Implementing lattice-switch Monte Carlo in DL-MONTE to unlock efficient free energy calculations

Kevin Stratford and Tom Underwood

Version 1.0, February 28, 2017

Abstract

Lattice-switch Monte Carlo (LSMC) is an efficient method for calculating free energy differences between crystalline phases. Here, LSMC is implemented in the general-purpose Monte Carlo simulation package DL-MONTE. This has enabled LSMC to be applied to a range of systems which have never before been possible to study via LSMC, including systems comprised of atoms or rigid molecules. These may interact via a wide range of force fields, including long-range Coulomb interactions, and may be under conditions of constant density or constant pressure. LSMC renders tractable computation of free energy differences for systems with large energy barriers between lattice structures or phases which are difficult to address using alternative methods.

In addition, a python toolkit has been developed to control the workflow of DL-MONTE simulations, and to perform common pre-processing and post-processing tasks. This allows scripted workflows to be undertaken efficiently. In particular, the toolkit can be used to perform 'histogram reweighting', which allows simulation data obtained at one set of control parameters (e.g., temperature and pressure) to be used to provide information regarding other nearby control parameters. This can reduce the number of simulations required, for example, to identify the exact location of the co-existence points and curves that form phase diagrams. The toolkit can be generalised to other sources of simulation data and is not bound to DL-MONTE.

1. Introduction

The ability to predict the free energy difference between two specified phases by theory alone would be invaluable to many fields of science. One important application is calculating phase diagrams in conditions inaccessible to experiment: at a given temperature and pressure the stable phase is that with the lowest free energy. Unfortunately, for most systems of practical interest it is not currently possible to predict free energy differences to the accuracy required for practical applications, even with hardware such as ARCHER at our disposal. Fortunately, significant gains in accuracy can be achieved by implementing more efficient methods within existing software. Lattice-switch Monte Carlo (LSMC) (see [9] for a review) is a method which allows the free energy difference between two crystalline phases to be calculated efficiently. However, this method has not been adopted by the scientific community at large, despite evidence suggesting it is orders of magnitude more efficient than the alternatives. This report describes such an implementation, and a number of associated developments carried out as an ARCHER embedded CSE programme project.

2. Lattice switch Monte Carlo

This section provides a brief background to lattice-switch (sometimes referred to as phase-switch) Monte Carlo as implemented in DL-MONTE. The method was originally developed to look at the hcp-fcc problem in hard-sphere solids [2, 3] but can be adapted to many types of system.

2.1. Background

A given system may be free to visit two phases A and B (and only phases A and B), in which case the equilibrium phase is that with the lower free energy. (In the NVT ensemble this is the Helmholtz free energy, and in the NPT ensemble the Gibbs free energy.) To determine the equilibrium phase, one wants to compute the free energy difference $\Delta F = F_A - F_B$ between the phases. Standard statistical mechanics can be used to show that the free energy difference may be related to the ratio of the probability that a simulation is in a given phase p, i.e., $\Delta F = \beta^{-1} \ln(p_B/p_A)$, where β is the appropriate Boltzmann factor.

In a molecular dynamics simulation, one could monitor the time spent in the different phases and use this as a proxy for the probabilities. The problems start if one phase or other is inaccessible, meaning that phase is effectively never visited during any reasonable simulation. This is precisely the situation when a large free energy barrier exists between the phases — the region of the phase space associated with one phase is very improbable at thermodynamic equilibrium. This itself is a barrier to investigation of systems in which phase changes occur.

Monte Carlo can hold an advantage in this situation: the freedom to design

Monte Carlo moves means that if one can design an appropriate trial move which samples both phases with the correct statistics, one can hope to compute free energy differences efficiently. This is what lattice-switch Monte Carlo does. More specifically, if the positions of atoms or molecules $\mathbf{r}_i$ in the system can be written as

$$\mathbf{r}_i = \mathbf{R}_i^A + \mathbf{\Delta} r_i, \tag{1}$$

where $\mathbf{\Delta} r_i$ is the displacement of a particle from an underlying lattice position $\mathbf{R}_i^A$ for phase A, then a lattice switch involves a trial move in which $\mathbf{R}_i^A$ is replaced by $\mathbf{R}_i^B$. In this way, the free energy barrier between the phases is bypassed and sampling of both phases in a single simulation is tractable.

While in principle this sounds simple, the catch is that it may still be difficult to generate moves which are accepted with reasonable probability. To help with this problem transition matrix Monte Carlo [5, 8] has also been implemented in DL-MONTE, This improves the efficiency of the lattice switch process [9]. In this way lattice switch utilises multicanonical sampling — as opposed to canonical sampling which is used in conventional Monte Carlo — to push the system over the free energy barrier separating the two phases (see [9] for details). Transition matrix Monte Carlo is a general method and can also be used with methods other than lattice-switch available in DL-MONTE.

2.2. Benchmarks

To use LSMC in practice, the DL-MONTE user supplies two trial lattice configurations — for example hcp and fcp — which are of interest for the system in question, and requests the lattice switch method. The user also requests appropriate control parameters such as the initial state, and the frequency of switch attempts in Monte Carlo steps. The details of the sampling method used to obtain the free energy estimate is also specified (for example, whether to use transition matrix Monte Carlo).

Standard output at regular intervals then includes information related to the current phase of the system and any sampling bias related to transition matrix sampling, in addition to the current energy and other standard measures. This allows a true estimate of the free energy difference between the phases to be obtained. Such estimates have been obtained for a number of standard benchmark problems to compare DL-MONTE results with values reported in the literature.

Table 1 shows the free energy difference recorded by DL-MONTE simulations. The first two cases consider simple hard sphere systems in the NVT and NPT ensemble respectively. Two crystalline structures are of interest: hexagonal close packed (hcp) and face-centred cubic (fcc). An ensemble of 16 LSMC simulations each containing 216 hard sphere atoms and consisting of some 864 million Monte Carlo updates are performed in each case to gather statistics. For the NVT case, the fixed density $\rho/\rho^\star = 0.7778$ is used, where $\rho^\star$ is the density corresponding to close packing for hard spheres. The energy difference is recorded in units of $10^{-5} k_B T$ [1, 3, 7]. For the NPT case, the pressure p, hard sphere diameter d and Boltzmann factor are chosen so that $\beta p d^3 = 14.58$ to compare with previous studies [1, 3].

The third example in Table 1 is for particles interacting with a standard Lennard-Jones potential in the NPT ensemble. Again the two phases hcp and fcc are of interest. A system of 216 atoms at $p = 0$ and $k_B T/\epsilon = 0, 1$, where ϵ is the energy scale of the potential. The free energy difference between the phases is reported in units of ϵ.

The final example in Table 1 is a hard dumbbell example in the NVT ensemble (a hard dumbbell being a "molecule" of two spheres connected by a rigid bond of a given length). This is again an fcc vs hcp problem, where the crystal lattice involves centre-of-mass positions of the dumbbells (which can rotate freely around their centre of mass). A 864 molecule system is used with a reduced density of $\rho = 1.15$ (see e.g., [6] for details). The free energy difference is that per molecule in units of $k_B T$.

These standard examples are available to users as part of the DL-MONTE regression test suite. The LSMC implemented within DL_MONTE is not currently compatible with all functionality in DL_MONTE. In particular, LSMC is currently supported only for: orthorhombic phases; atomic systems and systems of rigid molecules: flexible molecules are not supported. Moreover, metallic systems are not supported. The extension of DL-MONTE to such systems will be the subject of future work.

System	ΔF [reference]	ΔF DL-MONTE
Hard spheres (NVT)	133(4) [3]	137(4)
Hard spheres (NPT)	123(6) [9]	135(6)
Lennard-Jones solid	0.001283(7) [9]	0.001290(11)
Hard dumbbells	0.005(1) [6]	0.0043(5)

Table 1: A comparison of free energy differences ΔF computed using DL-MONTE for a number of standard benchmark problems. The first is a hard sphere system (HCP vs FCC) at constant volume using 216 atoms compared with [3]. The second is again hard sphere (HCP vs FCC), but at constant pressure with 216 atoms at (compared with [9]). See the text for details of the configuration and units in each case.

3. Workflow and the python toolkit

A given piece of computational work often involves numerous simulations, altering parameters and iterative refinement of results via inspection and analysis. Tools which help with such workflow, in addition to the main simulation itself, are therefore of interest.

As it is not possible to anticipate all possible workflows in advance, a toolkit approach using a scripting language such as python is an attractive way to help users manage their particular workflow in a flexible manner. This section describes some of the building blocks developed by the python toolkit to aid analysis of DL-MONTE results.

While the toolkit has been developed with DL-MONTE in mind, functionality such as histogram reweighting is relevant to many kinds of simulation. The toolkit can be extended to analyse other sources of data by adding a “plugin” class to organise the data into a standard, structured, form.

3.1. Pre-processing

Scripting infrastructure to allow the manipulation of standard DL-MONTE inputs describing the simulation configuration have been developed. DL-MONTE uses three standard inputs (“FIELD”, “CONFIG”, and “CONTROL” files) which are read at run time and control the configuration of the system and the details of the various parameters to be used.

The python toolkit provides classes which represent these inputs in a structured form. This allows for convenient input, output, and manipulation of simulation control information. The python structured format is readily restructured to other formats such as JSON or YAML.

The python toolkit provides other convenience functions for manipulating DL-MONTE inputs, which are expected to be in the same location, in one go. For example,

```
my_input = dlmonte.DLMonteInput.from_directory(my_directory)
```

provides a way to group the three standard input files into a single python object. Such convenience functions can also be used to automate repeated tasks, such as the running of tests.

3.2. Driving execution

The DL-MONTE executable program may be executed from within a python script using standard python mechanisms. Again, convenience methods are provided to aid common operations, e.g.,

```
myrun = dlmonte.DLMonteRunner(executable, working_directory)
myrun.execute()
```

executes DL-MONTE in the standard way provided appropriate input is present in the working directory. The scripting mechanism is sufficiently flexible to allow appropriate launch commands to be used (and is something which should not be tightly prescribed as such commands are often specific on HPC services such as ARCHER).

3.3. Postprocessing

Reading DL-MONTE output provides the basis of many toolkit operations. Functionality is provided to read either standard DL-MONTE “PTFILE” file output and YAML format output. Such data are easily accessed via standard python mechanisms, e.g,

```
data = myrun.output.yamldata.data
print(data)
```

These data structures are also suitable for histogrm reweighting, discussed in the following section.

3.4. Histogram Reweighting

The fundamental idea behind histogram reweighting is to use fully the sampling information provided by a single Monte Carlo simulation. Often, a simulation only reports averages of observable quantities of interest. Such averaging throws away potentially useful information. Histogram reweighting takes a set of instantaneous observable measurements from a simulation, which should be representative of sampling from the appropriate statistical distribution, and uses Boltzmann weights to compute the expectation value of that observable had the simulation been run at a nearby point in phase space of control parameters.

3.4.1. Data Model

To carry out reweighting the toolkit expects data to be presented in a standard way. The toolkit therefore provides an abstract class `ObservableData` which may be extended to provide access to data from any given source. A standard implementation is provided for DL-MONTE data sources, but an appropriate "plugin" extension could be provided for other simulation types.

It is expected that "observations" will consist of one or more time series of values: energy, volume, pressure, density, and so on. Further metadata concerning control parameters (ensemble type, parameters for potentials, bonds, angles, and so on) are also required. For DL-MONTE, these are provided by the three relevant input files for the simulation in question. The internal data representation uses `numpy` arrays so that manipulations can be carried out efficiently.

3.4.2. Reweighting observations and histograms

Given a series of instantaneous measurements of some observables quantity, which will be referred to as O_i, together with corresponding measurements of the energy

E_i, the observable can be reweighted to a new temperature in the NVT ensemble via:

$$\langle O \rangle_{\beta'} = \frac{\sum_i O_i \exp[-(\beta' - \beta)E_i]}{\sum_i \exp[-(\beta' - \beta)E_i]} \tag{2}$$

where β is the Boltzmann factor at which the measurements were taken, and β' is that relevant for the new (nearby) temperature. The exact details of the reweighting depend on the observable required and the ensemble of the simulation, along with the Hamiltonian of the system.

Standard observable data as represented in the python toolkit can be reweighted with some flexibility using the toolkit `Reweighter` class. For example, in the NVT ensemble, observations of the energy and the square of the energy can be used to compute the specific heat. The following code fragment illustrates the energy (`numpy` array `e1`) and the square of the energy (`numpy` array `e2`) being used to generate an array of values of the specific heat at new temperatures `kt_new`:

```
cv = numpy.zeros(nrw)
for nt in range(nrw):
    e1r = reweighter.reweight_obs(e1, kt_new[nt])
    e2r = reweighter.reweight_obs(e2, kt_new[nt])
    cv[nt] = (1.0/(volume*kt_new[nt]**2))*(e2r - e1r*e1r)
```

For each new temperature, the reweighting procedure provides a new estimate of the energy `e1r`, and the squared energy `e2r`, which may be combined to give the specific heat. A typical result for the Ising model is shown in Figure 1.

To reweight a histogram, the situation is similar. If the histogram associated with a time series of M measurements of observable O_i at, for example temperature kT, is written

$$H_{kT}(O_j) = (1/M) \sum_i \delta(O_i, O_j), \tag{3}$$

where O_j represent the histogram bins and $\delta(O_i, O_j)$ counts the observations in bin j, it is possible to reweight the histogram to a new temperature via

$$H_{kT'}(O_j) = \frac{\sum_i \delta(O_i, O_j) \exp[-(1/kT' - 1/kT)E_i]}{\sum_i \exp[-(1/kT' - 1/kT)E_i]}. \tag{4}$$

The toolkit provides a standard `Histogram` class which can be used in association

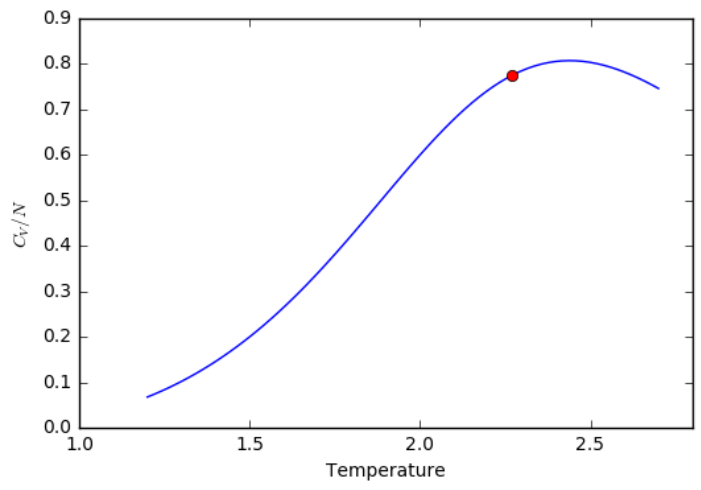

Figure 1: Values of the specific heat in the Ising model from a single simulation (red circle) are reweighted to a series of temperatures (blue line).

with the `Reweighter` class to reweight histograms in a flexible manner. An example of histogram reweighting is given in the following section.

Histogram reweighting cannot generate information where none exists. If one tries to reweight too far away from the parameters of the original simulation, and where no sampling is occurring in the simulation, errors will grow to dominate the result. Some care is required, and guides are available to the maximum safe extent of reweighting — the exact details depend what quantities are under consideration. Further discussion is given with the tutorial material provided with the python toolkit [4].

However, one can combine the information from two or more simulations of the same system at different control parameters to cover a larger area of phase space. Such multiple-histogram reweighting can also be undertaken with the toolkit.

3.4.3. Example: identifying co-existence

The following example uses a simple grand canonical Monte Carlo simulation in DL-MONTE to illustrate the utility of the python toolkit in identifying a good approximation to the correct location of a co-existence point using a single simulation.

(For this problem the location of the co-existence is known beforehand, the simulation has been chosen close to co-existence. Clearly, this cannot be true in the general case.)

The simulation uses a μVT ensemble in which standard Lennard-Jones particles are inserted or deleted with a given chemical potential μ, but otherwise do not move. The Hamiltonian for the system is

$$\mathcal{H} = -U + \mu N \tag{5}$$

where U is the total potential energy, μ is the chemical potential, and N is the number of particles currently in the system.

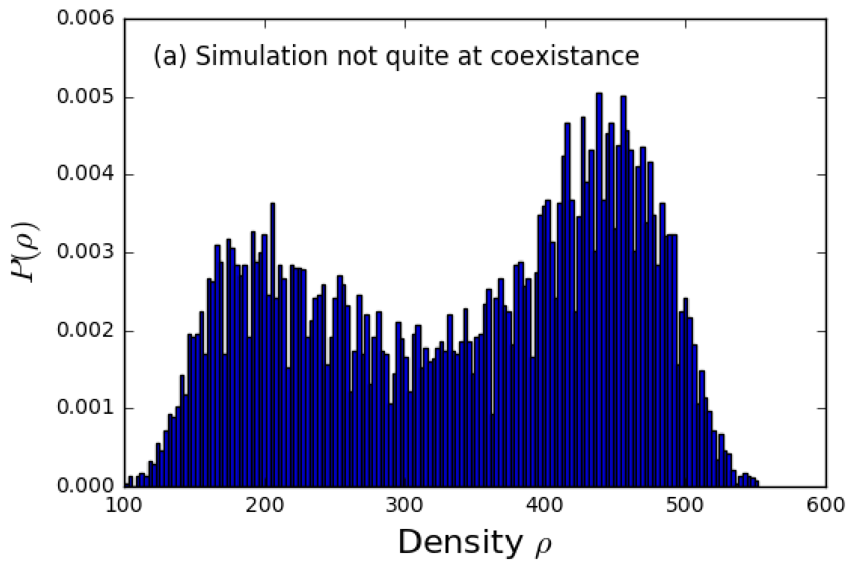

Figure 2: A histogram of the density in a GCMC simulation of Lennard-Jones particles. The two broad peaks in the histogram shows the simulation samples both a low density and a high density phase.

The simulation is run for 100 million Monte Carlo steps, and the number of particles in the system is recorded at regular intervals. If one plots a histogram of the density of the system (effectively just the number of particles in this case) from the set of measurements, one sees the situation in Figure 2. The two broad peaks in the histogram indicates that the simulation samples a low density and a high density phase, but the asymmetry indicates that the simulation is not exactly at co-existence. At co-existence one would expect a perfectly symmetric bimodal structure.

At this point one could undertake a series of additional simulations at nearby parameters to try to find a better estimate of co-existence. However, histogram reweighting provides a more efficient way to do this. Using the appropriate histogram reweighting machinery described in the preceding section, one might try reweighting to a lower value of the chemical potential (which favours the lower density phase) via code of the form

```
rho_histogram.weight_wrt("mu", mu_new)
```

The resulting reweighted histogram at the new chemical potential is shown in Figure 3. In this illustration, the new chemical potential is below the true co-existence value, and the result is weighted more to the low density phase: the value of μ chosen was too low.

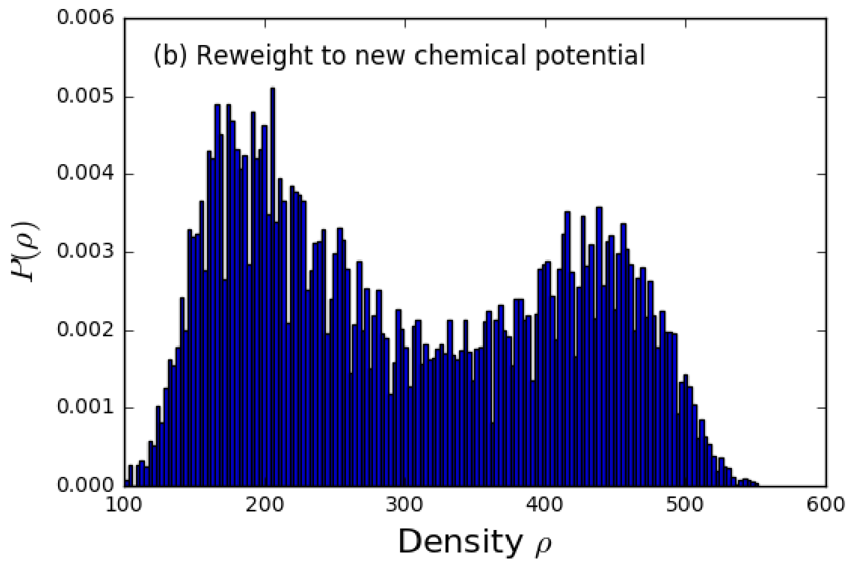

Figure 3: Reweighted histogram of density shifts the position of the maximum of the distribution into the low density phase.

It is now a relatively straightforward matter to use some standard python library functions to undertake an optimisation problem to identify a best estimate of co-existence as defined by some objective. In this case, an appropriate objective is one which tries to locate an optimally symmetric bimodal histogram. Figure 4 shows the optimal situation in this particular example. The value of the chemical potential produced by such an optimisation would then be the best estimate of the co-existence value. An associated error can be obtained via various standard techniques such as bootstrapping.

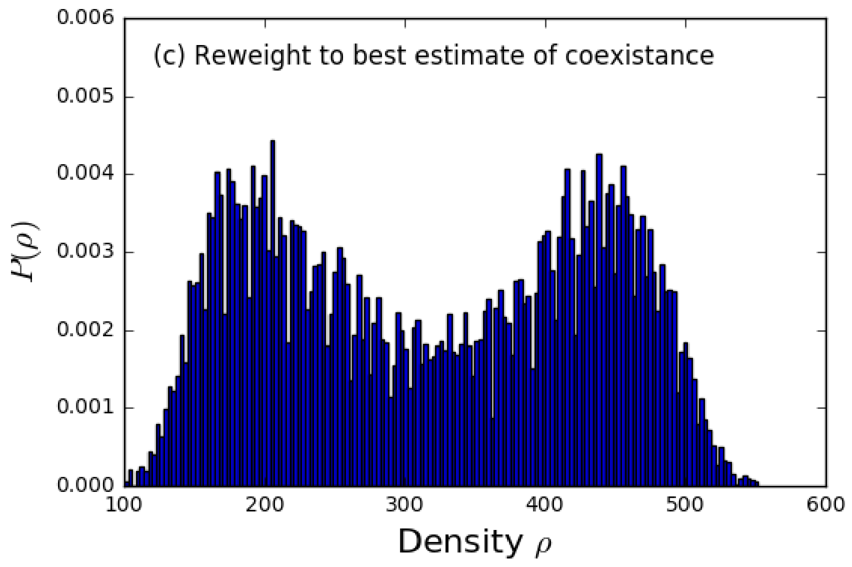

Figure 4: Optimal histogram determined by an objective which attempts to produce a symmetric distribution.

4. Summary

Implementing LSMC in DL_MONTE has enabled LSMC to be applied to a range of systems which have never before been possible to study via LSMC because relevant software never existed. LSMC can now be applied to systems comprised of atoms or rigid molecules interacting via a wide range of force fields, including long-range Coulomb interactions, under conditions of constant density or constant pressure.

This is beneficial to the community because it should enable certain free energies to be calculated in such systems more efficiently, i.e., using less CPU time, than was previously possible.

The development of the python scripting toolkit should allow workflows to be generated more efficiently. For example, a parameter sweep might involve 10 different simulations and the analysis of 10 different sets of results. Bespoke analysis is now provided in a single framework. Moreover the availability of histogram reweighting can reduce the number of simulations required in, for example, identifying the exact location of the co-existence points and curves that form phase diagrams.

Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (texttthttp://www.archer.ac.uk/).

References

[1] S. Bridgwater and D. Quigley, *Phys. Rev. E* **90** 063313 (2014).

[2] A.D. Bruce, N.B. Wilding, and G.J. Ackland, *Phys. Rev. Lett.*, **79**, 3002 (1997).

[3] A.D. Bruce, A.N. Jackson, G.J. Ackland, and N.B. Wilding, *Phys. Rev. E* **61**, 906 (2000).

[4] DL-MONTE: `https://ccpforge.cse.rl.ac.uk/gf/project/dlmonte2/`

[5] M. Fitzgerald, R.R. Picard, and R.N. Silver, *Europhys. Lett.*, **46**, 282 (1999).

[6] M. Marechal and M. Dijkstra, *Phys. Rev. E* **77**, 061405 (2008).

[7] S. Pronk and D. Frenkil, *J. Chem. Phys.* **110**, 4589 (1999).

[8] G.R. Smith and A.D. Bruce, *J. Phys. A: Math. Gen.* **28** 6623 (1995).

[9] T.L. Underwood and G.J. Ackland, Preprint `https://arxiv.org/pdf/1609.04329.pdf` (accessed February 2016).