Computational Physics - Final Assignment

FOLDING OF A LATTICE PROTEIN

USING THE METROPOLIS MONTE CARLO ALGORITHM

Contents

Practical information

Deadlines.
Deadlines are communicated through Nestor / e-mail.

Handing in the reports.

- Please hand in your reports (.pdf format) + your code (i.e. a working notebook file, or a separate .py file) before the indicated deadline**,** through Nestor.
- If there is a serious problem or exceptional circumstance that prevents you from meeting this deadline, please contact the course coordinator Prof.dr.ir. Patrick R. Onck at p.r.onck@rug.nl .
- Working together is allowed, but you **have to write and hand in your own code in the end, and have to write the report yourself**. Fraud will be taken seriously and appropriate actions will follow.
- If you do not hand in anything before this deadline, we will not register a grade. If you do hand in a report + files, a grade will be registered (also if it is an insufficient grade).
- If you do not meet the deadline, there is the possibility to do a 'resit'. This means that you get an extended deadline for your assignments. However, this will come at the cost of an additional, second, assignment.

Supervision and getting help.

- Supervision for the final problems is available. For this Monte Carlo assignment, the supervisor will be Albert Thie, who can be reached via email at a.s.thie@rug.nl .
- In the final weeks of the course (from tutorial 12 on, up to the exam week), the computer rooms are available to work in during the tutorial hours. The tutorial assistants will be present during some of these slots (how this is organized will be communicated later) to give you feedback on the pseudocode/code associated with your final assignments.
- I'm stuck: what do I do?
    - If you find that you get stuck on the assignments, then the teaching assistants are there to help! Please reach out timely if you find that you get stuck.
    - Before reaching out, please make sure to follow the debugging tips provided at the end of the notebook for tutorial B and/or do some Googling yourself. Try to make the bug/error visible and reproducible, such that your TA can help you most efficiently.
    - When reaching out for assistance, send via email the executable python file/notebook together with the steps (again: see tutorial B) to solve the problem.
    - If you want to drop by to discuss your code, please send an email beforehand: this might save you disappointment in case the TA does not have time or is not present.

Introduction

In this assignment, you will use the Metropolis Monte Carlo (MMC) algorithm and a simple protein model to simulate protein folding. For this, you will consider a H-P protein model, where a protein is modeled as a chain of either Hydrophobic or Polar monomers. You will pick a protein sequence and study its folding process via MMC simulations at various temperatures. In other words: you will try to find the lowest energy configuration possible.

The energy of the model protein is defined as

$$E = -\varepsilon f, \tag{1}$$

where E is the total energy of the chain, ε is the energy associated with a contact between two H-monomers and f is the total number of H-H contacts. A contact is defined as two H-monomers being present next to each other on the lattice, while not being each other's direct neighbours within the chain (i.e., not being bonded neighbours). Note that only H-H contacts count towards the energy, while H-P and P-P contacts do not. For example, if the neighbour next to an H-monomer is another H-monomer, it lowers the energy, but if it is a P it does not lower the energy of the protein chain.

Setting up the MMC program

In this section, you will build the actual MMC program to use in the protein folding simulations. For the tasks marked with an asterisk (*), you should start by writing out a pseudo-code. This pseudo-code is to be provided in the report.

Part one: Preparing, visualizing and analyzing the chain

The first step is to create an initial lattice configuration for your H-P sequence, where the initial configuration should be an ordered collection of discrete lattice points and corresponding monomer type, in such a way that each neighbour is exactly one lattice unit apart and each lattice site can at most be singly occupied by a protein monomer (i.e., no intersections or overlapping monomers). There are different approaches to do this. Here, you will use a self-avoiding random walk to generate the protein configuration where you randomly select a direction to try and place your next monomer (provided the lattice point is still free). Next, we need to visualize a protein chain and calculate its energy.

Tasks. Perform the following programming tasks:

1. *Write a random walk function that creates a set of N connected lattice points. Use discrete, integer-valued coordinates. For each of the positions, also randomly assign a bead type (either H or P) based on a weighting factor (hydrophobic fraction) that you can specify as input.

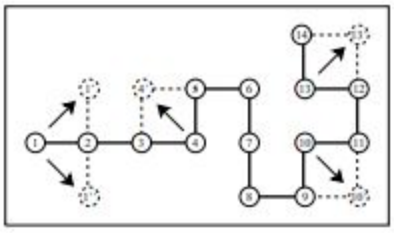
2. Grow your protein chain with $N = 25$ residues and confirm your output. Confirm that, on average, the ratio of H to P monomers agrees with the weighting factor. Make sure your program outputs the HP sequence, since you will have to store this later on.

3. Write a function to visualize the protein conformation. It may be helpful to clearly visualize the bonds between monomers, the monomer types and the H-H contacts.

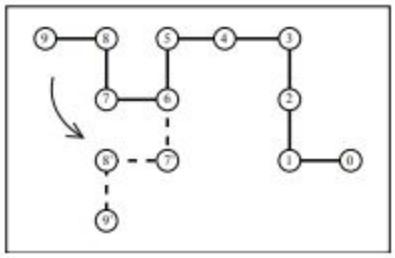
4. *Write a function to calculate the energy associated with a protein conformation. Use reduced units (this amounts to setting $\varepsilon = 1$ in your computations). According to Eq. (1), this function will have to count the number of H-H contacts. Verify the implementation of the energy function by visual inspection.

Part two: Implementing the Monte Carlo moves

Since you have initialized a chain conformation, the next step is to implement the two trial moves that allow you to sample a new spatial configuration of the same protein chain. Although a multitude of trial moves exist, you will implement the following two different moves: a *kink jump* (Fig. 1a), also called *end rotation* if it occurs at the first or final monomer, and a *pivot move* (Fig. 1b), where you pivot part of the polymer chain around a selected monomer.



(a) The kink jump or end rotation.



(b) The pivot move.

Figure 1. The Monte Carlo moves for this assignment.

Tasks. Now, expand your program using the following steps:

1. *Implement a kink jump / end rotation in a function (that takes as input the current protein conformation) using the below steps:
    a. Randomly pick a monomer from your chain.
    b. Determine whether a kink jump is possible by checking (i) the occupation of the grid nearby and (ii) the locations of its neighbouring monomers.
    c. If a kink jump is possible, determine the new coordinates of the monomer and update the chain configuration. Otherwise, keep the old configuration.

2. *In a similar fashion, implement the pivot move in a function that takes the current protein conformation as input) using the below steps:
    a. Pick a random monomer k from your chain, and choose which end of the chain will be pivoted. Also, choose the direction of rotation.

b. Rotate all monomers $1, 2, ..., k$ (or $k, ..., N$) by 90 degrees around this monomer, as shown in Fig. 1b. This move may seem complicated, but it essentially consists of a translation and rotation steps:
   i. Shift the to-be-rotated monomers such that monomer k lies at the origin.
   ii. Apply the rotation to each monomer using a rotation matrix:
   $$x_{new} = x_{old} \cos\theta - y_{old} \sin\theta, \quad (2a)$$
   $$y_{new} = x_{old} \sin\theta + y_{old} \cos\theta, \quad (2b)$$
   where θ is the rotation angle.
   iii. Shift the rotated monomers back such that monomer k is at its original position again.
c. If there is no overlap between the monomer positions in the new configuration, update the coordinates. Otherwise, keep the old configuration.

3. Make sure to (visually) verify that both moves have correctly been implemented!

Part three: The full MMC program

With a starting configuration, and subroutines for performing the kink jump / end rotation and pivot moves in place, it is time to write the overhead part of the program, which chooses when to perform the trial moves, and which applies the MMC algorithm to decide whether or not to keep the trial conformation. For your convenience, we recapitulate the MMC algorithm below, before introducing the necessary programming steps.

Starting from a configuration C_α, with an energy E_α, perform the following steps:

A. Create a trial configuration $C_{\alpha+1}$ by performing either a pivot move or a kink jump/end rotation (chosen randomly) using the functions from the previous part. The trial configuration is always different from the current configuration.
B. Calculate the energy $E_{\alpha+1}$ of this trial configuration
C. If the energy of the new configuration is lower than the previous energy, set the configuration to $C_{\alpha+1}$, and update the current energy to $E_{\alpha+1}$.
D. If the energy of the new configuration is higher than the previous energy, the probability to accept the new configuration is the Boltzmann weight $w = \exp(-(E_{\alpha+1} - E_\alpha)/k_B T)$. Accept the new configuration (like in step 3) if the Boltzmann weight exceeds a random number between 0 and 1. Otherwise, reject the move, retain the old energy and proceed to the next trial move.

Note: In this exercise we use reduced units. This means that we set $\varepsilon = 1$ and $k_B = 1$, or equivalently, that we can express the temperature in units of ε/k_B.

Tasks. *Adapt all functions you have written in a complete MMC program that takes as input the temperature, the chain length, the hydrophobic fraction and the number of MMC moves to perform. For your total program structure, you could foresee a comparable structure to that shown in Listing 1.

```
1.  Program for MMC folding simulation of N-residue HP protein
2.
3.  Input: temperature T, chain length N, maxIterations, samplingFrequency
4.  Initialize HP protein of length N
5.  Calculate initial energy Ea
6.  Store/Print sequence
7.
8.  For n<maxIterations
9.      Let r be a random number between 0 and 1
10.     If r>0.5:
11.         Try kink jump/end rotation until new configuration found
12.     Else:
13.         Try pivot move until new configuration found
14.
15.     MMC steps B-D:
16.         If accepted:
17.             Update conformation, energy
18.
19.     If n%samplingFrequency:
20.         Calculate/Store energies or other properties of interest for post-processing
```

Listing 1. Example structure of the MMC code.

Protein folding simulations

Now that your program is fully functional, it is time to run some physics simulations. We are going to investigate the folding behaviour of HP proteins as a function of temperature, chain length and hydrophobic fraction.

NOTE: Before starting, make sure your code is indeed completely functional! You should be able to perform a folding simulation for a randomly generated protein given a reduced temperature. Where applicable, do some code optimizations to save time during the simulations.

Benchmarking

1. Initialize a protein of 25 residues with a hydrophobicity fraction around 50%. Save the HP sequence that you use for reporting, as well as the initial configuration.

2. Pick a reduced (!) temperature at which you want to perform a benchmarking simulation. For your HP sequence, run the MMC program, while storing the energy of the system every 100 steps. Answer the following questions:
    a. How many iterations are needed to observe a reasonable convergence to a (fluctuating) minimum chain energy at this temperature?
    b. Does the minimum energy configuration correspond to a unique native state? In other words, if you repeat a simulation with the same parameters, will you obtain the same ‘folded’ conformations?
    c. (How) does this change with temperature or the hydrophobicity of your protein?

    NOTE 1: You can use a running average to interpret your results more clearly.
    NOTE 2: Try a few different temperatures until you observe convergence.

Simulated annealing

1. Based on the above observation, add a temperature loop to your program and choose a reasonable value for the amount of iterations, such that you can repeat your MMC simulation for enough different temperatures within a overseeable amount of computation time. For your specific protein, you will perform 'simulated annealing'; this means gradually reducing the temperature after every set amount of MMC iterations, until you reach temperatures that are very small. For each temperature, perform an MMC simulation and calculate the following properties:
    a. *During* the simulation, store the protein energy in an array after every set amount of MMC steps.
    b. *During* the simulation, calculate the radius of gyration, which is defined as
    $$R_g = \sqrt{\frac{1}{N} \left\langle \sum_{k=1}^{N} (\vec{r}_k - \vec{r}_c)^2 \right\rangle} \quad , \tag{3}$$
    where $\vec{r}_k$ is the position of monomer k, and $\vec{r}_c$ is the center-of-geometry of the protein. Store these values in an array as well.
    c. *After* the simulation, calculate the heat capacity for that specific temperature. The heat capacity is given by
    $$C = \frac{\langle E^2 \rangle - \langle E \rangle^2}{k_B T} \, , \tag{4}$$
    where the brackets indicate an ensemble average.

    NOTE: Before taking averages, give your protein enough time to relax to a given temperature. A (rudimentary) way to do this is to discard the first 10% of the trajectory, provided you perform enough MMC steps.

2. Make a few histograms of the energy and radius of gyration distributions. How do the averages of these distributions change with temperature? What about the width of the distributions?

3. Plot heat capacity versus temperature. You should observe a single peak at a specific temperature. If not, double-check that you performed enough MMC steps and that you are in the right temperature range.
    a. At which temperature do you observe the peak in the heat capacity?
    b. (How) can you relate the heat capacity plot to the average energy plot? And to the radius of gyration plot?
    c. What does the peak in the heat capacity indicate? Relate your answer to the plots you have produced.

The role of hydrophobicity

1. Try a few different values for the hydrophobic fraction of the protein. Produce again plots of heat capacity versus temperature for these new proteins. Do you observe a relation between the hydrophobic faction and the peak temperature?

2. What can you conclude on the folding process based on the amount of hydrophobic residues present in your protein? Support your conclusion by comparing the heat capacity curves for the different proteins, and by comparing the minimum energies and visualizations of the corresponding conformations.

Extra Information

Useful references

- Landau, R.H. and Bordeianu, C.C., 2015. *Computational physics: problem solving with Python.* John Wiley & Sons.
- Frenkel, D. and Smit, B., 2001. *Understanding molecular simulation: from algorithms to applications* (Vol. 1). Elsevier.
- Lau, K.F. and Dill, K.A., 1989. *A lattice statistical mechanics model of the conformational and sequence spaces of proteins.* Macromolecules, 22(10), pp.3986-3997.
- Wüst, T., Li, Y.W. and Landau, D.P., 2011. *Unraveling the beautiful complexity of simple lattice model polymers and proteins using Wang-Landau sampling.* Journal of Statistical Physics, 144(3), p.638.

Grading criteria

- In the table below you will find the required (sub)sections of the report, and the general grading criteria for each section.
- Each assignment/problem comes with its own specific sub-questions. Make sure to process the different sub-questions of the assignments in the relevant sections in the report (for example: no derivations in the results sections).
- The grading criteria are in line with the learning objectives, see the ocasys page of the course.
- **Please make sure that all of the below sections are actually present in your report, and that you try to follow the guidelines for these sections.**

1. **Introduction section:** This part of the report should at least contain (not specifically in the order as presented below):
    1. A description of the physical problem to be analyzed. You are encouraged to provide an image, or to consult literature sources to provide some additional background. Also indicate what properties, parameters, research questions etc. you will explore in your report.
    2. Announce which computational methods you will use, and explain the general purpose/context of these methods.
    3. The relevant formulae to solve (i.e. PDE's, ODE's, the general algorithm, etc.) and a brief discussion of any analytical solutions (if they exist) that you can compare your numerical result with. Make sure to quickly browse the literature (the references provided in your assignment are a good start) for some background on your assignment problem.

Weight: **1 pt**
Learning objective: **1**

2. **An algorithm/program structure section.** This part of the report should contain a description of the kind of algorithms that you used, and is the perfect place to showcase your understanding of the simulation method. This section should contain:
    1. An explanation of the algorithms that you used to solve the physical problem at hand.
    2. **An explanation, including pseudocode** that details how you implemented the algorithms that you yourself wrote for the assignment.
    3. If applicable: also explain in this section how the equations are discretized (i.e. what step sizes to take, implementation of boundary conditions, etc.).
    4. If applicable: Discuss what kind of errors (size, sources) you would expect, based on the algorithms that you used.

Weight: **2 pts**
Learning objective: **2**

3. **A results section (possibly combined with discussion section).** In this section, provide the output of your code (i.e. graphs, tables, visualizations, etc.) and provide your observations. This section should contain:
    1. Graphs of a decent quality. Make sure that the graphs are legible, that it is clear for the reader which variables are plotted (use axis labels) and that you include a legend in case you plot multiple types of data/multiple lines in the same graph. You are encouraged to make animations, you can send these separately.
    2. Your observations, based on the results. You should clearly indicate in the main text what you see in each figure (however, please interpret the results in the discussion section).
    3. Also included in the grade of this part is the 'correctness' and 'quality' of your results: are the results correct? Is your simulation converged?

Weight: **3 pts** (together with discussion section)
Learning objectives: **1,4**

4. **Discussion section (possibly combine with the results section).** In this section, you should form an interpretation of your results, and possibly draw comparisons to theory or expectations. This section should at least contain:
    1. An interpretation of the results in light of the questions of the assignment. How do your results compare to the theoretical or experimental results?
    2. A discussion on the applicability/accuracy/feasibility of the computational method to the problem that you studied
    3. If the method allows, incorporate a discussion on the quality of the results, i.e. assess the error sources and convergence of your program, and highlight possible alternatives or improvements.

Weight: **3 pts** (together with results section)
Learning objectives: **1,4**

5. **Conclusion section**. In this section, you should concisely but concretely summarize your report. This section should contain:
    1. A mention of the research problem/motivation
    2. A re-iteration of the used methods and developed algorithms.
    3. The main results obtained and the main conclusions inferred from these results (and the discussion section)
    4. A reflection/perspective statement: how did the computational method work out for this problem, and are there better ways to study your problem?

Weight: **1 pt**
Learning objective: **5**

6. **The (quality of the) complete code.** You do not need to list your complete code in the appendix of the report but are free to do so. What we **do** want to receive is the program as a jupyter notebook or a separate executable .py file. We will grade the quality of your code based on some of the aspects that you practiced throughout the course, and reward initiative to improve the working of your code. Points that we consider in judging the quality of your code are:
    1. Overall code structure: are your loops, algorithms etc., written efficiently? Do you use unnecessary loops, or initialize unnecessary variables? Do you use functions for repeating parts, and do you use the appropriate data structures?
    2. Legibility of your code for a third person. Did you use descriptive variable names or the proper amount of comments/docstrings? A little commenting goes a long way to help yourself and others to understand the code better.
    3. Features added by yourself that optimize the working of the code. Examples are the use of additional python packages for checkpointing/storing data (think of pickle, for example).

Weight: **2 pts**
Learning objectives: **1,3,4**