Project: Entropic spring – rediscover Hooke's law with Monte Carlo

MC sampling, statistical physics, random walks

Summary. You will build a simulation of rubber bands, being pulled by a force, in 1 dimension. You will calculate basic observables ("how stretched is the rubber band?") as function of force in two ways: 1) by reweighting, and 2) by direct simulation.

This exercise will *not* provide you with a large amount of template code to start with. Rather, it is up to you, to set up reasonable data structures etc. for this project.

The main goal of this exercise is to familiarize you with Monte Carlo methods. You should therefore *not* rely on library implementations of Monte Carlo selection apart from sampling a random number in the interval $R \in [0, 1]$

Learning goals. When you have completed this exercise, you will be able to:

- Translate a simple physical model (the entropic spring) into a working computational simulation using object-oriented programming.
- Apply Monte Carlo sampling techniques to generate and analyze ensembles of microstates, including the use of weighted (reweighted) samples.
- Quantitatively compare Monte Carlo results to analytical predictions, using ratio plots and statistical measures to validate simulations.
- Diagnose and interpret the performance and limitations of Monte Carlo methods, including ensemble overlap and effective sample size.

Physics background

We consider a simple model of a rubber band in one dimension. A chain consisting of $N \gg 1$ links, each of length a. For simplicity, we will assume that all configurations of the chain have the same energy, all that matters for the properties of the chain is how many times it doubles back on itself. We will thus have a number of links pointing in the +-direction, and the rest pointing in the −-direction. If we call the total length of the chain L, and the number of links pointing in the +-direction n, we have:

$$L = a(2n - N). \tag{1}$$

We know from basic thermodynamics, the relation between entropy (S) and force (f) at fixed energy (E) to be:

$$\vec{f} = -T \left(\frac{\partial S}{\partial \vec{x}} \right)_E, \tag{2}$$

at temperature T with $\vec{x}$ being the spatial extension[1]. We reduce to one dimension for simplicity:

$$f = -T \left(\frac{\partial S}{\partial L} \right)_E, \tag{3}$$

and our task is now to calculate the microscopic entropy, and see how it changes with L.

[1]Note: $\vec{x}$ may be more familiar from the definition of work: $dW = \vec{f} d\vec{x}$: work is equal to force times path.

Figure 1: Ludwig Boltzmann (1844–1906), Austrian physicist and one of the founders of statistical mechanics. He established the microscopic interpretation of entropy, relating it to the number of accessible microstates through eq. (4) This relation is famously engraved on his tombstone in Vienna's Zentralfriedhof cemetery, as seen on the picture (Credit: Thomas D. Schneider, MIT). Boltzmann's work laid the foundation for modern thermodynamics, kinetic theory, and statistical physics.

From Boltzmann's tombstone (see fig. 1), we know that the entropy counts the number of microstates corresponds to a given macrostate (Ω):

$$S = k_B \ln(\Omega), \tag{4}$$

with k_B being Boltzmann's constant. A macrostate is specified by the total extension L. Many microstates (different orderings of $\pm$ links) correspond to the same L.

The number of ways to arrange the chain such that n links point in the positive direction, is given by the Binomial coefficient:

$$\Omega(N, n) = \binom{N}{n} = \frac{N!}{n!(N-n)!}. \tag{5}$$

Using Stirling's approximation[2] we get:

$$\frac{S}{k_B} \approx N \ln(2) - \frac{L^2}{2Na^2}. \tag{6}$$

Exercise: Fill in the remaining steps of algebra in this derivation. (Hint: First you must apply Stirling's approximation. This gives you two logarithms, which you can Taylor expand in $L/(Na)$ which is always smaller than unity.)

Inserting in eq. (3), gives directly:

$$f = \frac{k_B T}{Na^2} L. \tag{7}$$

Comparing to Hooke's law: $f = kL$, we identify an entropic spring constant $k = k_B T/(Na^2)$.

Simulation tasks

Simulation task I: microstates and macrostates

Consider first an ensemble of unbiased microstates. That is, rubber bands which we are not pulling with any force, they are just random configurations.

- Set up reasonable data structures, a `Link` class and a `RubberBand` class both holding relevant parameters. The `RubberBand` class should have a method which calculates the total length L. The constructor of `RubberBand` should take N as a parameter, and randomly sample `Link`s pointing in + -or − direction, and store them as a data member.
- Monte Carlo sample a large number of rubber bands. Let $a = 1$ for simplicity, and let $N = 100$.
- Make a histogram of the $P(L)$-distribution. Compare to the analytic result: $P(L) = \Omega(N, n)/2^N$ (remember: Ω is the number of microstates with extension L, 2^N is the total number of microstates. The ratio is the probability.)
- Your comparison should (at least) consist of plotting the two overlaid with each other, a plot of the ratio of the two (unity means identical; you could also use purity $S/(S + B)$ if you wish) and some sort of quantitative measure, such as χ^2/ndf. In figure 2 you can see what the figure should look like and with inspiration for how to plot it and present it.

[2]Which states that $\ln(n!) \approx n \ln(n) - n$ when n is large.

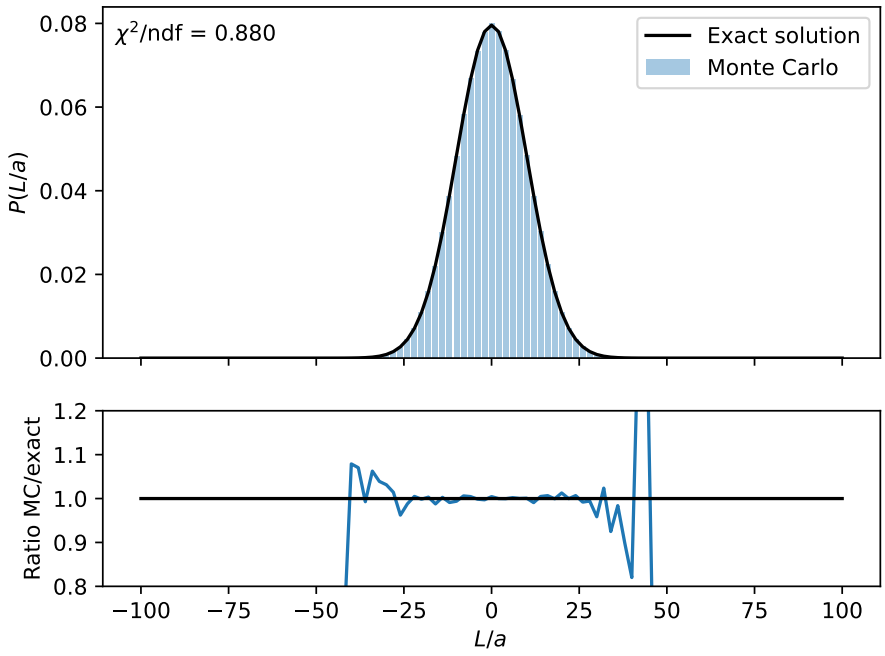

Figure 2: Unbiased rubber band dimensionless length distribution. Each rubber band has $N = 100$ links with $a = 1$. Monte Carlo ($M = 10^6$ samples) histogram is overlaid with the exact solution $P(L) = \Omega(N, n)/2^N$, the ratio plot shows good agreement (although diverging beyond the region of statistical coverage), as does the χ^2/ndf of approximately unity.

Simulation task II: pulling with a force, reweighted

We now introduce an external force f pulling the rubber band in the positive (+) direction. In the previous task you constructed a Monte Carlo sampling of rubber bands unaffected (unbiased) by a force, here we will treat a biased distributed. We will do it using a *reweighting technique*. This means that you do not need to resample the rubber bands, just calculate a new weight with which they enter.

Physical idea: Let E be the energy of a single microstate. We still assume that its internal energy remain constant (E_0). Applying a constant force f along the +-direction, a configuration with extension L does work $-fL$, so its total energy is:

$$E(\text{microstate with } L) = E_0 - fL. \tag{8}$$

At temperature T, such a state has the Boltzmann weight (with $\beta = (k_B T)^{-1}$):

$$w \propto e^{-\beta E} = e^{-\beta E_0} e^{\beta f L}. \tag{9}$$

We already know, that for a given macrostate of length L, there are exactly $\Omega(L)$ corresponding microstates. Adding all their weights together, gives the macrostate weight:

$$W(L) \propto \Omega(L) e^{\beta f L}. \tag{10}$$

Finally, the probability to obtain a given macrostate L, can be found by normalizing over all possible L, here dummy-labelled L':

$$P(L) = \frac{\Omega(L) e^{\beta f L}}{\sum_{L'} \Omega(L') e^{\beta f L'}}. \tag{11}$$

- Extend your `RubberBand` class with a method that calculates the Boltzmann weight of a given microstate when a force f is applied:

  $$w(\text{microstate}) = e^{\beta f L}, \qquad \beta = \frac{1}{k_B T}.$$

  Here L is the total length calculated by your method from Task I. Hint: Work in units where $k_B = 1$, and start with $T = 1$ such that $\beta = 1$.

- Generate a large ensemble of microstates (still unbiased, random ±-configurations).

- Construct a *weighted histogram* of the L distribution, where the microstate weight is used. This simulates what happens if the ensemble is pulled with force f.

- Compare your weighted histogram to the analytic result:

  $$P_f(L) = \frac{\Omega(N, n)\, e^{\beta f L}}{Z(f)}, \qquad Z(f) = \sum_L \Omega(N, n)\, e^{\beta f L}.$$

  Here $\Omega(N, n)$ is the same combinatorial factor you used in Task I, and $n = (L/a + N)/2$ from inversion of eq. (1).

- Make a plot comparing your Monte Carlo $P_f(L)$ and the analytic $P_f(L)$ for at three small nonzero value of f, e.g $f = \{0.01, 0.05, 0.1\}$. Again, plot their ratio. Be careful with χ^2 since it is poorly defined for reweighted samples. I recommend skipping it here.

- Try three larger values, e.g. $f = \{0.1, 0.5, 1.0\}$. Do you still see ok performance? Why not? Diagnose your reweighting by calculating the number of effective samples:

$$\mu_{\text{eff}} = \frac{1}{\sum_i \tilde{w}_i^2} \text{ with: } \tilde{w}_i = \frac{w_i}{\sum_j w_j}.$$

  Plot μ_{eff} as function of the force. When would you expect reweighting to start struggling?

Simulation task III: pulling with a force, direct simulation

As you saw in the previous task, reweighting works only as long as the unbiased and biased ensembles overlap sufficiently. When the force grows too large, only very few unbiased configurations contribute significant weight. To obtain reliable results at larger f, you must instead generate the configurations directly from the biased distribution.

Physical idea: When a constant external force f is applied, each link in the rubber band experiences a bias: it is more likely to point in the +-direction than in the −-direction. From the Boltzmann factor for a single link, the probability for a link to point in the +-direction is:

$$p_+(f) = \frac{w_+}{w_+ + w_-} = \frac{e^{\beta f a}}{e^{\beta f a} + e^{-\beta f a}} = \frac{1}{2}(1 + \tanh(\beta f a)), \tag{12}$$

where tanh in the last equality is the hyperbolic tangent function. Correspondingly:

$$p_-(f) = 1 - p_+(f). \tag{13}$$

Thus, for a given f you can sample each link independently with these probabilities, instead of assigning random ± directions with equal weight.

Exercise: Fill in the algebra needed to write the last equality in eq. (12). (Hint: Look up the definition of $\tanh(x)$ in terms of exponentials).

With that in place, we can calculate the expectation value for our upcoming "observable": $\langle L \rangle(f)$, the (average value) of length given a force. It is simply:

$$\langle L \rangle(f) = a \cdot (\text{number of links in the } + \text{-direction} - \text{number of links in the } - \text{-direction}) \tag{14}$$

$$= aN(p_+ - p_-) = Na(2p_+ - 1) = Na \tanh(\beta f a). \tag{15}$$

In the small-force limit $\beta f a \ll 1$, we can Taylor expand $\tanh(\beta f a) \approx \beta f a$ such that:

$$\langle L \rangle(f) \approx \frac{Na^2}{k_B T} f, \tag{16}$$

matching exactly what was found in eq. (7). The task is now to Monte Carlo simulate a large number of biased states and compare the results to theory.

- Change the constructor of `RubberBand` adding optional arguments `biased = True/False`, `kBT` and `force`. If those arguments are present, `Link` directions should be sampled from the biased distribution instead of unbiased[3].

[3]Note that eq. (12) directly gives a probability to be + as function of the given parameters. Instead of being + with $p = 0.5$, the link should now be + with probability $p_+(f)$.

- Generate a large ensemble (M) of rubber bands at several values of f using this biased sampling. For each f, compute the mean extension:

$$\langle L \rangle(f) = \frac{1}{M} \sum_{i}^{M} L_i(f),$$

  and its standard deviation.

- Plot $\langle L \rangle$ as a function of f for small and moderate values of f. Compare to the analytic results derived above:

  1. The full result derived from microscopic probabilities in eq. (14), and
  2. The linear (small-force) approximation, Hooke's law, in eq. (16).

- Use your simulation to extract the effective spring constant k_{eff} by fitting $\langle L \rangle$ vs. f in the linear region. Does k_{eff} agree with the theoretical value $k_B T/(Na^2)$?

- Finally, illustrate what happens when f becomes large. Show that the distribution $P_f(L)$ becomes narrow and strongly shifted, and that the linear approximation (Hooke's law) breaks down

What to submit?

Readable, documented code in a merge request to the exercise project. You should clearly describe your results and the methods used. It must at least include:

- Code:
  - Classes for `Link` and `RubberBand`.
  - Cleanly separable code for solving the different simulation tasks.
  - Reusable code for plotting comparisons with theory with a ratio plot.
  - Code for producing all comparison plots, reweighting plots and direct simulation plots.
  - Information about how to run the code, documentation of dependencies if needed.
- Documentation:
  - A clear description of the problem (no need to go through all the theory, but an overview).
  - Description of the theoretical curves you compare to.
  - Clear descriptions of the use of both direct simulation and reweighted techniques.
  - A discussion of when reweighted methods breaks down.

Finished early? Stretch goals

If you complete the core tasks ahead of time, pick one or more of the extensions below. Document what you changed and, if relevant, how it affected your results or the runtime of your code.

Python is so slow! (optimization, C++)

You may have noticed that simulating a large number of rubber bands can be painfully slow in Python. Delegate the simulation task to C++ and call it through an API (see the 2nd day Python tutorial from the beginning of the course). You should deliver:

- C++ code callable from Python with clean instructions for how to build and run through the API.
- Documentation of runtime improvements, for example number of rubber bands generated per clock time in Python vs. C++.

Animate a rubber band (visualization)

Create a short animation of a single rubber band as the force increases, showing the random links gradually aligning with the force direction. Let "time" in the animation be the force, which is gradually building up. Your animation should show the individual links. You should deliver:

- Runnable code to produce the animation.
- A .gif file uploaded to the gitlab.
- Explanation of your process and the animation in the documentation.

Happy Hacking!