17 < log.19 > vavg.19to17
  ```

- Note that the converged result from reweighing the run at $T^\star = 0.17$ is actually within statistical error of the average computed at $T^\star = 0.19$, and vice versa. Try to reweigh from $T^\star = 0.17$ to $T^\star = 0.23$, and from $T^\star = 0.19$ to $T^\star = 0.23$. Note that the final averages differ a lot, and that the cumulative average for $0.17 \to 0.23$ is much more irregular up to large times. This is a manifestation of the pathological statistics of reweighed sampling. Try to compute the fluctuations of the log-weight as a function of the target temperature, and estimate the range on which reweighing can be expected to work decently.
- Use another script to compute reweighed free energies

  ```
  $ ../src/mkrwfes.sh 0.17 < out.17.cv > n6.17
  $ ../src/mkrwfes.sh 0.19 < out.19.cv > n6.19
  $ ../src/mkrwfes.sh 0.17 0.19 < out.17.cv > n6.17to19
  $ ../src/mkrwfes.sh 0.19 0.17 < out.19.cv > n6.19to17
  ```

- Try to experiment with different temperatures. Look at the time evolution of n_6 and see how often transitions happen between the *fcc* and the liquid basin. If the objective was to sample thoroughly the transition state region, do you think that temperature reweighing could help?

4.2 Metadynamics

The code provided for these exercises also implement metadynamics using the two selected collective variables. In addition to the options of previous days, a few options that are specific to metadynamics are also included:

`wmeta = 0.1`	Height of each deposited hill
`sigmameta = 0.25`	Width of the repulsive hills
`stridemeta = 10000`	Stride for depositing new repulsive hills
`outhillsf = 'out.hills'`	Filename for saving the centers of repulsive hills deposited during the simulation

- Perform a metadynamics run, using the example presented in `pt2/`. Visualize the position of the hills and the bias as the simulations goes on.

  ```
  $ ../src/mdcode input.meta > log.meta &
  ...
  gnuplot> p 'out.hills' u 2:3:6 w p pt 7 ps 2 lt pal
  gnuplot> p 'log.meta' u 1:5 w l
  ```

- As the simulation proceeds, transitions between the *fcc* cluster and liquid-like clusters will become more and more likely.
- You can follow the build-up of the bias by using the `sumhills` utility to post-process the `outhillsf` file. The `-t nh` option allows you to plot the bias including only the first `nh` hills.

```
$ awk '{print $2,$3,$4,$5}' out.hills | head -n 1000 |
../src/sumhills   --nci 0,0 --ncf 25,16  --b 0.1,0.1 > bias.splot
...
gnuplot> set pm3d map; sp 'bias.splot' w pm3d
```

  - Note that at any instant in time there is a residual roughness on the FES as estimated from the negative of the bias, and that the width of the hills effectively set a limit to the resolution at which the free energy can be reconstructed
  - Try to run with a reduced hill height, and more frequent deposition. Is the runtime getting longer? This is actually the case – for such a simple system the evaluation of the repulsive bias can be as expensive as the physical forces. The evaluation of the bias gets slower as the simulation progresses, since the sum extends over all the configurations visited in the past. Efficient implementations of metadynamics typically represent the bias on a grid, which means that the cost does not vary with simulation time.

- Since metadynamics relies on an assumption of local equilibrium, one can also obtain the free energy or any observable by reweighing the trajectory, to subtract the effect of the bias. Since the bias grows continuously, however, only the last part of the trajectory contributes significantly to the averages, which means that convergence of averages is exceedingly slow. See for instance the trend of the cumulative average of the potential. In fact, extracting Boltzmann weights from a metadynamics trajectory can be made more efficient by "equalizing" the weight of different parts of the trajectory – an advanced technique that we will not discuss here.

```
$ ../src/mkrwavg.sh 0.168 < log.meta > vavg.rw
$ ../src/mkrwfes.sh 0.168 < out.cv > n6.rwfes
```

  - Compare the free energy with that computed from an unbiased run

4.3 Well-tempered metadynamics

The problem of residual roughness in the bias generated by metadynamics can be addressed by reducing systematically the height of the repulsive hills. The "well-tempered" metadynamics strategy does so by scaling the deposition rate by an exponential function of the already-deposited bias $e^{-B(\mathbf{s}(t),t)/\Delta T}$. This behavior can be specified in our code by specifying a non-zero ΔT using the option

`wttemp = 0.5`	Effective temperature used in the well-tempered damping of bias deposition rate

- Run a simulation using well-tempered metadynamics. Compare the growth of the bias with conventional metadynamics and with damped deposition rate.
- Compare the rate of transitions with the two approaches. Well-tempered metadynamics is somewhat less effective. This comes because

  1. the well-tempered bias does not flatten the free energy completely.
  2. the continued accumulation of bias in conventional metadynamics pushes the system around, accelerating diffusion even on a flat free energy surface. This is

however dangerous, as it leads to hysteresis effects and might systematically bias the estimation of the FES.

- Compare the final bias using `sumhills`, with that computed on the conventional metadynamics run.
  - Remember that the free energy in well-tempered metadynamics is

$$F(\mathbf{s}) = -B(\mathbf{s})\frac{T+\Delta T}{\Delta T} + C.$$

    You can apply this scaling directly from `gnuplot`, e.g.

```
gnuplot> sp 'bias.splot' u 1:2:(-$3*(1+0.168/0.5)) w pm3d
```

- Compute reweighted quantities. See how cumulative averages converge better than with conventional metadynamics, since the bias is now smoothly converging. Note that a more sophisticated analysis that balances the different parts of the trajectory would still be very beneficial.

5 Dimensionality reduction

In this exercise we will focus on post-processing the results of previous simulations to determine machine-learning collective variables (CV) that can give you a better understanding of the configuration space of LJ_{38}, and of the shortcomings of conventional CVs even in such a simple system.

We will use the sketch-map method [3], and the suite of programs that can be downloaded from `http://cosmo-epfl.github.io/sketchmap`.

We will first extract a set $\{X_i\}$ of reference configurations, described in terms of the set of all coordination counts between 4 and 13, and find their sketch-map projection by minimizing the objective function

$$\chi^2 = \sum_{i,j=1}^{N} \left[s\left(|X_i - X_j|\right) - s\left(|x_i - x_j|\right) \right]^2 . \tag{5.1}$$

Having obtained this map, one can more easily find the out-of-sample embedding of the remaining points, and analyze the simulation based on the map obtained by machine-learning.

Note that a `commands` text file is provided to avoid any misspelling in the following command lines but keep in mind that it's better to understand what they mean.

5.1 Extracting landmark points

Since the cost of minimizing iteratively Eq. (5.1) grows quickly with the number of reference points, so it is important to select a sub-set of the data from which one wants to build the map. Get the `out.all` data file from the first of the exercises on transition state theory, or run `mdcode` to generate it

```
$ ../../ex3/src/mdcode input.cv > log
```

Then, use `dimlandmark` to extract 500 reference points from the dataset:

```
$ awk '{if (NR%10==0) print $0}' out.all | dimlandmark -D 10 -n 500
   -w -unique -mode staged -gamma 0.5 -wgamma 0.5 > lj38.lm
```

Read the help string of the program (`dimlandmark -h`) to get a description of the options. Note that we pre-select a subset of the data set to speed up the evaluation. The *staged* mode of selecting the landmarks is a two-step procedure in which a larger set of points is chosen, approximately uniformly spaced one from another. Then, the probability density in the D dimensional space is estimated, and it is used to select the n landmark points according to $P(X)^{\gamma}$. You can try to visualize the selected landmarks with `gnuplot`, superimposing them to a free-energy surface for n_6 and n_8, to see how they change when the value of γ is modified

```
gnuplot> sp 'n6n8.fes' w pm3d, 'lj38.lm' u 3:5:(3) w p
```

5.2 Analyzing the distribution of data points

Before starting this section you might want to run sketch-map (5.3) and the out-of-sample embedding (5.4) because `dimproj` utility might need one hour and a half to run.
Sketch-map is based on restricting the similarity matching that underlies MDS methods to the range of distances that characterize adjacent meta-stable states. Compute the distribution of individual CVs, e.g. by

```
$ awk '{print $3}' out.all | histogram -xi 0 -xf 25 -n 500 -t 0.1 > n6
```

and look at the amplitude of fluctuations around maxima of the histogram in each dimension. Note that there are multiple scales in the fluctuations in probability density – sharp peaks with a width of about 3 units, superimposed with broad features with much larger breadth of about 5 units. This is a common problem when analyzing atomistic data in a glassy free-energy landscape – multiple shallow minima are grouped together to form extended regions of similar, closely-related structures. Also, consider that fluctuations in D dimensions are approximately $\sqrt{D}$ times broader than their one-dimensional projections, so depending on the resolution one wants to achieve one might want to select a threshold parameter for sketch-map between 1.5 and 10 units.

You may also want to compute the histogram of D-dimensional distances. Typically one wants to select a cutoff somewhere before the maximum of the distribution, which is dominated by long-range features

```
$ awk '{if (NR%10==0) print $0}' out.all > tmp
$ dimdist -D 10 -P tmp -maxd 20 -nbin 200 -lowmem > lj38.histo
```

Sketch map uses a sigmoid function defined as

$$s(r) = 1 - \left(1 + \left(2^{a/b} - 1\right)(r/\sigma)^{a}\right)^{-b/a}.$$

You may want to plot it superimposed with the histogram of pairwise distances to get a feeling of how different sets of distances are transformed by the function:

```
gnuplot> f(r,s,a,b)=1-(1.0+(2**(a*1.0/b)-1)*(r/s)**a)**-(b*1.0/a)
gnuplot> p 'lj38.histo' u 1:($2*10) w l, f(x,5,8,1)
```

5.3 Run sketch-map

Being an iterative minimization scheme, sketch-map requires a decent starting configuration and an optimizer that can get out of local minima to approach a global optimum. Without getting into details, you can use a simple script that takes care of the optimization procedure. **Remember** to delete the two-lines header of `lj38.lm` before proceeding.

```
$ sketch-map.sh
Please enter the dimensionality of input data 10
Do we have a similarity matrix ? n
Are points weighted [y/n]? y
Should I use the dot-product ? n
Please enter the periodicity of input data [0 if non-periodic] 0
Please enter the input data file name lj38.lm
Please enter the output data prefix lj38.5_8_1-5_2_2
Please enter high dimension sigma, a, b [e.g. 6.0 2 6 ] 5 8 1
Please enter low  dimension sigma, a, b [e.g. 6.0 2 6 ] 5 2 2
```

You can then remove intermediate files and diagnostics, and assemble a file with just the low-dimensional coordinates of the landmarks.

```
$ rm log global.* lj38.5_8_1-5_2_2.*[0-9]
$ awk '!/#/{print $1, $2}' lj38.5_8_1-5_2_2.gmds > lj38.ld
```

- Visualize the projected points, and verify by coloring how the sketch-map coordinates separate clearly points with different n_ks

  ```
  gnuplot> p '< paste lj38.ld lj38.lm' u 1:2:5 w p pt 7 lt pal
  ```

- [OPTIONAL] Try to run projections with different parameters – σ, a, b – and verify how much the projection changes

5.4 Out-of-sample embedding

Having obtained a map that assigns to each of the high-dimensional landmark points a corresponding embedding, one can proceed to project all the data in the original trajectory. You should use the utility **dimproj** todo so, specifying the high and low-dimensional references, the sketch-map parameters and giving in input the full trajectory:

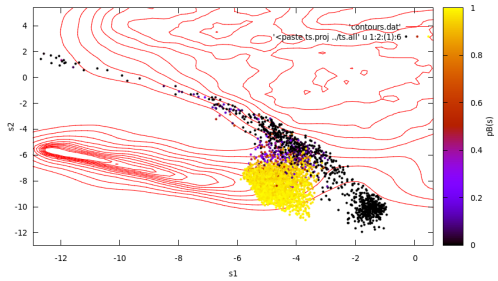
```
$ dimproj -D 10 -d 2 -P lj38.lm  -p lj38.ld -w -grid 15,16,151
   -cgmin 3 -fun-hd 5,8,1 -fun-ld 5,2,2 < out.all > out.proj 2> /dev/null
```

Make sure to discard the error log: this is development code and outputs a lot of junk that you don't need to worry about at this stage.

- Compute the free energy from the sketch-map projection. A (long) one-liner to do this is as follows

```
$ awk '{print $1, $2}' out.proj | ndhistogram -d 2 -g -xi -15,-15
   -xf 15,15 -n 150,150 -t 0.2,0.2 -adaptive 0.25 |
   awk -v kt=0.168 'BEGIN{print "# s1  s2  F(s1,s2)" }
   !/#/{ if (NF==0) print ""; else printf "%15.7e %15.7e %15.7e\n",
   $1, $2, -kt*log($3) } ' >  smap.fes
```

Figure 3: Sketch-map projection of the transition-state configurations obtained in 3.3, overlaid on the free energy surface and colored according to the value of the committor.



5.5 Analyzing the transition state ensemble

We can then see how sketch-map coordinates work to describe high-energy states and the transition state ensemble between the *fcc* and liquid-like states. You can find a file with the configurations saved from the exercises in 3.3, or you can use one from your own runs. You should first project the data points

```
$ awk '{for (i=3; i<=NF; i++) printf "%s ", $i; print "" }' ts.all |
   dimproj –D 10 –d 2 –w –P lj38.lm –p lj38.ld –grid 15,16,151
   –cgmin 3 –fun–hd 5,8,1 –fun–ld 5,2,2 > ts.proj 2> /dev/null
```

And then do some gnuplot magic to overlay the sketch-map free energy and the projection of the transition state data, colored according to the committor.

```
gnuplot> set table 'contours.dat';
gnuplot> set view map; set contour; unset surface;
gnuplot> set cntrparam levels incremental 0.5,0.1,1.5;
gnuplot> sp 'smap.fes' w l; unset table; res; set view map
gnuplot> sp 'contours.dat' w l, '<paste ts.proj ts.all' u 1:2:(1):6 w p lt pal pt 7 ps 0.5
```

The result should look similar to Fig. 3. Note that the value of the committor is different from zero or one in the region that is a saddle point on the sketch-map free energy – although a large portion of it is not sampled because the region we had selected based on just n_6 and n_8 did not describe properly the transition state ensemble.

5.6 Transferability of sketch-map

[OPTIONAL] This far you have used a thorough sampling of the accessible configuration space to build the initial map, and then projected additional data samples onto this map. In most cases, however, one does not have the luxury of complete sampling and so it is interesting to see how sketch-map can "fill in the blanks" and give reasonable description of the unexplored parts of the configurations space.

You can repeat the landmark selection and sketch-map construction using as input only the data samples from the transition state ensemble, and then project the whole trajectory from `out.all` onto this partial map. Can you still recognize the most important features of the energy landscape?

Bibliography

[1] David Wales. *Energy landscapes: Applications to clusters, biomolecules and glasses*. Cambridge University Press, 2003.

[2] Jonathan P. K. Doye, Mark A. Miller, and David J. Wales. The double-funnel energy landscape of the 38-atom lennard-jones cluster. *The Journal of Chemical Physics*, 110(14):6896, 1999.

[3] Michele Ceriotti, Gareth A. Tribello, and Michele Parrinello. Demonstrating the transferability and the descriptive power of sketch-map. *Journal of Chemical Theory and Computation*, 9(3):1521–1532, March 2013.