University of Göttingen
Institute for Theoretical Physics
Methods of Computational Physics, WiSe 2022/23
Instructor: Prof. Dr. S. Schumann
Tutors: Dr. E. Bothmann, M. Knobbe

Project 1: Monte-Carlo Basics

Go to Stud.IP ▸ Übung: Methods of Computational Physics ▸ CloCked to upload a single PDF document with your results and answers by November 8th, 10am.

1. Monte-Carlo integration (16 points)

a) Consider the integral

$$I = \int_a^b \mathrm{d}x\, f(x). \tag{1}$$

The Monte-Carlo estimator for I is given by

$$I_N = V \cdot \langle f \rangle_N = \frac{V}{N} \sum_{i=1}^{N} f(x_i) \quad \text{with} \quad I_N \xrightarrow{N\to\infty} I, \tag{2}$$

where the x_i are random numbers in the range $[a, b)$, sampled according to a uniform distribution, and $V = b-a$ is the integration volume. The estimator for the standard error of I_N is given by

$$\sigma_N = V \sqrt{\frac{\langle f^2 \rangle_N - \langle f \rangle_N^2}{N-1}}. \tag{3}$$

Implement a Monte-Carlo integrator for $f(x) = x^4$, using the parameters $a = 0$, $b = 1$ and $N = 1000$, and report your results for I_N and σ_N. How many multiples of σ_N are between your result of I_N and the exact result for I? What does this mean? (4 points)

b) Repeat your calculation of I_N with $M = 1000$ different sets of random numbers. Plot a histogram of the resulting I_N and fit a Gaussian distribution. Compare the standard error σ you obtain for the Gaussian distribution with the Monte-Carlo estimator of the standard error of I_N found in part a). (4 points)

c) Improve your Monte-Carlo integrator using importance sampling. As sampling distributions, use $g(x) = 2x, 3x^2, 4x^3$ and $5x^4$. For each $g(x)$, calculate I_N and σ_N as above and report your results (e.g. in a table or plot). Additionally, make a log-log plot of σ_N versus N for each $g(x)$. Discuss your results. (8 points)

2. Random Walk in 2D (8 points)

a) Implement a random walk in two dimensions. The random walk should start at the origin, $(x_0, y_0) = (0, 0)$. For each step, choose uniform random values for $\Delta x'$ and $\Delta y'$ in the range $[-1, 1)$. Then normalise the step to be of unit length:

$$\Delta x = \frac{1}{L}\Delta x' \qquad \Delta y = \frac{1}{L}\Delta y' \qquad L = \sqrt{\Delta x'^2 + \Delta y'^2}. \tag{4}$$

Draw 2D plots of three independent random walks, each with $N = 1000$ steps. (4 points)

b) Now perform $M = 1000$ independent simulations with $N = 10000$. Plot a histogram of the distance from the origin after the last step, R_N. Compare the *root-mean-square* distance $R_{\mathrm{rms},N} = \sqrt{\langle R^2 \rangle_N}$ to the theoretical expectation of $R_{\mathrm{rms}} = \sqrt{N} \cdot r_{\mathrm{rms}}$ for large N, where $r_{\mathrm{rms}} = 1$ given our unit step size. (4 points)

3. Protein Folding as a Self-Avoiding Random Walk (12 points)

A protein is a large molecule made up of a chain of building blocks called monomers. Let us consider one containing two different monomers. One is a non-polar hydrophobic (H) monomer that is repelled by the surrounding water. The other is a polar (P) monomer that is attracted by the water. The spatial structure of the protein results from a folding process in which random coils of chains rearrange themselves into a configuration of minimum energy E.

Our goal now is to create a variation on the random walk problem that models the folding process and produces the lowest energy state of a H–P-sequence of various lengths, see Fig. 1. The random walk can only visit the nodes of a regular 2D square lattice, and each node can only be visited once, i.e. the random walk is self-avoiding. We take the energy of the protein to be $E = -\epsilon f$, where ϵ is a positive constant, and f is the number of H–H neighbours on the lattice that are *not* direct neighbours on the chain. Accordingly, we expect the natural states of H–P sequences to be those with the largest possible number f of H–H contacts.

Implement the described model as follows:

1. Set up the random walk on a regular 2D square lattice with 31 grid points in each dimension. If the point $(1, 1)$ denotes the corner in the lower left, then $(16, 16)$ is the centre of the grid. Begin by placing a random monomer at the centre. For the probability of an H monomer, use $p_{\mathrm{H}} = 0.7$. Accordingly, the probability of a P monomer is $p_{\mathrm{P}} = 1 - p_{\mathrm{H}} = 0.3$.

2. Take a step in a random direction. After each step, choose a monomer at random, using the same probabilities p_H and p_{P}, and place the monomer on the new lattice site.

3. Restrict the walk such that the positions available for each step are the empty neighbouring sites.

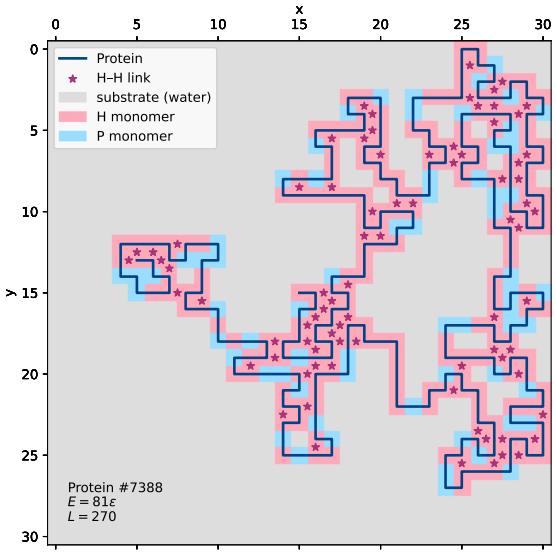

Figure 1: A protein model as a self-avoiding random walk on a 2D lattice, where each lattice site holds either a hydrophobic (H) or a polar (P) monomer. The "H–H links" between lattice nodes indicate regions where two H monomers are neighbours (without being directly connected via the chain).

4. The walk stops if there are no empty sites available. Record the energy E and the length L of the chain.

Run $M = 1000$ such simulations, and plot histograms of the resulting energies E and L. What is the maximum E found? Plot the proteins for three selected simulations (e.g. with small, intermediate and large energy E). Finally, plot a heatmap (i.e. a 2D-histogram) of the resulting values for E and L. What does this tell us about the correlation of the two quantities?

4. Radioactive decay (6 points)

Imagine having a sample of $N(t)$ radioactive nuclei at time t. Simulate the decay of the sample by increasing the time t in discrete steps of Δt, and at each time count how many nuclei have decayed during the last Δt interval. The simulation quits when there are no nuclei left. The probability of a nucleus to decay per unit of time should be $\lambda = 0.03\,\mathrm{s}^{-1}$.

a) Repeat the simulation for $N(0) = 10$, 100, 1000, 10000 and 100000, and plot the $\log N(t)$ versus time t in a single plot. Also add the theory lines for the continuous decay model (for which $N \to \infty$ and $\Delta t \to 0$ is assumed), i.e. $N_{\mathrm{cont.}}(t) = N(0)e^{-\lambda t}$. Approximately, for which $\log N$ do the simulation results begin to appear stochastic (instead of approximating an exponential)?
(4 points)

b) Repeat the simulations and plotting as in part a), but using $\lambda = 0.3\,\mathrm{s}^{-1}$. What do you observe? Explain your finding.
(2 points)